Authenticated key exchange protocols in three parties

Benjamin Richard

ANNÉE 2017

UNIVERSITÉ DE RENNES 1

UNIVERSITE BRETAGNE LOIRE

THÈSE / UNIVERSITÉ DE RENNES 1
sous le sceau de l'Université Bretagne Loire

pour le grade de

DOCTEUR DE L'UNIVERSITÉ DE RENNES 1

Mention : Informatique

Ecole doctorale MATISSE

présentée par

Benjamin Richard

préparée à l'Institut de Recherche en Informatique et Systémes Aléatoires (IRISA), UMR n^{o}6074

Étude des protocoles d'authentification et de dérivation de clefs en 3 parties.

Thèse soutenue à Rennes
le 30 Aout 2017

devant le jury composé de :

Michel FERREIRA ABDALLA
Directeur de recherches, CNRS/ENS (rapporteur)

Refik MOLVA
Professeur, EUROCOM (rapporteur)

Frédéric CUPPENS
Professeur, IMT Atlantique Rennes (examinateur)

Jacques TRAORE
Chercheur, Orange Labs Caen (examinateur)

Pierre-Alain FOUQUE
Professeur, U. Rennes1 (directeur de thèse)

Gilles MACARIO-RAT
Chercheur, Orange Labs Chatillon (co-encadrant)

Maria Cristina ONETE
PostDoc, U. Rennes1 (co-encadrant)

Remerciements

En premier lieu, je tiens à remercier mon directeur de thèse, Pierre-Alain Fouque, pour son soutien, sa patience et son aide au quotidien. Au delà de ses indispensables qualités scientifiques dont j'ai pu bénéficier durant mes 3 années de thèse, je tiens à le remercier pour cette expérience, qui a su m'aider à grandir aussi bien d'un point de vue personnel que professionnel. Merci également à Maria Cristina Onete, de s'être jointe à cette thèse, pour son soutien au quotidien, et pour les nombreuses discussions qu'on a pu avoir ensemble. Je remercie aussi Gilles Macario-rat pour m'avoir encadré au sein d'Orange, d'avoir accepté de me suivre et de m'avoir fait confiance durant mon changement de sujet de thèse.

Je tiens ensuite à remercier l'ensemble des membres du jury d'avoir accepté de participer à ce jury. Un merci tout particulier à Michel Ferreira Abdalla et Refik Molva d'avoir accepté d'être les rapporteurs de cette thèse.

Je voudrai également remercier Jannick Dreier et Steve Kraemer, pour m'avoir accueilli à titre occasionnel dans les locaux du Loria à Nancy. Merci à eux pour leurs disponibilités et pour les échanges toujours très productifs et intéressants que nous avons eu ensemble.

Je tiens à remercier l'ensemble de l'équipe DSS d'Orange Labs pour m'avoir accueilli durant ces trois années de thèse. Un merci particulier à Pascal pour avoir essayé en vain de recadrer mes gouts cinématographiques, Todor pour avoir su mettre un peu de lumière dans le monde de la standardisation, Alex pour nos nombreuses parties de basket et nos débriefings de l'actualité sportive, et surtout Tiphaine, ma responsable de stage de 3ième préférée. Merci aux autres thésards de galère (Xiao, Maxime) à qui je souhaite bon courage pour la suite. Merci à tous les autres Xavier, Ghada, Said, Kahina, Michael, Matthieu ... d'avoir participé quotidiennement à la bonne ambiance de cette équipe. Une petite pensée aux anciens Laura, Georgian, Aurélien, Pierre, Amira, Nizar, Wafa avec qui j'ai pu avoir des conversations sérieuses et prolifiques, mais aussi des fou-rires très agréables. Merci aussi à tous les autres que je n'ai pas cité, mais qui étaient présent au quotidien :)

Merci à l'ensemble des membres de l'équipe EMSEC de l'IRISA pour leur accueil et pour la bonne ambiance qu'ils ont su instaurer au sein de nos bureaux. Je tiens à remercier l'ensemble du groupe des thésards made in EMSEC : Alban le roi des Spares au bowling, Baptiste mon élève de beaufitude préféré, Chen mon portugais préféré, Claire une des pilières de la pause café, Pauline la bordelaise de service, Florent et s'est "pourquoi pas" quand tu lui proposes à manger, et Raphaël le taulier du jeudi soir.

Merci aussi aux titulaires Patrick, le sérieux du Badminton, Stéphanie qui sait organiser des soirées barbecue à merveille et Adeline, pour son aide et son soutien. Petite pensée à l'ancien, Brice Minaud, sa présence au badminton et sa sympathie quotidienne nous manque à tous, et à la petite nouvelle, Angèle, qui je suis sur, s'épanouira à merveille dans cette équipe. Merci à toi Cyrille, le meilleur co-bureau qu'un thésard puisse espérer. Merci à toi mon ami pour toute l'aide que tu m'as apporté, nos fou-rires et nos chants. La meilleure ambiance était chez nous à ne pas en douter :p Ah oui merci, aussi d'avoir supporter les 30 degrés que j'imposai régulièrement dans notre bureau.

Je tiens à présent à remercier ma famille pour leur aide et leur soutien. Un remerciement tout particulier à Thomas et Charlotte. Vous m'avez offert plus qu'un canapé pour dormir durant ces trois années. Merci à vous mes parents, sans vous je ne serai pas l'homme que je suis aujourd'hui. Je vous dois beaucoup, et je vous en serai éternellement reconnaissant.

Merci à toi mon amour, Kun, d'avoir su répondre présent pendant les moments de doutes. Cette thèse nous a permis de nous rencontrer, et rien que pour cela cette thèse valait le coup d'être vécue. Cette thèse prend fin mais en ce qui nous concerne, ce n'est que le début...

Résumé en français

Protocoles AKE

La transmission de données personnelles entre deux entités à travers un canal non sécurisé comme Internet ou les voix radios au sein des réseaux mobiles, est l'un des enjeux majeurs en cryptographie. L'établissement d'un canal sécurisé permet d'échanger des données sensibles en garantissant leurs confidentialités et intégrités durant leurs échanges. Dans le but de garantir ses propriétés, l'utilisation de primitives cryptographiques telles que des algorithmes de chiffrement authentifié, est requise. Ses algorithmes nécessitent que les deux entités possèdent une clef secrète au préalable échangée. Par conséquent, un échange au préalable de clefs entre les deux entités doit être effectué entre les deux entités afin d'établir le canal sécurisé. L'établissement d'un canal sécurisé se repose sur l'exécution d'un protocole d'échange de clefs authentifiés, appelé AKE (Authenticated Key Exchange).

Propriétés de sécurité classiques. Les protocoles AKE exécutés entre deux entités doivent garantir l'établissement d'un canal sécurisé, à travers l'échange d'une ou plusieurs clefs temporaires (autrement appelées clefs de sessions), en considérant la présence potentielle d'un adversaire de type "Homme-du-Milieu" (MitM) capable d'observer et d'intercepter l'ensemble des communications entre ses entités. Cet attaquant peut être soit passif (espionnant les communications échangées) ou actif (capable de stopper, réordonner et injecter les messages de son choix). Considérant les adversaires de type MitM, les protocoles AKE doivent garantir les propriétés suivantes:

- La confidentialité des clefs de sessions échangées : un adversaire ne peut pas obtenir la moindre information à propos des clefs de sessions durant leur établissement.
- La confidentialité des secrets permanents : un adversaire ne peut pas obtenir la moindre information à propos des informations secrètes permanentes qui sont utilisées durant l'exécution du protocole.
- L'authentification : toutes les entités qui doivent être authentifiés durant le protocole AKE ne peuvent pas être impersonnifiées par un adversaire.
- La "non-rejouabilité" des messages : cette propriété requit la fraicheur des messages échangés durant l'exécution du protocole AKE.

Une propriété supplémentaire primordiale : le respect de la vie privée. La notion de vie privée peut prendre différent sens, comme le soulignent Pftzmann et Hansen [95]. Dans ce manuscrit, les notions qui seront utilisées, sont celles qui sont habituellement utilisées pour étudier les protocoles d'authentification et de dérivations de clefs, e.g., la confidentialité de données caractéristiques des utilisateurs, et de traçabilité. Les utilisateurs sont tous caractérisés par des données personnelles permanentes qui les caractérisent telles qu'un identifiant, des états, une localisation etc. Un premier niveau de respect de la vie privée des utilisateurs est garanti par une propriété consistant à

assurer la confidentialité de l'ensemble de ses données. Cette propriété est essentiellement capturée par l'incapacité pour un attaquant de type "Homme-du-Milieu" d'obtenir de l'information à propos de données caractéristiques des utilisateurs. La non traçabilité des utilisateurs est la notion la plus forte, qui en plus de considérer la confidentialité des données usagers, considère qu'un attaquant ne peut pas tracer un utilisateur, ce qui est potentiellement possible même sans avoir des informations sur ses données permanentes. En effet, par exemple, en reliant certaines informations temporelles, le profil de l'utilisateur pourrait être retrouvé.

Typiquement, les protocoles AKE requièrent la garantie de non traçabilité des utilisateurs, qui est globalement définit par le fait qu'aucun adversaire n'est capable de distinguer si deux exécutions distinctes du protocole ont impliqué le même client ou deux clients distincts. Dans ce manuscrit, la non traçabilité des utilisateurs sera étudiée en considérant les attaquants de type "Homme-du-Milieu". Une garantie plus forte, vus comme un anonymat total des utilisateurs, se base sur le fait que même un honnête participant ne peut pas tracer l'identité des autres participants au sein des communications. Une telle propriété pourrait être atteinte, dans une certaine mesure, si le protocole AKE reposait sur une authentification basée sur un groupe, comme par exemple les signatures de groupe.

Sécurité Prouvable. Étudier la sécurité des protocoles cryptographiques (plus précisément les protocoles AKE) est essentiel. De tels protocoles ont été définis sans nécessairement avoir une analyse de sécurité fiable. La sécurité prouvable est un outil générique et mathématiques pour étudier la sécurité des protocoles en considérant l'ensemble des adversaires potentiels définis au sein d'un modèle de sécurité. Ce modèle définit les capacités des adversaires et fournit une définition formelle des différentes notions de sécurité requises par le protocole étudiée. Définit durant les années 80, il y a globalement deux approches pour formaliser la sécurité : soit on utilise le modèle calculatoire soit le modèle formel. Bien que l'approche des méthodes formelles présente de nombreux avantages, un inconvénient important reste présent : il ne prend pas en compte la sécurité des primitives sous-jacentes ainsi que la taille des paramètres du protocole. Cela signifie que la preuve obtenue par un modèle formel n'est pas facile à quantifier à partir du moment où en pratique les primitives cryptographiques ne sont pas idéalisées et la taille des paramètres est indispensable pour évaluer la sécurité d'un protocole. Par conséquent, il nous semble préférable d'utiliser un modèle et des preuves calculatoires afin de pouvoir fournir une analyse de sécurité des protocoles aussi proche que possible de la réalité.

Contexte de nos recherches

Nos recherches se focalisent sur une utilisation spécifique des protocoles AKE, où un serveur mandataire est requis entre le client et le serveur. Un serveur mandataire peut être requis pour différentes raisons d'ordre pratiques telles que l'éloignement géographique entre le client et le serveur, le cout et la latence des communications, etc. La présence d'une telle troisième entité intermédiaire qui acte au sein des communication entre client et serveur impliquent différentes modifications à la fois au niveau de l'établissement du canal sécurisé mais aussi au niveau des communications au sein du canal sécurisé. Ce serveur mandataire est considéré comme un fournisseur local de services délégué par le serveur afin de gérer localement un certain nombre de services pour un certain nombre de clients. De plus, la présence d'un serveur mandataire implique de nouvelles considérations (propriétés et adversaire) sur la sécurité. En pratique, le serveur mandataire n'est pas forcément considéré comme une entité de confiance totale (comme le serveur principal). Ainsi, nous considérerons les serveurs mandataires comme des entités partiellement de confiance, qui peuvent être potentiellement malicieuses, i.e., qu'elles peuvent vouloir obtenir des informations supplémentaires ou acquérir une autonomie par rapport au serveur, supérieure à

ce qui lui a été autorisé. En ce qui concerne les protocoles étudiés dans ce manuscrit, nous considérerons que les serveurs mandataires malicieux ont pour objectif d'apprendre de l'information à propos des clients et serveurs, ainsi que de les impersonnifier sans en avoir l'autorisation. Le canal sécurisé est souvent établi entre le serveur mandataire et le client, avec l'aide du serveur principal. Dans notre manuscrit, nous nous sommes focalisés sur deux types de protocoles AKE en particulier : les protocoles AKA utilisés au sein des réseaux mobiles 3G et 4G afin de sécuriser la voix radio, ainsi que le protocole TLS dans le cas spécifique où celui-ci requière un serveur mandataire. Dans cette section, nous fournissons différentes informations concernant le contexte d'utilisation de ces différents protocoles.

Réseaux Mobiles. Au sein des réseaux mobiles 3G et 4G, les protocoles AKA (Authenticated Key Agreement) sont typiquement utilisés entre deux entités : un client mobile et son propre opérateur. Un des objectifs principaux pour les opérateurs est de fournir un certains nombres de services (appels téléphoniques, envoie de SMS, connections internet etc.) à l'ensemble de ses clients, peu importe leurs localisations. En pratique, une troisième entité, que l'on appellera *fournisseur local de services* est requise. Les communications entre le fournisseur local de services et un client mobile sont exécutées à travers un canal publique à base d'ondes radios, alors que les communications entre le fournisseur et l'opérateur se situent au sein d'un cœur de réseau sécurisé. Pour une question de sécurité, certains services ne peuvent être fournis qu'au sein d'un canal sécurisé. Par conséquent, il est indispensable de sécuriser les communications de la voix radio. C'est là que les protocoles AKA interviennent. Nous précisons qu'au sein des architectures 3G et 4G, un certain nombre d'entités passives actes durant l'exécution des protocoles AKA. Néanmoins, ces entités ne font que de la retransmission de messages sans modifier leurs contenus.

Le fournisseur local de services n'est pas toujours une entité de confiance, pour les clients mobiles et les opérateurs. D'un point de vue géographique, on considère que le client a deux possibilités : soit il se situe au sein du réseau mobile géré par son propre opérateur, soit au sein de celui d'un autre opérateur. Dans le premier cas, le fournisseur local de services est géré par l'opérateur du client, et dans ce cas, il est considéré de confiance par l'opérateur. Dans le second cas, quand le client est en situation d'itinérance, les services sont fournis par une entité locale gérée par l'opérateur local, qui est différent de l'opérateur du client. Par conséquent, dans le cas de l'itinérance, le fournisseur local de services ne peut être considéré que partiellement de confiance, malgré les différents accords d'itinérances établis entre les différents opérateurs. L'opérateur fait confiance aux fournisseurs local de services uniquement pour fournir certains services à ces clients en itinérance, mais il doit se prémunir d'un éventuel comportement malicieux de leurs parts, ayant pour objectif d'obtenir des informations secrètes concernant les clients mobiles. A contrario, les fournisseurs requièrent différentes informations temporelles, telles que des clefs de sessions indispensable afin de sécuriser la voix radio.

Délégation de livraison de services. Le protocole HTTPS (HyperText Transfer Protocol Secure) est utilisé afin de sécuriser les communications entre un navigateur web (le client) et un site Internet (le serveur), en utilisant le protocole TLS (Transport Layer Security). Ce dernier garantit la confidentialité et l'intégrité des données échangées. TLS est un protocole asymétrique exécuté entre deux entités, le client et le serveur. Si les deux entités sont éloignées d'un point de vue géographique, une certaine latence des communications apparait, i.e., le transfert des données sera lent quelques soient les possibilités de routage. Une des approches pour atténuer cette latence, consiste en la délégation de livraison de services, dénommée CDN (Content Delivery Network), nécessitant un réseau de serveurs mandataires locales à proximité des clients. Le serveur mandataire stockera et délivra les services aux clients qui sont à proximité à la place du serveur principal. L'architecture CDN implique d'avoir une totale confiance envers les serveurs mandataires, à partir du moment où l'ensemble des informations (notamment celles requissent pour

établir le canal TLS) requissent pour délivrer les services délégués par le serveur doivent être stockés au sein de ses serveurs mandataires. De plus, la présence d'une telle entité implique certaines problématiques génériques pour la divulgation inconditionnelle du trafic des clients vers les serveurs, et plus globalement, pour la sécurité globale des protocoles qui sécurisent les communications entre le client et le serveur. Les serveurs mandataires sont souvent vus comme indésirable d'un point de vue de la sécurité, ainsi que pour le respect de la vie privée des clients.

CloudFlare propose une variante aux CDNs, appelée Keyless SSL [101], qui améliore la latence et atténue certains risques, en particulier lorsque les serveurs principaux ne souhaitent pas que les serveurs mandataires possèdent et utilisent des informations secrètes longues durées (clefs secrètes etc.) en leurs noms. Dans ce cas, le protocole TLS est modifié et exécuté en tenant compte de cette troisième entité intermédiaire (le serveur mandataire) et de son potentiel malicieux. En effet, les serveurs mandataires ne sont pas nécessairement de confiance, et leurs possibles comportements malicieux doivent être pris en compte.

État de l'art

Sécurisation de la voix radio des réseaux mobiles. Les protocoles AKA, apellés UMTS-AKA et EPS-AKA, sont utilisés respectivement au sein des réseaux mobiles 3G et 4G afin de sécuriser la voix radio en établissant un canal sécurisé entre le client mobile et le fournisseur local de services. Pour cela, une authentification mutuelle des deux entités ainsi qu'un échange de clefs de sessions sont nécessaires. La sécurité des protocoles AKE, classiquement éxécutés entre deux entités, fut introduite par Bellare et Rogaway [40]. Sachant que les protocoles AKA requiérent une troisième entité et que celle-ci est active dans l'établissement du canal sécurisé, le modéle de sécurité classique fourni par Bellare et Rogaway [40] et l'extension proposée par Bellare, Pointcheval et Rogaway (BPR) [39] ne peuvent pas être directement importés pour modéliser la sécurité des protocoles AKA. Par conséquent, le manque d'un modèle de sécurité calculatoire complet et précis prenant en compte l'ensemble des caractéristiques des protocoles AKA, implique l'impossibilité de fournir une analyse de sécurité calculatoire de ses protocoles.

Cela fait plus de 15 ans que la première version du protocole AKA (UMTS-AKA) est utilisée, ce qui explique pourquoi une large littérature éxiste autour de la sécurité de ce protocole. Un important résultat sur les propriétés d'authentification mutuelle et sur l'indistinguabilité des clefs de sessions établies, a été proposé par Zhang, et Zhang et al. [**?**, 109]. Ils y décrivent différentes faiblesses remettant au cause ces propriétés de sécurité. Ces faiblesses sont basées sur la corruption des fournisseurs locaux de services et sur des attaques par rejeu. Ils proposent ainsi certaines contremesures. Malheureusement, ces dernières sont encore vulnérable à un certain nombre d'attaques et les contre-mesures proposées ne prennent pas en compte les considérations pratiques.

Les protocoles AKA sont instantiés avec deux ensembles d'algorithmes cryptographiques : TUAK and MILENAGE. Mise à part une étude de Gilbert [70] qui fournit une étude hors-contexte de la sécurité des algorithmes de MILENAGE montrant qu'ils fonctionnent globalement comme un mode compteur sur l'algorithme AES qui dérive successivement différentes clefs, il n'est pas clair que les propriétés prouvées d'indistinguabilité sont vraiment utiles pour garantir la sécurité des protocoles AKA. De plus, pour autant que nous le sachions, la sécurité des algorithmes TUAK n'a jamais été étudié.

Des preuves de sécurité des protocoles AKA ont été proposé dans plusieurs papiers [30, 109], spécialement quand ils sont instanciés avec les algorithmes MILENAGE. Les résultats les plus pertinents utilisent des outils automatiques de vérification formelle afin de proposer une preuve de sécurité. Ces outils se reposant sur des modèles formelles, ces preuves ne sont pas facile à quantifier, au vus des approximations établies autour des paramètres et algorithmes cryptographiques.

Nous considérons que seule une analyse calculatoire peut fournir une analyse de sécurité complète et fiable en considérant les différentes caractéristiques des protocoles AKA (primitives cryptographiques, tailles de paramètres, utilisation d'états etc.).

La majeure partie des papiers concernant les protocoles AKA se focalisent sur l'analyse du respect de la vie privée des clients mobiles. Trois principales attaques ont été établies : les attaques basées sur les IMSI catcher [62], IMSI paging [97, 98], et sur l'impersonification des fournisseurs locaux de services par corruption [**?**]. Ces attaques prouvent que durant l'éxécution des protocoles AKA, le niveau de respect de la vie privée des clients mobiles est insuffisant. De plus, plusieurs autres vulnérabilités e.g. [**?**, 30, 33, 87, 103, 108] montrent que les protocoles AKA ne respectent les notions de respect de vie privée standardisées par le 3GPP. De telles vulnérabilités rendent possibles l'espionnage et la surveillance de masse illicites, et plus spécialement donnent la possibilité à des attaquants d'identifier et de tracer les clients mobiles.

Les deux versions (3G et 4G) des protocoles AKA étant assez similaires, le protocole EPS-AKA est souvent vus comme aussi faible, d'un point de vue sécurité et respect de la vie privée des clients mobiles, que le protocole UMTS-AKA. A notre connaissance, aucune analyse ne compare la sécurité et le respect de la vie privée des clients mobiles de ses deux protocoles.

Établissement d'un canal sécurisé dans le cadre des délégations de livraison de services web.

Au vus des nombreuses utilisations et de son architecture non-orthodoxe, l'ensemble des versions du protocole TLS ont fait l'objet de nombreuses analyses de sécurité, pour n'en citer que quelqu'uns [24–28, 34, 36, 46–50, 54, 58, 61, 67, 69, 71, 74, 75, 79, 81, 83, 84, 89, 94, 105, 107]. La dernière version de TLS (1.3) est actuellement en cours de standardisation, et est conçue pour éviter les nombreuses faiblesses présentes dans les différentes versions du protocole. La dernière version de TLS 1.3 est disponible sur le lien suivant : `https://tlswg.github.io/tls13-spec/`.

Traditionnellement, les communications sécurisées sont établies en utilisant des protocoles d'authentification et d'échanges de clefs (AKE) comme détaillés dans les modéles de sécurité [39, 40, 56]. Néanmoins, TLS ne peut pas garantir la sécurité AKE traditionnelle à cause de certaines caractéristiques [73, 81] du protocole notamment du au chiffrement des derniers messages du protocole TLS. Par conséquent, une propriété de sécurité plus faible a été formalisé, denommé ACCE (Authenticated and Confidential Channel Establishment), qui fournit une spécification cryptographique précise pour TLS. Krawczyk, Paterson, et Wee [81] ont prouvé la sécurité de TLS 1.2 quand l'authentification mutuelle est requise. Les premières versions de TLS 1.3 ont aussi été analysés avec des modéles de sécurité proches du modéle ACCE [61, 82].

Les scénarios nécessitant une troisième entité intermédiaire active dans l'exécution du protocole TLS n'ont pas reçu autant d'attention que le protocole TLS exécuté classiquement entre deux entités. A notre connaissance, aucun modèle de sécurité n'a été proposé, adaptant le modèle de sécurité ACCE, pour un modèle de sécurité incluant les spécificités dues à la troisième entité.

Dans le contexte de la délégation de livraison de services web, deux variantes aux architectures CDN furent proposées: Keyless SSL [101] et mcTLS [91]. Ces deux variantes ont pour but de sécuriser les communications entre le client et le serveur de manière plus efficace que de sécuriser indépendamment les communications entre client et serveur mandataire, et entre serveur mandataire et serveur. Autant que nous le sachions, seulement un papier [101] a analysé la sécurité l'architecture Keyless SSL. Stebila et Sullivan ont fourni une description, une liste informelle des notions de sécurité que Keyless SSL doit garantir, ainsi qu'une étude de la performance de cette nouvelle architecture afin de voir l'amélioration de la latence par rapporte aux architectures CDN. Malheureusement, aucune analyse calculatoire ou formelle de sécurité n'a été proposé.

Une seconde architecture, appelée mcTLS, a été proposé afin de proposer une seconde alternative à base de serveurs mandataires. Dans cette variante, le protocole TLS est modifié afin que le client et le serveur puissent s'accorder explicitement sur la délégation des autorisations de lectures

et d'écritures des serveurs mandataires intermédiaires.

Contributions

Dans le but d'analyser la sécurité de l'ensemble de ses protocoles d'authentification et de dérivations de clefs, les modèles de sécurité doivent être établis en prenant en compte l'ensemble des caractéristiques de ces protocoles, et notamment l'utilisation active et la sécurité de la troisième entité intermédiaire. Dans un premier temps, nous fournissons un nouveau modèle de sécurité AKE qui considère la troisième entité intermédiaire actif dans les communications client-serveur, comme une important part du modèle. Les propriétés de sécurité de ses protocoles, au sein de ce nouveau modèle, sont établies en considérant à la fois les adversaires de type "Homme-du-Milieu", i.e. l'indistinguabilité des clefs ainsi que la non-impersonification des entités qui doivent être authentifié, et le potentiel comportement malicieux des fournisseurs de services locales, i.e. la confidentialité des clefs de sessions ainsi que le maintien d'une certaine dépendance de ses entités envers les serveurs principaux. Dans un second temps, un premier modèle de sécurité ACCE, généralisant le modèle ACCE en deux entités pour les protocoles qui font intervenir un serveur mandataire intermédiaire actif dans les communications client-serveur. Ce modèle de sécurité intègre différentes notions de sécurité, les classiques, i.e. l'authentification des différentes entités ainsi que la sécurité du canal établi, mais aussi deux autres propriétés provenant du serveur mandataire, qui garantissent que le serveur mandataire ne puisse pas être plus dépendant que ce que serveur lui autorise (*accountability* and *content soundness*).

Dans ce manuscrit, nous fournissons une analyse de sécurité de différents protocoles d'authentification et de dérivations de clefs en trois entités dans différents contextes : les protocoles AKA utilisés au sein des réseaux mobiles, et Keyless SSL utilisés au sein des communications entre un navigateur et un serveur web. Nous insistons sur le fait que de telles analyses sont indispensables vus que ces protocoles sont déjà implémentés et utilisés au quotidien. Nous fournissons une analyse de sécurité complète du protocole UMTS-AKA et nous prouvons que ce protocole atteint la sécurité AKE désirée à partir du moment où les fournisseurs locaux de services ne sont pas corruptibles. De plus, nous analysons la sécurité du protocole EPS-AKA afin de voir s'il garantit plus ou moins les mêmes notions de sécurité que le protocole UMTS-AKA. Nous utilisons une approche modulaire à base de réductions afin d'établir la sécurité de ses deux protocoles : dans un premier temps, nous prouvons la sécurité de ses protocoles est atteinte sous la condition que l'ensemble des primitives cryptographiques utilisées sont pseudo-aléatoires, et dans un second temps, nous vérifiions que les deux instantiations possibles (TUAK et MILENAGE) peuvent atteindre une telle propriété. Nous proposons (à notre connaissance) la première analyse calculatoire complète et rigoureuse des protocoles AKA et de leurs deux instantiations. Finalement, nous fournissons une analyse de sécurité du protocole Keyless SSL en capturant les différentes caractéristiques dues à la délégation de services à un serveur mandataire. Notre analyse souligne différentes faiblesses de sécurité et ainsi que des problématiques pratiques. Ainsi, nous proposons une nouvelle version de Keyless SSL qui garantit une meilleure sécurité dans le nouveau modèle 3ACCE.

Une propriété transverse à la sécurité mais indispensable au sein des réseaux mobiles est le respect de la vie privée des clients mobiles. Considérant les nombreuses faiblesses remettant en cause le respect de la vie privée des clients mobiles, notamment celles permettant la surveillance de masse et la traçabilité, durant l'éxécution des protocoles AKA, Van Den Broek *et al.* [104] et Arapinis *et al.* [31] ont proposé respectivement des variantes des protocoles AKA. Nous utilisons une approche calculatoire afin de prouver que l'ensemble de ses variantes échouent encore pour garantir le respect de la vie privée des clients mobiles. Nous proposons des améliorations qui garantissent l'ensemble des propriétés liées à la vie privée des clients mobiles définis par le 3GPP, en considérant les attaquants de type "Homme-du-Milieu". De plus, nous prouvons que nos variantes

sont optimales d'un point de vue de la sécurité et du respect de la vie privée des clients mobiles, à partir du moment où la structure globale de ces protocoles est maintenue. Dans ce sens, nos variantes sont optimales. Nos variantes prennent également en compte les problématiques pratiques liées au contexte des réseaux mobiles (coût élevé des communications au sein de l'équipement mobile et du cœur de réseau, la confiance limitée envers les fournisseurs locaux de services dans le cas de l'itinérance etc.) . Cependant, certaines exigences telles que les interceptions légales doivent également être prises en considération pour une éventuelle normalisation. Afin de répondre à ces différentes exigences additionnelles, deux variantes supplémentaires ont été proposées pour une éventuelle standardisation au sein des réseaux mobiles actuels (3G et 4G) et futurs (5G).

La majeure partie de ces résultats ont fait (font) l'objet de publications au sein de conférences et journaux.

Contents

Abbreviations & Notations

Abbreviations

The following abbreviations are used throughout this work:

2-(S)ACCE	2-party (Server) Authenticated and Confidential Channel Establishment
3GPP	3rd Generation Partnership Project
3-(S)ACCE	3-party (Server) Authenticated and Confidential Channel Establishment
Acc	Accountability
ACCE	Authenticated and Confidential Channel Establishment
AE	Authenticated Encryption
AEAD	Authenticated Encryption with Associated Data
AES	Advanced Encryption Standard
AK	Anonymity Key
AKA	Authenticated Key Agreement
AKE	Authenticated Key Exchange
AMF	Authentication Management Field
AN	Access Network
AS	Access Stratum
AuC	Authentication Center
AV	Authentication Vector
BS	Base Station
C	Client
CA	Certificate Authority
C.Imp	Client Impersonation
CK	Cipher Key
CDN	Content Delivery Network
CN	Core Network
CS (domain)	Circuit Switched
CS (property)	Channel Security
CSound	Content Soundness
C.Unlink	Client Unlinkability
DES	Data Encryption Standard
DH	Diffie Hellman
DHE	Diffie Hellman Ephemeral
DoS	Deny of Services
EA	Entity Authentication
eNB	evolved NodeB
EPC	Evolved Packet Core
EPS	Evolved Packet System
E-UTRAN	Evolved UTRAN
FBS	False Base Station
GGSN	Gateway of the GMSC

GSM	Global System for Mobile communication
GUMMEI	Global Unique MME Identifier
GUTI	Global User Temporary Identifier
IK	Integrity Key
IKE	Internet Key Exchange
IMEI	International Mobile Equipment Identity
IMSI	International Mobile Subscriber Identity
IND-CPA	Indistinguishability under Chosen Plaintext Attack
IND-CCA	Non-adaptive Indistinguishability under Chosen Ciphertext Attack
IND-CCA2	Adaptive Indistinguishability under Chosen Ciphertext Attack
INT-CPA	Indistinguishability under chosen PlainTeXT attack
INT-CCA	Indistinguishability under chosen CipherTeXT attack
IP	Internet Protocol
KDF	Key Derivation Function
K-Ind	Key-Indistinguishability
KSI	Key Set Identifier
HMAC	keyed-Hash-Message-Authentication-Code
HLR	Home Location Register
HSS	Home Subscriber Server
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
LAC	Location Area Code
LAI	Location Area Identity
LTE	Long-Term Evolution
MAC	Message Authentication Code
MB	Middlebox
MCC	Mobile Country Code
mcTLS	multi-context Transport Layer Security
ME	Mobile Equipment
MEA	Mixed Entity Authentication
MitM	Man-in-the-Middle
MME	Mobility Management Entity
MMEI	MME Identifier
MNC	Mobile Network Code
MSC	Mobility Switching Center
MSIN	Mobile Subscriber Identification Number
NAS	Non-Access Stratum
NMSI	National Mobile Subscriber Identity
Op	Operator
PDN-GW	Packet Data Network GateWay
PKE	Public-Key Encryption
PKI	Public-Key Infrastructure
PLMN	Public Land Mobile Network

PMSI	Pseudonym Mobile Subscriber Identity
PRF	PseudoRandom Function
PRG	PseudoRandom Generator
PRP	PseudoRandom Permutation
PS	Packet Switched
P-TMSI	Pseudonym TMSI
RAI	Routing Area Code
RNC	Radio Network Controller
RRC	Radio Resource Control
S	Server
SACCE	Server Authenticated and Confidential Channel Establishment
SGSN	Serving GPRS Support Node
SGW	Serving GateWay
SIM	Subscriber Identity Module
S.Imp	Server Impersonation
SLHAE	Stateful Length-Hiding Authenticated Encryption
Sound	Soundness
SSL	Secure Sockets Layer
St.conf	State Confidentiality
TLS	Transport Layer Security
TMSI	Temporary Mobile Subscriber Identity
UE	User Equipment
UID	Unique IDentifier
UMTS	Universal Mobile Telecommunication Network
UP	User Plane
USIM	Universal Subscriber Identity Module
UTRAN	Universal Terrestrial Radio Access Network
VC	Virtual Client
VLR	Visited Location Register

Notations

The following notations are used throughout this work:

$\oplus$	Bitwise exclusive-Or operation
$\mid$	Bitwise Or operation
$\&$	Bitwise And operation
$x \| y$	Concatenation between x and y
$\lvert v \rvert$	The bit size of v
$\lfloor v \rfloor_{i..j}$	The bitstring from position i to j of the initial bitstring v
$v \xleftarrow{\$} S$	v is chosen uniformly in the set S
$\mathbf{E_K}$	Encryption function with key K
$\mathbf{D_K}$	Decryption function with key K

Chapter 1

Introduction

Contents

1.1 AKE Protocols

Transmitting private data between two parties across an insecure channel as Internet or radio-frequency channel, is a fundamental goal in cryptography. The establishment of a secure channel permits to exchange sensitive data providing confidentiality and integrity services on the exchanged data. In order to provide these services, cryptographic primitives are required, e.g., authenticated encryption algorithms. These algorithms require both parties to have pre-exchanged a key, to be applied. Thus, an *handshake* of temporary/session key is required to establish the secure channel. This handshake is provided by Authenticated Key Exchange (AKE) protocol.

Classical Security Requirements. Two-party AKE protocols must guarantee the establishment of a secure channel, by a key agreement, even in the presence of a powerful Man-in-the-Middle adversary (MitM), capable of observing and intercepting all the communication across the network. The attacker may be either passive (merely eavesdropping on exchanged messages) or active (able to stop, reorder, or inject messages of his choice). These protocols must guarantee the following properties in the presence of such MitM attackers:

- The confidentiality of the exchanged session key: an adversary cannot obtain any information on the current session key during the agreement.
- The confidentiality of the long-term key: an adversary cannot obtain any information on the long-term key during the agreement or from the session keys.
- Authentication: all the entities which have to be authenticated cannot be impersonated by any adversary.

- The non-replayability of messages: this property requires the freshness of the exchanged messages during a fresh execution of the protocol.

Additional Paramount Requirement: User Privacy. The notion of user privacy can take many forms and meanings, as outlined by Pfitzmann and Hansen [95]. In this manuscript, the most relevant notions are those which usually encountered in secure authentication protocols. We will assume that lack of subject given users are associated with long-term personal data (an identifier, some user-specific state, a location etc.). A weak degree of privacy is captured by the requirement of confidentiality of user-specific data. This property is usually captured by the inability of a Man-in-the-Middle adversary to recover one or multiple pieces of user information. User untraceability is a stronger notion than confidentiality. Indeed, users may be *traced* even without knowing their full identifiers or distinguishing data. By linking metadata, the full profile of a user could be reconstructed. Typically, AKE protocols require the guarantee of user untraceability, which is defined in terms of adversary's ability of distinguishing whether two protocol executions involve the same user or two different users. In particular, an adversary cannot distinguish even one bit of any of the user's identifiers. In this manuscript, we consider user untraceability with respect to MitM attackers. A stronger guarantee, that of full anonymity, would require that even an honest participant cannot trace the identity of its communication party. This could be achieved to some degree if the authenticated key-exchange protocol relied on some group-based authentication, such as group signatures.

Provable Security. Study the security of the cryptographic protocols (more precisely the AKE protocols) is essential. Such protocols have been defined without necessary having a correct security analysis. Provable security is a generic and mathematics tool to provide the security of a protocol considering all the possible adversaries in a defined security model. This model defines abilities of adversaries and provides a formal definition of the security notions the protocol has to provide. Emerging during the eighty's, there are two main approaches to formalizing security: the computational model and formal model. Although the formal method approach has many advantages, one important disadvantage is that it does not give an exact reduction to the security of the underlying primitives, which means that the proof statement is not easy to quantify, it is hard to approximate the tightness of the proof, and the size of the parameters. Thus, a correct and fully computational cryptographic model and proof are preferable to formal verification. Our first work will be to define a specific security model as close to the real-world-context as possible.

1.2 Application Scenarios

Our researches are focused on a specific case of the AKE protocol, where a proxy is required between client and server. Such a proxy could be required for different practical reasons (geographic positions, costs, efficiency, etc.). The presence of a third (intermediate) participant into a two-way handshake causes modification in both the handshake and in the application layer protocols of the AKE protocol. This proxy entity is considered as a service provider where some client's services are delegated from the server to the proxy. Moreover, the presence of the proxy implies new security requirements and new adversary. Indeed, the proxy is not entirely trusted, it must be considered potentially malicious. For the protocols considered in this manuscript, we will want to prevent semi-trusted proxy from learning information about endpoints long-term state, and from impersonating the endpoints without authorization. The secure channel is often established between the semi-trusted intermediary party and the client, with the help of the server. In this work, we have focused on two protocols: the authenticated key agreement protocols in mobile networks, and the proxied handshake transport layer security protocol used for securing Internet traffic. In this section, we provide some context information on both these protocols.

1.2.1 Mobile Networks

In 3G and 4G mobile networks, the symmetric Authenticated Key Agreement (AKA) protocols are typically executed between two entities: a mobile client and its own operator. One of the main goals of the operator is to provide services to all its mobile clients no matter their location. In practice, a third entity is required, called a local *service provider*. Communication between service providers and mobile clients are executed across a public channel using air radio waves, while communication between service provider and operator are located in a secure core network. For some notable security reasons, some services have to be exchanged toward a secure channel. Thus, a secure channel has to be established between the mobile client and the local service provider. We note that in the 3G and 4G architectures, some other (passive) entities act in the AKA protocols. However, these devices act only as matters broadcasting messages to a specific entity without modifying them.

The local service provider is not always trusted by the two endpoints. The mobile client finds itself in one of two situations: either the client is located in a geographic area managed by its own operator, or in an area managed by another operator. In the first case, the local service provider is managed by the client's operator, and may thus be trusted. In the second case, when the client is abroad in a *roaming* situation, services are provided by a service provider affiliated with a different operator. Thus, in a *roaming* situation, the service provider cannot be consider trusted despite any roaming agreements established between the two operators. It is only trusted to provide services, but they must not learn the client's long-term secrets (known only to the client and the operator); by contrast, service provider has to learn short-term secret values, such as session keys, which are necessary for the transmission of the required services across the secure channel.

1.2.2 Content Delivery Services

HyperText Transfer Protocol Secure (HTTPS) establishes communication between web browsers (end-clients) and popular websites (origin-servers) using the well-known Transport Layer Security (TLS) protocol to guarantee data confidentiality and integrity services. TLS is a two-party asymmetric protocol executed between an end-client and an origin-server. If the two entities are situated geographically far away, latency appears, i.e. the data transfer will be slow involving extensive routing. An approach to reduce the latency, is the use of Content Delivery Networks (CDNs), which provide a network of proxies close to the client. The proxy or edge server, will store and deliver services on behalf of the origin-server, but from the clients proximately. The CDN architecture implies a total trust of the proxy since all information (notably the one requires to establish the secure TLS channel) requires to deliver client's services are delegated to the proxy. Moreover, the presence of such an entity implies some widely issues for the all-or-nothing disclosure of end user traffic to proxies and, more globally, for the global security of the handshakes protocols using them. These edge servers are sometimes viewed as undesirable in a security sense, notably for privacy concerns and the implied modification in a part of original Internet architecture.

CloudFlare proposed a variant of proxied TLS, called Keyless SSL [101] , which mitigates some risks, especially for "high-value" origin servers which may require some CDN-based performance boost but do not want the CDN to handle long-term private keys on their behalf. In this case, TLS has to be executed considering an active third entity (the intermediate) which is only partially trusted by the endpoints, and thus which could be considered as malicious.

1.3 State of the Art

1.3.1 The Authentication and Key Agreement in Mobile Networks

The AKA protocols, called UMTS-AKA and EPS-AKA, are used respectively in the 3G and 4G networks to establish a secure channel between the mobile clients and the service providers, requiring a mutual authentication and session keys agreement. An Authenticated Key Exchange security was introduced by Bellare and Rogaway [40] is guaranteed. Since the three-party setting and protocol features on the mobile client side, the classic security model provided by Bellare and Rogaway [40] and its extension Bellare-Pointcheval-Rogaway (BPR) [39] cannot simply be "imported" to model the security of these protocols. Thus, the lack of a precise computational security model, implies the impossibility to provide a complete computational analysis of the AKA protocols.

Since the first version of the AKA protocol (UMTS-AKA) has been used for more than fifteen years, there is a wealth of literature on the security of this protocol. An important result on the mutual authentication and key-indistinguishability properties of AKA was provided by Zhang and Zhang et al. [**?**,109]. They described attacks against both these properties based on service provider corruption and replay attacks, and proposed countermeasures to fix the weaknesses. Unfortunately, the proposed countermeasures are still vulnerable to some attacks and do not take into account practical considerations.

AKA protocols are instantiated with two sets of cryptographic algorithms: TUAK and MILENAGE. While Gilbert [70] provides an out-of-context security proof for the MILENAGE algorithms, showing they operate as a kind of counter mode in deriving key materials (MILENAGE runs AES multiple times as a one block to many blocks expansion function), it is unclear whether the proved indistinguishabililty properties are strictly useful to guarantee the security of the full AKA protocol. Moreover, as far as we know, the security of TUAK algorithms has never studied.

A security proof for the AKA protocols is given in few papers [30,109], especially when instantiated with MILENAGE. The closest results to a security proof use automated (formal) verification. Thus, the proof statement is not easy to quantify, making it hard to approximate the tightness of the proof and the size of the parameters. We only consider a computational analysis can provide a correct and complete security analysis of AKA, considering its various features (cryptographic primitives and parameters, etc.).

Most of the papers concerning the AKA protocols focus on the user privacy. Three main attacks in the literature, namely IMSI catcher attacks [62], IMSI paging attacks [97, 98], and impersonation by server-corruption [**?**], already proved that AKA does not offer the desired degree of client privacy. Additionally, further vulnerabilities, e.g. [**?**, 30, 33, 87, 103, 108] show that the AKA protocols do *not* attain even the basic-most privacy standardized requirements. Such vulnerabilities imply illicit eavesdropping and mass-surveillance, and especially point out the ability to identify and trace mobile clients.

Since both versions (3G and 4G) of the AKA protocols are similar, the EPS-AKA protocol is often assumed to provide only as much security of the UMTS-AKA protocol as possible. To the best of our knowledge, no complete security analysis comparing the security and privacy of the two protocols has been provided.

1.3.2 Transport Layer Security Handshakes in Content Delivery Services

Due to its importance and to its non-orthodox design, all the versions of TLS protocol were subject to numerous analyses, to quote just a few [24–28, 34, 36, 46–50, 54, 58, 61, 67, 69, 71, 74, 75, 79, 81, 83, 84, 89, 94, 105, 107]. The next version of TLS (1.3) is currently being standardized, and it is

designed to avoid many of the pitfalls in earlier versions of the protocol. The latest TLS 1.3 draft is available in https://tlswg.github.io/tls13-spec/.

Traditionally, secure communication is attained by using an authenticated key-exchange protocol as detailed in security models [39, 40, 56]. However, TLS cannot guarantee AKE-security due to subtle issues [73, 81] with the encryption of the last messages of the TLS handshake. Thus, a weaker model was formalized, called *authenticated and confidential channel establishment* (ACCE), which specifically provides a precise cryptographic specification for TLS. Krawczyk, Paterson, and Wee [81] have proved the security of TLS 1.2 with mutual authentication. Early drafts of the TLS 1.3 protocol have already been analyzed for security in models similar to ACCE [61, 82].

Three-party TLS scenarios have not received such attention as 2-party TLS protocol. No security model has been proposed adapting the 2-party ACCE security for the 3-party ACCE proxied handshake protocol.

We firstly refer to Keyless SSL as a handshake delegation protocol, providing a proxied variant of the TLS handshake. As far as we know, only one paper [101] is focused on the analysis of the Keyless SSL handshake delegation. Stebila and Sullivan provided the description and informal security requirements of Keyless SSL and studied the performances defining the efficiency against the content delivery services latency. Unfortunately, no computational and formal security proofs were given.

A different proxying scenario is considered in mcTLS [91], which modifies the TLS handshake so that clients and servers can both explicitly grant read/write permissions to proxies. In this manuscript, however, we only consider solutions that use unmodified TLS handshakes between clients and proxies.

1.4 Contributions

In order to analyse security of such protocols, security models have to be established taking into account features of protocols, notably the three-party setting. We firstly provide a new 3-party AKE security model which includes some specific security requirements. These requirements are formulated with respect to both Man-in-the-Middle (MitM) adversaries, i.e. key-indistinguishability and impersonation resistance security, and untrusted service provider, namely state-confidentiality and soundness. We also detail a first full 3-party ACCE (3ACCE) security model, generalizing of the 2-party ACCE security model, considering the three-party-setting as an important part of this model, for secure handshakes between client and server with the presence of intermediary entity. The security of such 3-party handshake is defined by four properties: the classic ones (entity authentication and channel security), and the two specific ones (accountability and content soundness).

In this manuscript, we will provide security analysis of 3-party handshake protocols in two different contexts: the AKA protocols in mobile networks and Keyless SSL in Content Delivery Networks. We stress that such analysis are important for any protocol deployed in real-life scenarios. We provide a formal security analysis of the UMTS-AKA protocol in its complete three-party setting. We prove that the UMTS-AKA protocol attains AKE-security as long as service provider cannot be corrupted. Additionally to this study, we analyze the security of the EPS-AKA protocol and see if it guarantees all or at least more requirements than the 3G version. We use a modular proof approach to establish the security of these protocols: the first step is to prove the security of AKA protocols assuming the pseudorandomness of the internal cryptographic functions. The second step proceeds to show that TUAK and MILENAGE guarantee the required pseudorandomness. We also provide (to our knowledge) the first complete, rigorous analysis of AKA protocols and the two instantiations. We provide a formal security analysis of the considered protocol Keyless SSL

and captures content delivery network CDN's characteristics. Such an analysis points out some security and practical weaknesses. Then, we propose a new design of Keyless SSL achieving better security in the 3ACCE considerations.

A paramount requirement in mobile networks is *user privacy* which is vital for mobile client. Considering the leak of client privacy in the AKA protocols due to the client-tracking attacks, Van Den Broek *et al.* [104] and respectively Arapinis *et al.* [31] proposed new variant of AKA protocols. We use the approach of provable security to show that these variants still fail to guarantee the privacy of mobile clients. We propose improvements of AKA protocols, which retain most of their structure and respect practical necessities, but which provably attains security with respect to malicious service provider and Man-in-the-Middle (MitM) adversaries. Moreover, we prove that any variant of the AKA protocols retaining the mutual authentication, cannot guarantee stronger privacy properties than the ones guaranteed by our variants. In this sense, our proposed variants are optimal. Our fix protocols also take into account practical considerations of mobile network key agreement. However, some requirements such as lawful interception must also be considered for a possible standardization and use in real-life scenarios. In order to respond to such considerations, two additional variants were proposed towards a possible standardization in current (3G and 4G) and future (5G) mobile networks.

Nearly all those results have been published either in conference papers or journal papers

1.5 List of Publications

- **Cryptographic Analysis of 3GPP AKA Protocol: ACNS 2016** *with Stéphanie ALT (DGA), Pierre-Alain FOUQUE (IRISA / Université de Rennes1), Gilles MACARIO-RAT (Orange Labs), Cristina ONETE (INSA/IRISA Rennes).*
- **Better Privacy for the 3GPP AKA Protocol: PETS 2016 and APVP 2016** *with Pierre-Alain FOUQUE (IRISA / Université de Rennes1), Cristina ONETE (INSA/IRISA Rennes).*
- **Towards 5G Authenticated Key-Exchange: the security and privacy of the AKA Protocol: RWC 2017** *with Pierre-Alain FOUQUE (IRISA / Université de Rennes1), Cristina ONETE (INSA/IRISA Rennes).*
- **Content Delivery over TLS: A Cryptographic Analysis of Keyless SSL: Euro S&P 2017** *with Karthikeyan BHARGAVAN (Inria de Paris), Ioana BOUREANU CARLSON (Imperial College London), Pierre-Alain FOUQUE (Université de Rennes 1/IRISA), Cristina ONETE (INSA/IRISA Rennes).*
- **Does the EPS-AKA protocol guarantee more security and client privacy than UMTS-AKA protocol?In submission.** *with Pierre-Alain FOUQUE (IRISA/Université de Rennes1), Cristina ONETE (INSA/IRISA Rennes).*

1.6 List of Patents

- **Procédé d'enregistrement d'un équipement utilisateur dans un réseau de communication: Orange Labs** *with Todor GABISHEV (Orange Labs) and Gilles MACARIO-RAT (Orange Labs).* Patent n° FR1659583 has been filed for the INPI in October 2016.
- **Procédé d'authentification mutuelle entre un équipement utilisateur et un réseau de communication : Orange Labs** *with Todor GABISHEV (Orange Labs) and Gilles MACARIO-RAT (Orange Labs).* Patent n° FR1659585 has been filed for the INPI in October 2016.

Chapter 2

An Overview of Architecture and Security to Mobile Networks

Contents

2.1 Mobile Networks Architecture

Mobile networks emerged in the late 80s, and have since become essential to modern-day life. Over 7 billion subscribers regularly use mobile networks today, for purposes such as sending messages, making calls, and Internet navigation. Every ten years, mobile networks take a generation leap, expanding in terms of new architectures and services. The second generation of mobile networks (2G) was introduced in the early 90s. The most widely-deployed version of 2G architectures is the Global System for Mobile Communication (GSM). It revolutionized mobile communications as it replaced analogue signals by digital technology, offering new services, such as MMS privileges (pictures, movies etc.). With an overage flow rate of 9.6 Kbits per second, 2G networks were the first to provide fast voice- and data-transfer.

The third generation of mobile networks (3G) were introduced at the beginning of the twenty-first century. The most common version is the Universal Mobile Telecommunication System (UMTS). Another version called CDMA-2000 was however proposed in parallel in the North of America and Asia. The UMTS version was introduced since the seven main standard development organizations wish to propose a common unique international norm. These seven entities compose the 3rd Generation Partnership Project (3GPP), which allows for global international roaming for mobile subscribers. A new network architecture is provided, split into three parts:

the user equipment, the access and core network. With UMTS networks, 3GPP increases system capacity voice transfer for improving the high flow rate multimedia services. The maximal theoretical flow rate is 1.920 Mbits per second, which is much more efficient compared to GSM networks. In 2008, 3GPP defined the fourth generation of mobile networks. The complete system is denoted LTE/EPC, where LTE (Long-Term-Evolution) is the new radio technology and EPC (Evolved Packet Core) is the new core network. The latter relies only on Internet Protocols to protect the communication in the core network. In general the 4G system is referred to as EPS for Evolved Packet System. 4G networks propose a better flow rate notably for data transfers (the download speed is 100 Mbits/s and the upload speed is 50 Mbits/s), reduces latency, and offers better interoperability with 2G and 3G. In this chapter, we present the different architectures and security mechanisms of the UMTS and EPS mobile networks.

2.1.1 The UMTS Mobile Network Architecture

A mobile network operates on radio waves ranging in frequency between 900 and 2100 MHz. The UMTS architecture is specified in technical specifications 23.002 [17] and 25.301 [19]. For 3G mobile networks, 3GPP introduced a new network architecture, split into three main parts. The first of these is the *User Equipment* (UE), consisting of a Mobile Equipment (ME) and a secure element (USIM). The second 3G element is the *Access Network* (AN) denoted UTRAN for Universal Terrestrial Radio Access Network, which consists of several base stations (NodeB) and Radio Network Controllers (RNC). The last 3G component is the *Core Network* (CN), which allows communication between operators. The global architecture is described in Figure 2.1 and further detailed in the following sections.

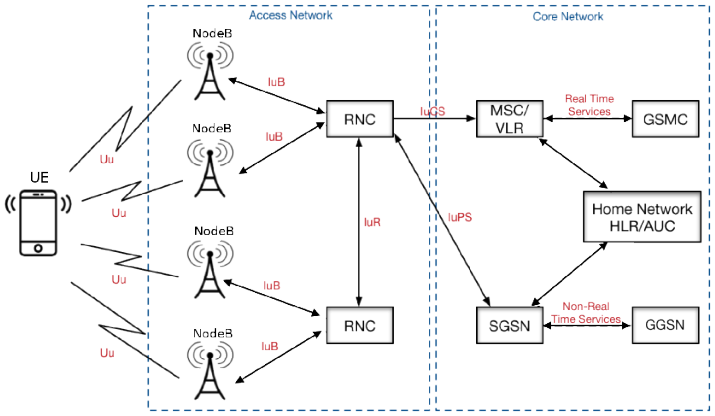


Figure 2.1: 3G architecture overview. The communication interfaces and domains names are outlined in red.

User Equipment. The User Equipment (UE), also called terminal, has a radio connection used to reach the network and services of the operator it subscribes to, across the access network UTRAN. 3G user equipments are composed by two main elements: a physical element, called Mobile Equipment (ME) and a secure element (USIM). ME can be either a smartphone or a laptop computer with a mobile broadband adapter. It is mainly manufactured by equipment vendors and not operators. It is divided into two distinct groups: the mobile termination which globally manages the radio features and the sent data formatting, and terminal equipment which notably manages the communication and order requests to the secure element. The Universal Subscriber Identity Module (USIM) is the secure element in 3G networks and replacing the SIM cards used in 2G networks. The mobile equipment is almost a neutral entity, i.e., not constructed for a specific operator, contrary to USIM cards, which are made for a specific operator and stores all the data

and features related to the operators.

Access Network. The Access Network in UMTS is called the Universal Terrestrial Radio Access Network (UTRAN). The access network allows the subscriber equipment to connect to the core network. Moreover, it is responsible for ensuring several other functionalities, such as the security of the radio infrastructures (the confidentiality and integrity of exchanged data), subscriber mobility (geographic position-estimation), and the management of radio resources for the subscriber communication and synchronization. The UTRAN network contains two types of elements: base stations and radio network controllers. In order to allow coverage for a maximal territory, the latter is split into a multitude of cells. Each cell represents a geographic area in which all the communications are managed by a base station, called NodeB. A base station can manage one or several radio cells. The range of the NodeB depends on the size of radio cells, the existence of natural obstacles (mountains, plain, relief, town, topography), the density of population in managed cells, etc. The second element of an access network is the Radio Network Controller (RNC), which controls radio transmission to and from the base station. RNC is the subscriber's to the core network. In UMTS architectures, the radio channel between the client and the RNC is protected. The base station also manages other services, such as handover and macro-diversity. For more details, we refer readers to [92].

Core Network. The Core Network (CN) in UMTS ensures communications between operators. It manages the routing of communications (data and voice) to and from the external networks. It consists of all equipments and data that can be used to guarantee the subscriber identification, location updates, security management, etc. The CN is divided into the home- and the serving-network. The Home Network (HN) contains the long-term information of all the operator's subscribers. The Serving Network (SN) contains the temporary information of all the subscribers in a roaming situation. The roaming relates to the ability for a subscriber to have access to some services as phone call or SMS/MMS privileges, when it is located in a geographic area unmanaged by its own operator. The HN is mainly based on two main entities: the Home Location Register (HLR) and The Authentication Center (AuC). The HLR contains for each subscriber, all the long-term public information (permanent identifier, phone number, etc.), as well as private data (long-term keys and security algorithms). The same entity also keeps track of the identity of the visitor location register. The second network entity AuC inter-operates with the HLR to use the stored data of subscribers. It generates authentication messages and session keys, which will be used in the SN to ensure authentication and the confidentiality/integrity of data across the radio channel.

When a packet is sent from the RNC to the core network, it is managed either by the Mobility Switching Center & Visited Location Register (MSC/VLR) or by the Serving GPRS Support Node (SGSN), depending on whether the packet is related to a real-time services or to non-real-time services. Real-time services concern the phone conversations (video telephony, video games, multimedia applications etc.) and process the voice and signalisation traffic from the radio network controller requiring a high-flow-rate. Non-real-time services are less sensitive to the time of transfer; this is the case of web browsing, use of e-mails, management of video games, etc. These entities are in charge of the routing in a network, inter-connection with other external networks and different services (location, handover, roaming etc.). They manage a specific area for specific services. We note that two entities cannot manage the same area for the same service. The Gateways of the MSC (GMSC) and the SGSN (GGSN) manage the connexion with the external networks.

2.1.2 The EPS Mobile Network Architecture

The new network 4G is called EPS or LTE/EPC. It is composed by a new access network denoted LTE for Long-Term-Evolution and a new core network denoted EPC for Evolved-Packet-Core. The EPS architecture is specified in technical specifications 23.401 [14] and 36.300 [16]. It only

supports data services implying the merge from the non-real and real time services management.

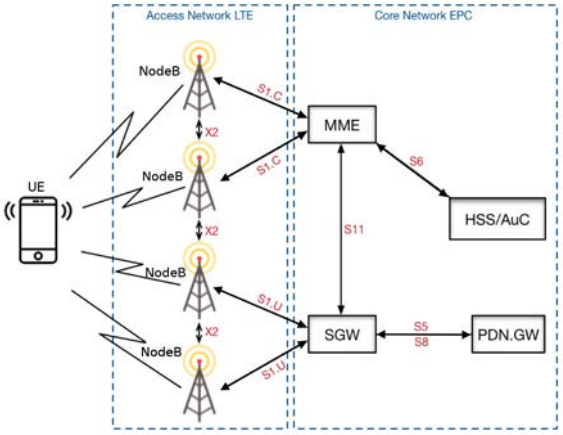


Figure 2.2: 4G architecture overview.

It provides IP connectivity between user equipment and the packet data network, and supports all the different generation of access network. This architecture is detailed in Figure 2.2.

User Equipment. 4G networks reuse the USIM cards involved in 3G networks. Only the access network and core network are evolved in the 4G networks.

The Access Network. The Long Term Evolution of 4G (LTE) also called the Evolved Universal Terrestrial Radio Access Network (E-UTRAN) is an extension of the 3G UTRAN architecture. The main goals of this access network is to provide a higher data and flow rate, lower latency, an enhancement of the quality of services and the interoperability with the older third generation of mobile networks [21]. Contrary to the UTRAN architecture which is composed by both NodeB and RNC, the LTE access network is based on an unique type of entities, called *eNodeB*. That efficiently reduces the latency. The eNodeB notably secures the radio communication from and to user equipments providing encryption and integrity services as well as the generation of session keys.

The Core Network. The Evolved Packet Core network (EPC) is based on a simplified network architecture. It consists of the Mobility Management Entity (MME), a Home Subscriber Server (HSS), the Serving GateWay (SGW) and the Packet Data Network GateWay (PDN-GW). By communicating with the HSS, the MME will obtain subscriber information for the base station notably initiating paging and the authentication of the mobile device. The HSS acts as an extension of the 3G home location register, and contains all types of subscriber information, such as permanent identifier, authentication data and MME identity to which the UE is connected. As in the case of 3G networks, an authentication center denoted AuC, is attached. The SGW routes and forwards user data packets from mobile equipment elements towards base stations. This gateway completes the interface towards E-UTRAN and ensures the interoperability to other networks. The PDN-GW component must guarantee the connection between the EPC and external IP networks, routing packets to and from packet data networks.

2.2 Mobile Networks Security Considerations

Secure communications between mobile subscribers and their associated operator networks require security in both access and core networks. In this section, we will detail how a secure channel is established in the access network and why the communication across the core network can be considered as secure.

2.2.1 UMTS Security Considerations

3GPP proposes a straight evolution of GSM networks in the release from 1999 and the specification TS.33.102 [1] has been proposed for the security of UMTS networks. The security requirements were formalized to take into account for the new services (electronic business, pictures, videos sharing etc.) provided by 3G networks to mobile subscribers. Such transmission required a much stronger security than guaranteed by GSM networks.

UMTS security overview. 3GPP in the technical specification [13], gives an overview of the UMTS security architecture in the figure 2.3 detailed below:

- Access Network Security (A) provides users with secure access to UMTS services and protect against (passive) attacks on the radio access link.

- Network Domain Security (B) protects against (passive) attacks on the wire line network and allows nodes in the provider domain to exchange signalling data securely.

- User Domain Security (C) provides secure access to USIM cards in the mobile stations.

- Application Domain Security (D) allows the secure messages exchange between applications in the user and in the provider domain.

- The visibility and security configuration allows the user to observe whether a security feature is currently in operation and if certain services depend on this security feature.

We recall the 3G networks propose a new cellular architecture splitting the network into two parts: the access and the core networks. The main idea was to separate the roles of radio-network operator and service operator. In the following sections, we detail the security of both parts. Additionally to the security of the networks, the interoperability with the 2G was indispensable as explained in the technical specification TS.31.900 [3]. Beyond the necessary infrastructures implying new entities and additional structures and how the security during the interoperability is, we consider that the main problematic is the ability for an adversary to force a subscriber to communicate over an older interface (2G) exploiting its weaknesses. This consideration will be not detailed in this section, and we refer readers [3] for more details.

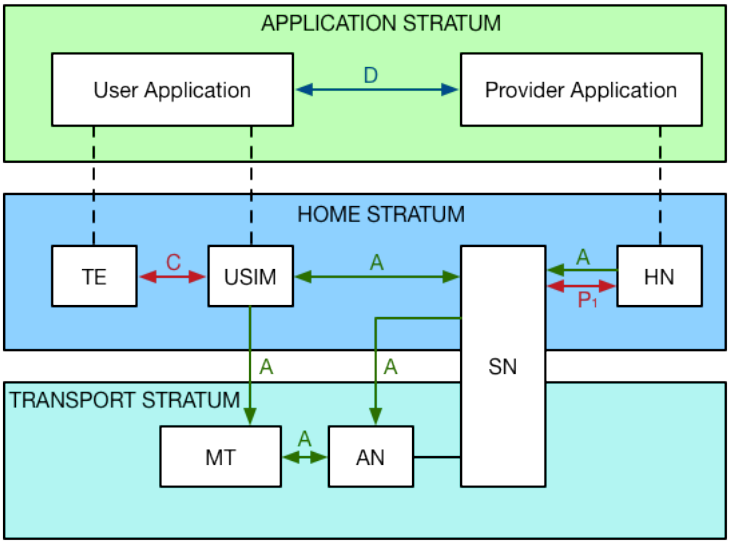


Figure 2.3: UMTS security overview.

Access Network Security. In the UMTS architecture, the user mobile equipment communicates with the core network, more specifically its own operator, via a radio-frequency channel in the radio access network. In the context of mobile networks, mobile services are granted to clients over a secure channel. Such a secure channel is established by a security set-up procedure. We detail this procedure in Figure 2.4.

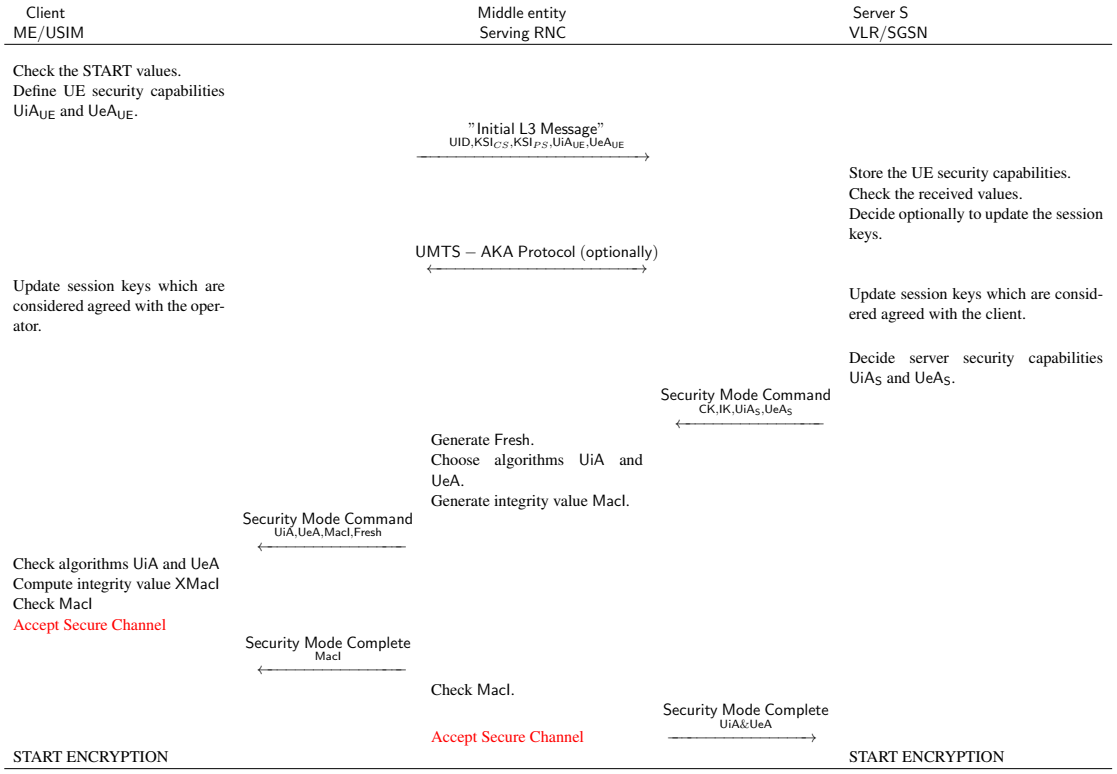


Figure 2.4: The set up 3G procedure.

This procedure is stateful. The client stores, from a previous connection, the parameters consists to the t-uples $(\mathsf{Start}_{\mathsf{cs}}, \mathsf{Start}_{\mathsf{ps}}, \mathsf{Threshold}, (\mathsf{CK}, \mathsf{IK}, \mathsf{KSI})_{\mathsf{ps}}, (\mathsf{CK}, \mathsf{IK}, \mathsf{KSI})_{\mathsf{cs}})$. At the beginning for the first connection, all the counters are initialized depending the client's operator, and no session keys are set. The values $\mathsf{Start}_{\mathsf{cs}}, \mathsf{Start}_{\mathsf{ps}}$ are counters for the Circuit-Switched (CS) and Packet-Switched (PS) domains, and $\mathsf{Threshold}$ is the maximal allowed value of these counters, as initialized by the network. These values are stored and maintained in the mobile equipment, not in the USIM card during an ongoing connection. These values provide a means to limiting the lifetimes of the used security key. The session keys (the cipher key CK and integrity key IK) which ensure the confidentiality and integrity messages exchanged across the radio access network (UTRAN). The key set identifier, denoted KSI, is associated with the retained session keys pair. This 3-bit value allows to identify the session keys pair. The value $'111'$ is restricted and expresses that the keys are obsolete and need to be reallocated.

The subscriber requires mobile services and needs, hence to establish a connection with the network. At the beginning of a new attempted connection, the user equipment checks its own START values and updates the key set identifier KSI to the value $'111'$ if necessary (i.e., if one of the START values exceeds $\mathsf{Threshold}$). The sets of encryption and integrity algorithms respectively denoted $\mathsf{UiA}_{\mathsf{UE}}$ and $\mathsf{UeA}_{\mathsf{UE}}$ which UE would like to use to protect the future communications. These parameters are transmitted, at the very beginning of the connection, in plaintext. The client then initiates the communication by sending *user identification* message to the VLR/SGSN

via the RNC. This message[1], also called an IMSI attachment is a tuple $(\mathsf{KSI}, \mathsf{IMSI})$. An IMSI is the International Mobile Subscriber Identity which is a permanent identifier, unique per each customer. We note that a temporary identifier, previously exchanged with the server, can replace the IMSI value to guarantee user identity confidentiality. The key set identifier permits to either reuse session keys or reallocate these keys from authenticated and key-agreement protocol, called UMTS-AKA. This protocol provides a mutual authentication and a session-key-establishment between the network and the user equipment, and we detail it, with the user identification, in Section 2.3.1. If this protocol is carried out, the Start values will be reset to 0.

After agreeing on session keys (if necessary), the VLR/SGSN chooses its own sets of encryption and integrity algorithms, listing them in order of preference. Then, it sends them in a *security mode command* to the RNC. This message also includes the fresh cipher and integrity keys $(\mathsf{CK}, \mathsf{IK})$ which will be used to protect the communication between the RNC and UE. Receiving the security mode command from the VLR/SGSN, the RNC selects the encryption and integrity algorithms that will be used later, by comparing its own preferences to the sets received from the user.

Next, the RNC generates a random value denoted Fresh, which will ensure the freshness of the connection. From this point onwards, the integrity of the exchanged messages can be guaranteed. The RNC sends a message to the client including the following values: the chosen algorithms UeA and UiA, the random value Fresh, and a MAC value called Macl, which ensures integrity. MAC value is computed over the random value Fresh, the algorithm UiA and the key IK. Upon receiving such a message, the client compares the list of received algorithms it proposed at the beginning. Then, it checks the MAC value Macl, computing a value XMacl over the received UiA and the Fresh value. If all the verifications are successful, then the client accepts the secure channel establishment. Otherwise, the procedure is dropped.

If the agreement is accepted, the UE sends to the RNC a message called *security mode complete*, which includes the same MAC value Macl. The RNC receiving this message, verifies this message and sends a message to the VLR/SGSN confirming the agreement and including the name of the chosen algorithms UiA and UeA. At this step, the procedure is finished.

The communication between the user equipment and the RNC can now be ciphered/deciphered using the algorithm UeA and the cipher key CK. The blockcipher KASUMI [9, 10] and the stream cipher SNOW-3G [11, 12] are the only algorithms standardized by 3GPP for the UMTS networks, and thus must be included in all options provided for the UiA and UeA. We note that since 2005, following the attacks of Biham [51], Dunkelman [63] and Bar-On [35], KASUMI is no longer considered secure.

Core Network Security. The global core network is divided into several core networks managed by each operator. The core network traffic is not necessarily ciphered, but every operator may implement protection mechanisms for their own network transmission link. Most of the threats to the communication in the core network are similar to those affecting the application layer. The most important security problems are:

- Denial of Services (DoS): the goal of DoS attacks is to ensure a long-term impossibility for a service to be received or provided, by generating notably disturbance traffic.. In this sense, DoS attacks do not directly threaten the security of communication, although it is often used as a means of breaking privacy.

- Social engineering: often disregarded by cryptographic analysis, social engineering is one of the most efficient attacks when looking for specific secret data, such as e.g., a PIN number.

[1] We note that this message is, more general by an *Initial L3 Message* and can be different than an IMSI attachment. Indeed, this message is generic request and can be for example, a request of location update or a paging response.

- Sniffing: eavesdropping is another common attack that aims to collect subscriber information, allowing even a passive adversary to monitor great amounts of connections without detections.
- Spoofing: in this attack, adversary uses the IP address associated with a legitimate entity, thus communicating with other users, i.e., the attacker replaces the real receiver or sender in a communication attempt.
- Session Hijacking: the adversary hereby obtains access to mobile communication services for incoming and outgoing calls of a specific subscriber.

Though at first similar to the GSM architecture, the UMTS infrastructure subsequently evolved in terms of the core network structure, using the internet protocol, denoted IP, as its main network-layer protocol used. We usually associate UMTS architectures with this improved core network security. The IPSec [77] protocol is the basic tool in protection of network domain traffic, providing confidentiality and integrity of the communication in the IP layer. Complementary to IPSec, the Mobile Application Part Security protects messages in the application layer. Consequently, we will assume that the core network is secure, providing confidentiality and integrity for all communications. We refer readers to [59] for the security of the IPSec protocol.

2.2.2 EPS Security Considerations

The EPS security is a slight improvement of UMTS networks security. Keeping the secure elements and each security mechanisms, 3GPP has to update the requirements to the new architectures and security contexts. The last version of the specification containing the security requirements of the EPS networks have been published in 2016 in [8].

EPS security overview. The level security of this new network is upgraded by proposing some modifications in particular in the access network. Most security features of the UMTS generation were maintained, we recall these here:

- User permanent identity confidentiality using temporary identifiers.
- Basic features and concepts of the authenticated key-exchanged protocol (mutual authentication, session keys establishment, use of authentication vectors, sequence number).
- Confidentiality services for data and signalling over the radio channel.
- Integrity services on signalling data.
- Agreement of the algorithms UeA and UiA.
- Use of the same 3G standardized algorithms except KASUMI.
- Global security architecture in the core network.
- Use of the USIM secure element in the user equipment.

We note that USIM cards are re-used in 4G networks for business reasons, so as to avoid having to replace the secure elements of older phones. Another requirement is the interoperablilty with the UMTS and GSM networks. The interoperability with the GSM infrastructures is restricted and possible only for insecure services. The technical specifications [8, 21] detail several security requirements of the EPS networks which are different from those in UMTS, notably:

- The deployment of a new authenticated and key-agreement procedure called EPS-AKA, including new key hierarchy.
- The use of new temporary identifiers.
- Standardization of new algorithms notably replacing KASUMI.
- Resistance to attacks for evolved base stations.
- Integration of security for non-3GPP access.

The attacker model considered in EPS networks includes adversaries whose goal is to deny service to either the client or the network, but also attacks seeking to break the authentication, confidentiality, or integrity properties at each layer, and finally threats to privacy. The capabilities of the adversary range from passive eavesdropping to active network access.

The evolved base stations, called eNodeB, are endpoints for major EPS security mechanisms, i.e. entities in which the secure data is stored rather than just playing a relaying role like NodeB in 3G networks. These entities may be located in exposed, unprotected locations, such as building roofs, public, and outdoors places, which cannot be totally controlled by operators. Thus, evolved base stations are assumed vulnerable.

Evolved base stations store session keys previously exchanged by the EPS-AKA protocol to secure the radio communication toward the access network. Consequently, the stations require element to support the secure storage of sensitive data. Such data includes some long-term keys used for the authentication for the core network. The keys must never be used outside the secure element, allowing the base station and its secure element support the execution of sensitive functions, guaranteeing confidentiality and integrity services. The security of base stations is detailed in [8].

Access Network Security. All the security functionalities in the LTE access network between the user equipment and the evolved base stations are defined in Technical Specification TS 33.401 [8]. As in the UMTS networks, protecting the radio-link in the access network is the most important point. This is the most vulnerable link to attacks. The security of the access network is split into four parts: the confidentiality of the user and devices identities, mutual authentication between the user equipment and the network, confidentiality, and the integrity of the signalling data.

One of the main goals is protecting personal user information which would allow passive adversaries to identify him. As in 3G architectures, 3GPP continues to only consider security with respect to passive adversaries. They focus on the confidentiality of one part of the permanent identity, called the Mobile Subscriber Identification Number (MSIN), which is unique for each user. They additionally consider the confidentiality of the international mobile equipment identifier (IMEI), which is a new requirement in this generation of mobile networks. The mutual authentication between the user equipment and the network is still required in the 4G network by means of a variant of the authenticated key agreement, called EPS-AKA. This protocol is similar to its homologue in UMTS networks, but includes a new session keys hierarchy, a new temporary identifier, and a means for the client to check the local network is identity. We detail this protocol in Section 2.3.2.

Following the agreement of session keys between ME, BS and MME, the confidentiality of signalling and user data is required. Integrity is only required on the signalling data NAS and RRC, in order to protect against replays.

The security set-up protocol is designed to exchange session keys and negotiate the algorithms to be used to provide confidentiality and integrity over the radio channel. Similarly to its UMTS equivalent, this protocol is split into three phases: a tracking area update request, an optional authenticated key-agreement protocol, and a secure mode command. The user equipment presents

its security capabilities and algorithms by an attachment procedure, sending Tracking Area Update request messages to the base station. This request is forward to the MME. After choosing a set of existing session keys (or generating new ones by means an authenticated-key-agreement protocol), the three entities compute the same session keys. The MME and the base station independently choose this own list of algorithms to be used with the UE. The MME is responsible to the AS-level algorithms negotiation, while the base station responsible to the NAS-level algorithms negotiation.

Core Network Security. All the security functionalities included in the evolved packet core network allow to securely exchange data. The radio link in the access network, for which the air interface is used, is considered to be the most vulnerable part of the network systems. The core network is similar to its 3G equivalent, even although the evolved packet core is not split into packet and circuit switched. The core network still has to guarantee the mutual authentication, confidentiality and integrity of all the exchanged data between each entities. As specified in TS.33.210 [5], the protocol IPSec is used [77] to guarantee confidentiality and integrity services. For the mutual authentication, the protocol IKEv2 (Internet Key Exchange version 2) [76] is used, relying on certificates and the establishment of trust. The version of the required X.509 certificates, are detailed in the technical specification TS 33.310 [18].

New EPS Standardized Algorithms. As in all the mobile network generations, the system should be flexible, meaning that new algorithms can be introduced and outdated ones can be removed, without major modifications in the global infrastructures. In response to the attacks [51, 63] presented on the KASUMI block cipher, the EPS infrastructure has been introduced with a new block cipher, implementing AES [66] with counter mode [64]. This cipher suite is denoted as 128-EEA2. An extension to the SNOW 3G algorithm in UMTS, denoted as 128-EEA1 allows both traditional 128-bit encryption and an optimal 256-bit mode. For integrity, AES-based cipher suites use CMAC, whereas SNOW 3G is extended to provide both 128 and 256-bit output. The former method is labelled 128-EIA2, whereas the latter is called 128-EIA1. For AES, similar to the specific case in the ciphering mode, ETSI-SAGE (which is the organization who studies the standardized algorithms) chose the Cipher-Based Message Authentication Code (CMAC) mode [64]. These algorithms may be chosen arbitrarily and independently to protect the AS and NAS layers, as opposed to the UMTS case, when a single suite must be chosen for both layers. We note that from the release 11 of the technical specifications, a third algorithm appears called ZUC [22] was standardized to be used in China (since chinese authorities require the use of algorithms made in China). Finally, 4G networks require unlike UMTS, the use of key derivation functions for the new hierarchy and is detailed later in Section 2.3.2, this KDF is based in the Keyed-Hash Message Authentication Code (HMAC) mode [80] using the cryptographic hash function SHA-256 [65]. We note that the EPS-AKA protocol uses its own algorithms detailed in Section 2.3.3.

2.3 The AKA protocols

Although the first version of the authenticated-key-agreement protocol (AKA) was introduced for GSM networks, that version bears little resemblance to what AKA is today. In 2G, no requirement of authentication exists, and the output and input sizes of cryptographic parameters and functions are small.Consequently, we choose to describe only the authenticated-key-agreement (AKA) protocols used in the 3G and 4G architectures.

2.3.1 AKA in the 3G Networks: The UMTS-AKA Protocol

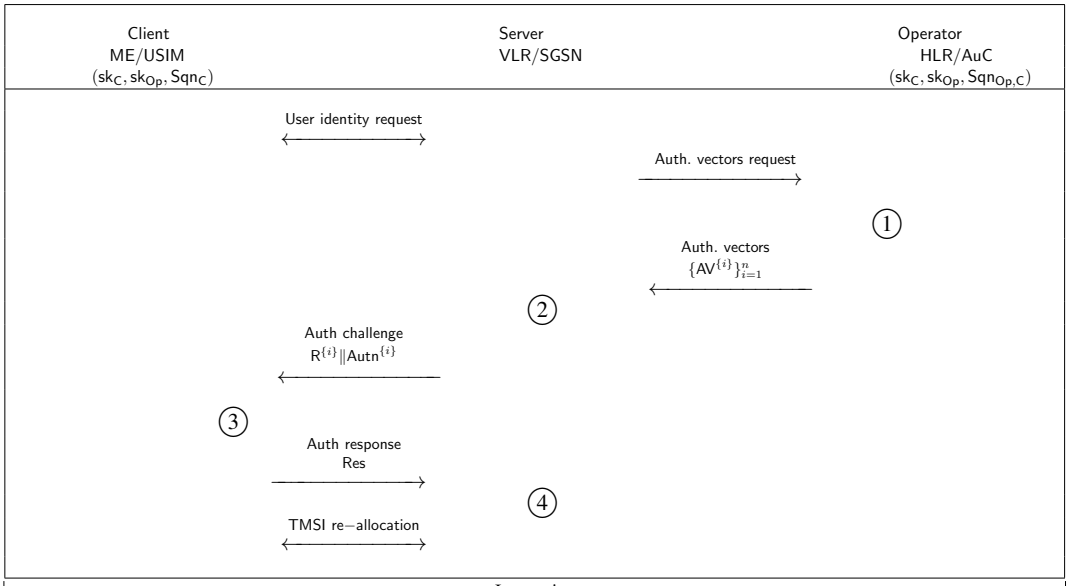
The UMTS-AKA protocol features two main *active* actors: the client/subscriber (in 3GPP terminology ME/USIM) and the server (denoted either VLR/MSC or SGSN [2]). The third, only selectively-active party is the operator (denoted HLR/AuC). The tripartite set-up of AKA was notably meant for roaming, for which the server providing mobile coverage is not the client's operator, and may be subject to different legislations and vulnerabilities. Thus, although the server is trusted to provide services to a client across a secure channel, it must not learn long-term sensitive information about either clients or their home operators. Using the server as a mere proxy would ideal; however, communications between servers and operators are (financially) expensive.

Clients C and operators Op use both the client's secret key sk_C and the operator's secret key sk_Op. Subscribers to the same operator all share the operator's own secret key, in practice a 256-bit integer. This value is not directly stored on the phone, but rather an intermediate value, obtained by deriving the operator key, the subscriber key and several constants, is embedded in the secure element of the phone, i.e., the USIM card. Thus, whereas this value enters in all future runs of the cryptographic algorithms, it is never stored in clear on the user's mobile. The client and operator also keep track of sequence numbers Sqn_C (resp. $\mathsf{Sqn}_{\mathsf{Op},\mathsf{C}}$), updated after each successful authentication (by a simple, predictable procedure, e.g. incrementing them by a fixed value). If these states are too far apart, the client prompts a re-synchronization. The three parties: clients, servers, and operators, also know the client's permanent identifier IMSI. Clients and servers must keep track of tuples (IMSI, TMSI, LAI), the last two values forming a unique temporary identifier, which is updated at every session. This value is included to avoid replay attacks, since subscribers cannot generated (pseudo)random values from USIM cards. We note that, the next generation of SIM cards will be able, to generate such values. However, a main consequence of the lack of client-side randomness is that freshness must be guaranteed only using one random value rather than two. This is done by using a sequence number which UMTS-AKA is stateful. Both ME/USIM and the HLR/AuC keep track of counters, denoted respectively Sqn_C and $\mathsf{Sqn}_{\mathsf{Op},\mathsf{C}}$; these sequence numbers are meant to provide entropy and enable network authentication (from HLR/AuC to ME/USIM). Technically, one can view the user's sequence number as an increasing counter, while HLR/AuC keeps track of the highest authenticated counter the user has accepted.

The UMTS-AKA protocol uses a set of seven functions: $\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3, \mathcal{F}_4, \mathcal{F}_5, \mathcal{F}_1^*, \mathcal{F}_5^*$. The first two are used to authenticate a MAC value, proving that both participants know the same subscriber key sk_C and the same operator key sk_Op. The $\mathcal{F}_1$ algorithm is called *the network authentication function*. As its name implies, it allows the subscriber to authenticate the network. Furthermore, this function provides the integrity of the data used to derive keys (in particular authenticating the random, session-specific value R). Algorithm $\mathcal{F}_2$ is called the *subscriber authentication function* and it provides a client-authentication functionality. The following three algorithms, $\mathcal{F}_3, \mathcal{F}_4, \mathcal{F}_5$, are used as *key derivation functions*, outputting respectively a cipher key (CK), an integrity key (IK), and an anonymity key (AK), all derived on input the subscriber key sk_C, the operator key sk_Op, and the session-specific random value R. Notice that the master key sk_C is only known to HLR/AuC and ME/USIM, but not to the intermediate entity VLR, which is the server. The last key, AK, is used to mask the sequence number Sqn, but it is not part of the session keys. Its function is to blind the value of Sqn, since the latter may leak some information about the subscriber. In order to ensure that no long-term desynchronization of sequence numbers occurs, the UMTS-AKA protocol provides a re-synchronization procedure between the two participants, in which the user forces a new sequence number on the backend server, using the $\mathcal{F}_1^*$ and $\mathcal{F}_5^*$ algorithms to authenticate this value much in the same way that the client has authenticated its

[2] As explained in section 2.1.1, the server could be either a SGSN or a VLR/MSC. In our explanation for a easier lecture, we only denote the server as a VLR

own sequence number and random value. Another crucial design choice is that of the TUAK and MILENAGE functions. $\mathcal{F}_1$ and $\mathcal{F}_1^*$ are different and independent, as are $\mathcal{F}_5$ and $\mathcal{F}_5^*$. This ensures security even with resynchronizations. Figure 2.5 details the challenge-response structure of the UMTS-AKA procedure.



Instructions:

Client	Server	Operator
③: Compute AK using $\mathsf{R}^{\{i\}}$. Recover $\mathsf{Sqn}^{\{i\}}$ (from AK). Check $\mathsf{Mac_S}$ value. If $\mathsf{Sqn}^{\{i\}} \in (\mathsf{Sqn_C}, \mathsf{Sqn_C} + \Delta)$: Compute: $\mathsf{CK} \leftarrow \mathcal{F}_3(\mathsf{sk_C}, \mathsf{sk_{Op}}, \mathsf{R}^{\{i\}})$, $\mathsf{IK} \leftarrow \mathcal{F}_4(\mathsf{sk_C}, \mathsf{sk_{Op}}, \mathsf{R}^{\{i\}})$, Set $\mathsf{Res} := \mathcal{F}_2(\mathsf{sk_C}, \mathsf{sk_{Op}}, \mathsf{R}^{\{i\}})$. Update $\mathsf{Sqn_C} := \mathsf{Sqn}^{\{i\}}$. Else *re-synchronization*	②: Store $\{\mathsf{AV}^{\{i\}}\}_{i=1}^n$. Choose $\mathsf{AV}^{\{i\}}$ one by one in order. Then it sends the related challenge. ④: $\mathsf{Res} \stackrel{?}{=} \mathsf{Mac_C}$.	①: For each $i = 1, \ldots, n$, compute: Generate $\mathsf{R}^{\{i\}}$. Compute: $\mathsf{Sqn}^{\{i\}} \leftarrow \mathsf{inc}(\mathsf{Sqn_{Op,C}})$ $\mathsf{Mac_S^{\{i\}}} \leftarrow \mathcal{F}_1(\mathsf{sk_C}, \mathsf{sk_{Op}}, \mathsf{R}^{\{i\}}, \mathsf{Sqn}^{\{i\}}, \mathsf{AMF})$, $\mathsf{Mac_C^{\{i\}}} \leftarrow \mathcal{F}_2(\mathsf{sk_C}, \mathsf{sk_{Op}}, \mathsf{R}^{\{i\}})$, $\mathsf{CK}^{\{i\}} \leftarrow \mathcal{F}_3(\mathsf{sk_C}, \mathsf{sk_{Op}}, \mathsf{R}^{\{i\}})$, $\mathsf{IK}^{\{i\}} \leftarrow \mathcal{F}_4(\mathsf{sk_C}, \mathsf{sk_{Op}}, \mathsf{R}^{\{i\}})$, $\mathsf{AK}^{\{i\}} \leftarrow \mathcal{F}_5(\mathsf{sk_C}, \mathsf{sk_{Op}}, \mathsf{R}^{\{i\}})$, $\mathsf{Autn}^{\{i\}} \leftarrow (\mathsf{Sqn}^{\{i\}} \oplus \mathsf{AK}^{\{i\}}), \mathsf{AMF}, \mathsf{Mac_S^{\{i\}}}$. $\mathsf{AV}^{\{i\}} := \mathsf{R}^{\{i\}}, \mathsf{CK}^{\{i\}}, \mathsf{IK}^{\{i\}}, \mathsf{Autn}^{\{i\}}, \mathsf{Mac_C^{\{i\}}}$, with $\mathsf{Sqn_{Op,C}} = \mathsf{Sqn}^{\{i\}}$. End For.

Figure 2.5: The UMTS-AKA Procedure.

The UMTS-AKA protocol proceeds in several subparts. Over the *insecure channel*, client and server perform the *user identification* step. At the end of this step, the VLR must establish the IMSI of the client. An *authenticated-key-exchange* is executed between the three entities: the ME/USIM, the VLR and HLR/AuC. Finally, the server and client allocate a new temporary identifier over the *new secure channel*.

User Identification

This procedure starts when the user equipment switches on. In order to identify to the server , the client's mobile equipment receives a *user equipment request* and responds to the VLR, in clear text, with a user identifier UID. This value can be either the permanent IMSI or a temporary TMSI, which is a value exchanged between the server and the subscriber during a previous session for which both entities were mutually authenticated. More precisely, the TMSI is only used in the CS domain and another temporary identifier, denoted Packet-Temporary Mobile Subscriber Identity (P-TMSI) in the PS domain. Since the two identifiers have a similar structure, we will

only consider the temporary identity TMSI. Note that all permanent and temporary identifiers are stored in the USIM card. The user identification is summarily depicted in Figure 2.6. The permanent identifier IMSI of a client consists of the following values:

- The Mobile Country Code (**MCC**) consists of three digits. The **MCC** uniquely identifies the origin country of the mobile subscriber.
- The National Mobile Subscriber Identity (**NMSI**) consists of two values. The first one, the Mobile Network Code (**MNC**) consists of 2 or 3 digits (depending the value of the **MCC**). This value identifies the home network of the mobile subscriber. The second value is the Mobile Subscriber Identification Number (**MSIN**) which identifies the mobile subscriber within the Public Land Mobile Network (**PLMN**).

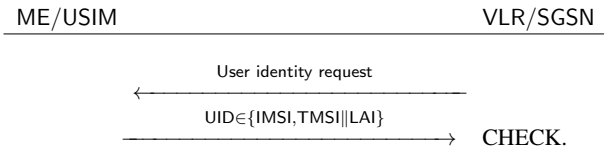


Figure 2.6: User identification by the permanent identity.

The TMSI may not exceed 15 digits.The allocation of Mobile Country Codes is administered by the ITU-T, more specifically in the recommendation E-212. The allocation of the **NMSI** is the responsibility of each administration of each country. The same Mobile Network Code should not be assigned to more than one **PLMN**. In a roaming situation (i.e. when the client is located in a foreign **PLMN**), the value **NMSI** is analyzed for information transfer. These TMSI are exchanged in order to guarantee the uniqueness of the user equipment request during following sessions. In practice, the IMSI is used either for the first session or when the serving network cannot retrieve the IMSI from the temporary identity. Then, the VLR forwards the IMSI of the subscriber to the HLR/AuC. Since the TMSI has only local significance (i.e. within a VLR and in the area controlled by a VLR, or within an SGSN and the area controlled by an SGSN), the structure and encoding of it can be chosen by agreement between operator and manufacturer in order to meet local needs. A TMSI is a local number and its construction is specified in the technical specification 23.003 [7]. The TMSI consists of 4 bytes. It can be coded using a full hexadecimal representation. In order to avoid the double allocation of TMSIs after a restart of an allocating node, some parts of the TMSI may be related to the time when it was allocated or contain a bit field which is changed when the allocating node has recovered from the restart. The TMSI is (re)allocated only in ciphered form as we will detail after. The network must never allocate a TMSI with all 32 bits equal to 1 (this is because the TMSI must be stored in the USIM, and the USIM uses 4 octets with all bits equal to 1 to indicate that no valid TMSI is available). The temporary identifier TMSI is never used alone. A Location Area Identity, denoted LAI (and Routing Area Identifier (RAI) for the P-TMSI) is added to the temporary identity to allow a serving network to retrieve a unique identity IMSI associated to the temporary data. This value can be used only in a specific given area: the TMSI is always accompanied by the location area identification to avoid ambiguity. The LAI consists of the three following elements:

- A Mobile Country Code, which is the same 3-bit value included in the permanent identifier.
- A Mobile Network Code, which is the same 3-bit value included in the IMSI.
- A Location Area Code, which is a 8-bit value identifying a location area within a PLMN.

In addition, the Routing Area Identifier (RAI) also includes a routing area code. The UMTS-AKA protocol starts with the identification of the ME/USIM to VLR. At first, the mobile equipment receives a user request from the VLR and then responds in cleartext, with a pre-exchanged TMSI as detailed previously.

The VLR, which represents the serving network, manages a data base of correspondences between the IMSI and the TMSI values. A new TMSI must be allocated at least every time the location updating procedure is used, as soon as a TMSI is used, it needs to be replaced. At that point, the mobile station de-allocates the old value and allocates the fresh temporary identity required from the VLR. We note that this value needs to be store in non-volatile memory together with the associated LAI. So they are not lost when the ME/USIM switches off. When the ME/USIM receives a *user identity request* from the VLR, it sends its current TMSI||LAI. When it receives its value, the VLR verifies if the LAI matches the current ME/USIM. If that is not the case, the VLR starts a *Local* TMSI *Unknown Procedure*. Otherwise, it tries to recover the corresponding permanent identity using its database to accept the identification. If such a value cannot be recovered, the VLR sends a *Permanent Identity Request* to the ME/USIM, which answers with its IMSI. All these flows are exchanged in cleartext.

Local TMSI **Unknown Procedure.** If the LAI does not match the VLR, a *Local TMSI Unknown Procedure* is executed by finding the VLR that issued the used TMSI, i.e. the VLR denoted VLR_0 corresponding to the sent LAI, the current VLR may recover the IMSI correponding to that (TMSI, LAI) tuple or a "error message" which in turn issues a *Permanent Identity Request* to the mobile station. This request is also used when the user registers for the first time in a serving network. This protocol is detailed in Figure 2.7.

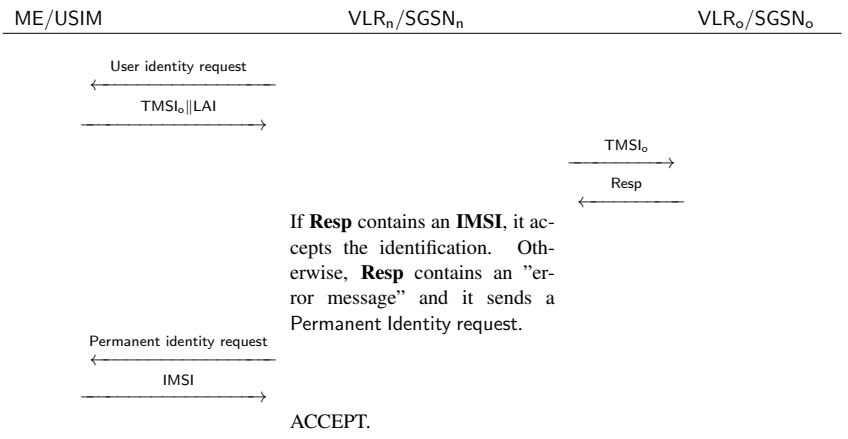


Figure 2.7: Local TMSI Unknown Procedure

Challenge-Response.

At the end of the identification phase, the VLR has been able to recover the subscriber's IMSI, and thus its associated operator. In the challenge response phase, the VLR sends an authentication-vectors request including the permanent identifier of the client to its related operator. Upon receiving the IMSI, the HLR/AuC generates a new sequence number Sqn and an unpredictable variable R. By using the subscriber's key sk_C and the corresponding operator key $\mathsf{sk}_{\mathsf{Op}}$, it then generates a list of n unique authentication vectors AV composed of five values: R, Mac_C, CK, IK, Autn. For every authentication vector, the sequence number is updated. The update procedure depends on the chosen method. The specifications feature a first method which does not take into account the notion of time, and which basically increments by 1 the most significant 32-first value of the sequence number. A second and third subsequent methods feature a time-based sequence num-

ber update based on a clock giving universal time [13]. The authentication vector is generated as follows:

$$\begin{aligned}
\mathsf{Mac_S} &\leftarrow \mathcal{F}_1(\mathsf{sk_C}, \mathsf{sk_{Op}}, \mathsf{R}, \mathsf{Sqn}, \mathsf{AMF}),\\
\mathsf{Mac_C} &\leftarrow \mathcal{F}_2(\mathsf{sk_C}, \mathsf{sk_{Op}}, \mathsf{R}),\\
\mathsf{CK} &\leftarrow \mathcal{F}_3(\mathsf{sk_C}, \mathsf{sk_{Op}}, \mathsf{R}),\\
\mathsf{IK} &\leftarrow \mathcal{F}_4(\mathsf{sk_C}, \mathsf{sk_{Op}}, \mathsf{R}),\\
\mathsf{AK} &\leftarrow \mathcal{F}_5(\mathsf{sk_C}, \mathsf{sk_{Op}}, \mathsf{R}),\\
\mathsf{Autn} &\leftarrow (\mathsf{Sqn} \oplus \mathsf{AK}) \| \mathsf{AMF} \| \mathsf{Mac_S},
\end{aligned}$$

where $\mathsf{Mac_S}$ is the message authentication code of the network to the subscriber, $\mathsf{Mac_C}$ is the message authentication code of the subscriber to the network and AMF is an administrative authentication and key management field (which is a known, public constant). This 16-bit value is not strictly standardized ans its usage depends to the related operator. Some examples as the lifetime of the session keys are detailed in the technical specification 33.102 [13].

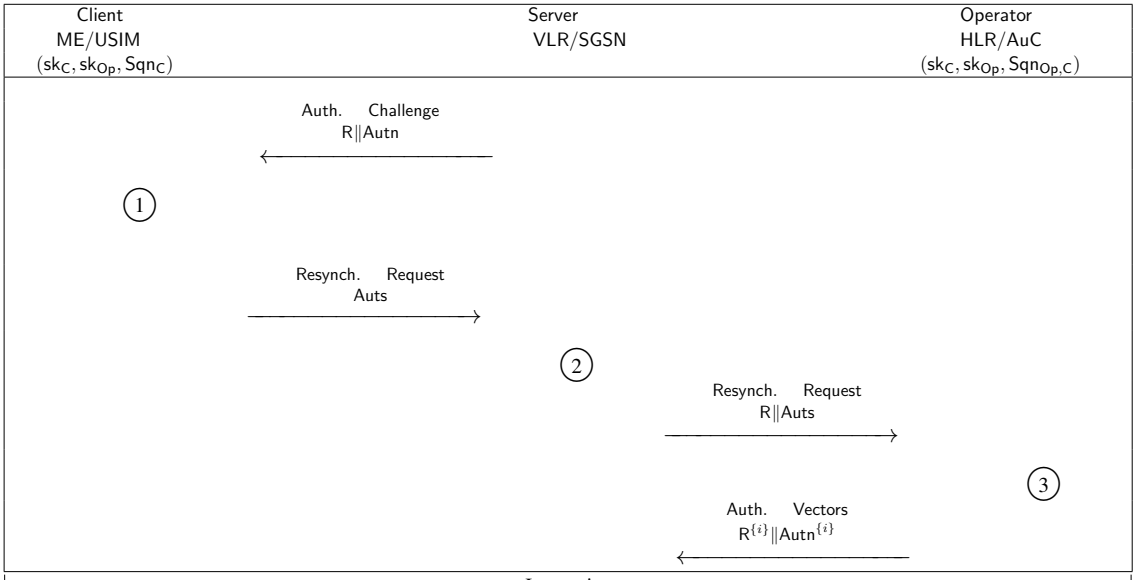
The HLR/AuC sends a batch of the authentication vectors AV to the VLR. This batch may be restricted to a single authentication vector (notably when the client is in a roaming situation). If the batch contains multiple authentication vectors, the operator will not be required in each session and which reduces the communication in the core network (between service provider and operator) which is financially expensive.

Upon the reception and storage of these vectors, when the server initiates an authentication and key agreement, it selects the next authentication vector from the ordered array and stores $\mathsf{Mac_C}$ and the session keys CK and IK. Then, it forwards $(\mathsf{R}, \mathsf{Autn})$ to ME/USIM. Note that the session keys $(\mathsf{CK}, \mathsf{IK})$ are not used during the key agreement; the additional key AK used to mask Sqn is independent of the other keys, thus preventing circularities in the security proof. The key AK is *not* used later to secure the channel. These choices facilitate our analysis and proofs.

The ME/USIM verifies the freshness of the received authentication challenge. To this end, it recovers the sequence number by computing the anonymity key AK, which in its own turn depends on three values: $\mathsf{sk_C}$, $\mathsf{sk_{Op}}$, and the received R. Then, the user verifies the received $\mathsf{Mac_S}$ by computing $\mathcal{F}_1(\mathsf{sk_C}$, $\mathsf{sk_{Op}}$, R, Sqn , AMF) with the received R and Sqn values. If Mac value does not verify, the user sends an *authentication failure* message back to the server, and the user abandons the procedure. In case the execution is not aborted, the ME/USIM verifies if the received Sqn value is in a correct range with respect to its stored $\mathsf{Sqn_C}$ value[3]. If the Sqn is out of range, the user sends a *synchronization failure* message back to the server, which triggers a *re-synchronization procedure*, depicted further in Figure 2.8. The $\mathsf{Mac_S}$ value does not only ensure the integrity, but also the authentication of the network by ME/USIM. If the two previous verifications are successful i.e., if the received authentication challenge is fresh, the network is authenticated by the ME/USIM. Then, the ME/USIM computes CK, IK, and $\mathsf{Res} \leftarrow \mathcal{F}_2(\mathsf{sk_C}, \mathsf{sk_{Op}}, \mathsf{R})$. To improve efficiency, Res, CK, and IK could also be computed earlier, at the same time that AK is computed. Finally, the user sends Res to server. If $\mathsf{Res} = \mathsf{Mac_C}$, the server successfully authenticates the ME/USIM. Otherwise, the VLR will initiate an *authentication failure* report procedure with the HLR/AuC. Note that the verification of the sequence number by the ME/USIM will cause the rejection of any attempt to re-use an authentication challenge more than once.

The Re-synchronization Procedure.

[3]The sequence number Sqn is considered to be in the correct range relatively to $\mathsf{Sqn_C}$ if and only if $\mathsf{Sqn} \in [\mathsf{Sqn_C}, \mathsf{Sqn_C} + \Delta]$, where Δ is defined by the operator.



Instructions:

ME/USIM	VLR	HLR/AuC
①: Compute the value AK with the value R. Recover Sqn with AK. Check if the received Mac_S is correct. Check if Sqn is in the correct range w.r.t. the stored Sqn_C. Re-synchronization Procedure: Compute $AK^* \leftarrow \mathcal{F}_5^*(sk_{Op}, sk_C, R)$ and $Mac^* \leftarrow \mathcal{F}_1^*(sk_{Op}, sk_C, R, Sqn_C, AMF)$. Forge $Auts = (Sqn_C \oplus AK^*)\|Mac^*)$.	②: Add the R value.	③: Compute the value AK^* with the value R. Recover Sqn_C with AK^*. Check if the incremented $Sqn_{Op,C}$ is in the correct range related to Sqn_C. If it is not the case and if only the received Mac^* is correct, then $Sqn_{Op,C} \leftarrow Sqn_C$. Otherwise, it aborts the procedure. It sends a new list of authentication data vectors initiated with $Sqn_{Op,C}$.

Figure 2.8: The re-synchronization procedure of UMTS-AKA protocol.

The re-synchronization procedure is used when the subscriber detects that the received sequence number is not in the correct range, but that it has been correctly authenticated. The single goal of this procedure is the re-initialization of the sequence number which does not immediately imply any mutual authentication or key agreement (rather the procedure triggers a new authentication attempt). The ME/USIM sends an *synchronization failure* message, consisting of a parameter Auts, with $Auts = (Sqn_C \oplus AK^*)\|Mac^*$ with $Mac^* = \mathcal{F}_1^*(sk_{Op}, sk_C, R, Sqn_C, AMF)$ and $AK^* = \mathcal{F}_5^*(sk_{Op}, sk_C, R)$. The $\mathcal{F}_1^*$ algorithm is a MAC function with the additional property that no valuable information can be inferred from Mac^* (in particular this function acts as a PRF). Though similar to $\mathcal{F}_1$, the $\mathcal{F}_1^*$ algorithm is designed so that the value Auts cannot be computed relying on the output of $\mathcal{F}_1$. Furthermore, the anonymity key AK^* generated by the client at resynchronization is obtained via the $\mathcal{F}_5^*$ algorithm, rather than by $\mathcal{F}_5$, even if the same random value R is used. Upon receiving a synchronization failure message, the server does not immediately send a new user authentication request to the ME/USIM, but rather notifies the HLR/AuC of the synchronization failure, sending the parameter Auts and the session-specific R. Note that, since VLR does not have the keys sk_{Op} and sk_C, it cannot verify the validity of Auts. When the HLR/AuC receives this answer, it creates a new batch of authentication vectors after checking the Auts value. Depending on whether the retrieved, authenticated Sqn indicates that the HLR/AuC's sequence number is out of range or not, the backend entity generates vectors starting either with the last authenticated sequence number, or from the authenticated sequence number sent by the user. More precisely, the HLR/AuC retrieves Sqn_C by computing $\mathcal{F}_5^*(sk_C, sk_{Op}, R) \oplus \lfloor Auts \rfloor_{48}$. Then, it verifies if its own incremented $Sqn_{Op,C}$ is in the correct range relatively to Sqn_C. If $Sqn_{Op,C}$ verifies this property,

the new list of authentication vectors are initiated with $\mathsf{Sqn}_{\mathsf{Op,C}}$ (the stored value). If $\mathsf{Sqn}_{\mathsf{C}}$ is out of range with respect to $\mathsf{Sqn}_{\mathsf{Op,C}}$ the operator verifies Mac^* . If this step is successful, the HLR/AuC resets the value of $\mathsf{Sqn}_{\mathsf{Op,C}}$ to $\mathsf{Sqn}_{\mathsf{Op,C}} := \mathsf{Sqn}_{\mathsf{C}}$ and sends a new list of authentication data vectors initiated with this updated $\mathsf{Sqn}_{\mathsf{Op,C}}$. This list may also contain only a single authentication vector. Figure 2.8 details this resynchronization procedure.

Reallocation of the Temporary Identifier.

At the end of the key derivation, both entities (ME/USIM and VLR) need to update the temporary identifier with a new value. To allocate new TMSI_n, the VLR generates this value in its own LAI [4], but only after a successful identification based on the old TMSI_o, and a successful key setting permitting to share the ciphering key CK (by the UMTS-AKA protocol). The VLR sends the new TMSI_n to the ME/USIM in a ciphered mode by using the A5/3 algorithm included in the UeA set of algorithms, using the derived key CK. These algorithms are described in the technical specification TS 43.020 [20]. The ME/USIM recovers with the key CK the new TMSI_n, stores it and de-allocates TMSI_o. Then ME/USIM sends an "acknowledge message" in cleartext to confirm its allocation. After receiving this message, the VLR de-allocates the TMSI_o and stores TMSI_n. If the VLR does not receive such a message, the network will associate the user's IMSI with both TMSI_o and TMSI_n. For the next identification, the mobile station can use either value (TMSI_o and TMSI_n). This will allow the network to determine the TMSI stored in the ME/USIM; the association between the other TMSI and the IMSI can then be deleted, to allow the unused TMSI to be allocated to another ME/USIM. In theory, the temporary identifier has to be used only once and reallocated after each secure channel establishment. We note that in practice it is not really the case.

2.3.2 AKA in the 4G Networks: The EPS-AKA Protocol

Although the UMTS-AKA protocol forms the backbone of EPS-AKA, i.e., the form of AKA used in 4G network, the latter comes with few differences which we detail below. Recall that one modification in 4G architectures is replacing the intermediary entity VLR/SGSN by a Mobility Management Entity (MME) instead VLR/SGSN. As for UMTS-AKA, the EPS version requires two long-term symmetric keys: the subscriber key and the operator key. The protocol is still stateful relying on the sequence number. A difference is that the temporary identifier use in the 4G networks is called Global User Temporary Identity (GUTI) instead the TMSI.

User Identification.

In the 4G architecture, the user identification proceeds as in the 3G architecture, except that the entities use a temporary identifier denoted GUTI instead of the TMSI and the LAI. To start the protocol, the client must to identify to the service provider, allowing it to associate the client with its permanent IMSI.

A new temporary identifier: GUTI. The Global User Temporary Identity (GUTI) is meant to provide an unambiguous identification of the user, without revealing the latter's permanent identifier mobile networks. It is constructed from server and user identifiers. This 64-bit identifier is composed by the following values:

- The Globally Unique MME Identifier (**GUMMEI**) which identifies the MME that allocated

[4] They are not recommended methods to generate the TMSI as the technical specification 23.003 [7] (*"the structure and coding of it can be chosen by agreement between operator and manufacturer in order to meet local needs"*). They only provide one advice (some parts of the TMSI may be related to the time)

the GUTI. It is composed by the MCC and MNC values described in Section 2.3.1 previously detailed and a MME identifier (**MMEI**). The latter has two components: a 16-bit MME Group identifier and an 8-bit MME Code.

- The value **M-TMSI** uniquely identifies the UE similarly to the 3G temporary identifier.

Since the GUTI has one component which identifies the MME, this value also allows the MME and mobile network identification, but without any authentication. See the technical specification 23.003 [7] for more details.

A Local GUTI Unknown Procedure. In the 4G architectures, the location area identifier (LAI) is useless (since the GUMMEI identifies the MME), and only the GUTI is required for a user identification. The Local GUTI Unknown Procedure (which replaces the Local TMSI Unknown Procedure) behaves as described in Figure 2.9.

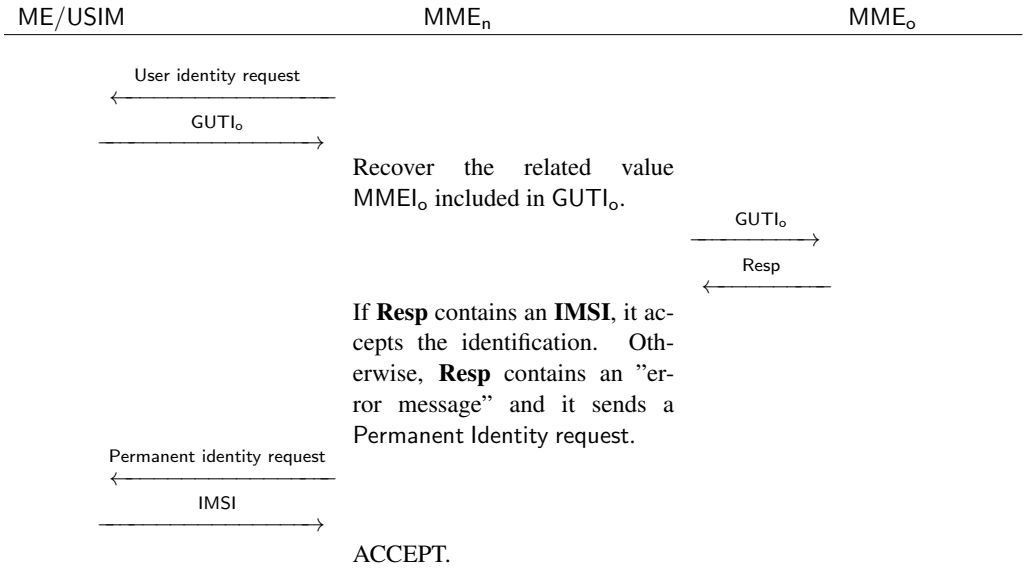
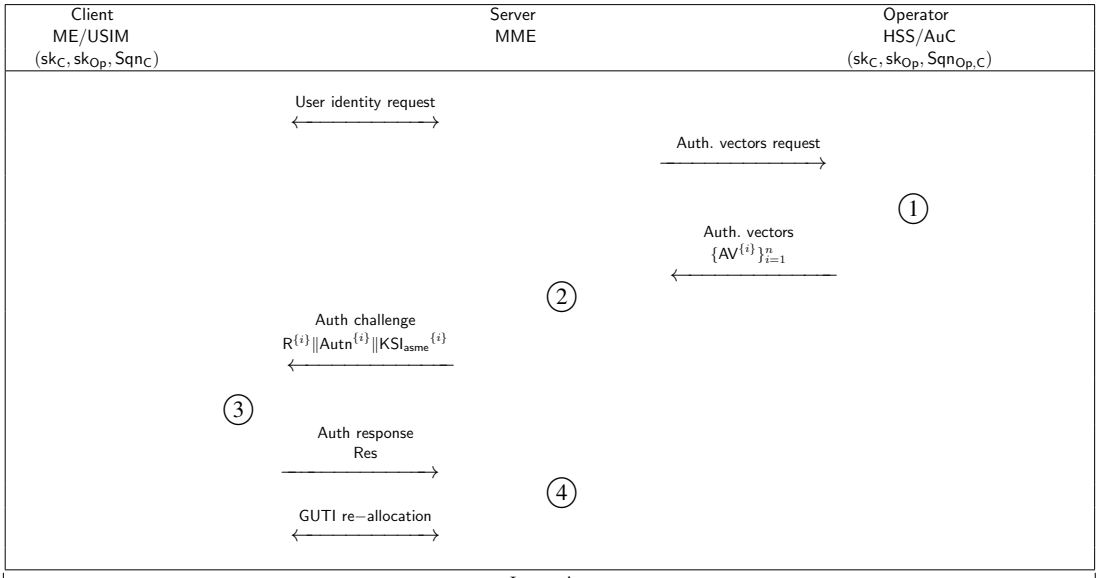


Figure 2.9: Local GUTI Unknown Procedure in 4G networks.

Challenge-Response.

The challenge-response of the EPS-AKA behaves almost as the UMTS-AKA protocol described in Section 2.3.1. Figure 2.10 describes the new procedure.
This procedure is similar, but differs in the management of the session keys. Indeed the two session keys CK and IK computed at the end of the protocol are no longer used to protect the radio link for interoperability reasons. They are only used to compute a local main session key called K_{asme}. This key is included in the authentication vectors sent by AuC to the MME, and while the two old session keys CK and IK are removed. After receiving this new batch of authentication vectors, the MME sends to the USIM via user equipment the random challenge R and an authentication vector Autn for network authentication from the selected authentication vector. It also includes an identifier KSI_{asme} for the user equipment which will be used to identify the K_{asme} (and further keys derived from the K_{asme}) that results from the EPS-AKA procedure. The KSI_{asme} is allocated



Instructions:

Client	Server	Operator
③: Compute AK using $R^{\{i\}}$. Recover $Sqn^{\{i\}}$ (from AK). Check Mac_S value. If $Sqn^{\{i\}} \in (Sqn_C, Sqn_C + \Delta)$: Compute: $CK \leftarrow \mathcal{F}_3(sk_C, sk_{Op}, R^{\{i\}})$, $IK \leftarrow \mathcal{F}_4(sk_C, sk_{Op}, R^{\{i\}})$, $K_{asme} \leftarrow KDF(CK\|IK, 0x10\|ID_{SN}\|\|ID_{SN}\|\|(Sqn \oplus AK)\|\|(Sqn \oplus AK)|)$. Set $Res := \mathcal{F}_2(sk_C, sk_{Op}, R^{\{i\}})$. Update $Sqn_C := Sqn^{\{i\}}$. Else *re-synchronization*	②: Store $\{AV^{\{i\}}\}_{i=1}^{n}$. Choose $AV^{\{i\}}$ one by one in order. Then it sends the related challenge. ④: $Res \stackrel{?}{=} Mac_C$.	①: For each $i = 1, \ldots, n$, compute: Generate $R^{\{i\}}$. Compute: $Sqn^{\{i\}} \leftarrow inc(Sqn_{Op,C})$ $Mac_S^{\{i\}} \leftarrow \mathcal{F}_1(sk_C, sk_{Op}, R^{\{i\}}, Sqn^{\{i\}}, AMF)$, $Mac_C^{\{i\}} \leftarrow \mathcal{F}_2(sk_C, sk_{Op}, R^{\{i\}})$, $CK^{\{i\}} \leftarrow \mathcal{F}_3(sk_C, sk_{Op}, R^{\{i\}})$, $IK^{\{i\}} \leftarrow \mathcal{F}_4(sk_C, sk_{Op}, R^{\{i\}})$, $AK^{\{i\}} \leftarrow \mathcal{F}_5(sk_C, sk_{Op}, R^{\{i\}})$, $Autn^{\{i\}} \leftarrow (Sqn^{\{i\}} \oplus AK^{\{i\}}), AMF, Mac_S^{\{i\}}$. $K_{asme} \leftarrow KDF(CK\|IK, 0x10\|ID_{SN}\|\|ID_{SN}\|\|(Sqn \oplus AK)\|\|(Sqn \oplus AK)|)$. $AV^{\{i\}} := R^{\{i\}}, CK^{\{i\}}, IK^{\{i\}}, Autn^{\{i\}}, Mac_C^{\{i\}}$, with $Sqn_{Op,C} = Sqn^{\{i\}}$. End For.

Figure 2.10: The EPS-AKA Procedure.

by the MME and uniquely identifies the K_{asme}. It is stored in the UE and serving MME together with the GUTI (if one is available), allowing the K_{asme} to be cached and re-used during subsequent connections without re-authentication. A new authentication procedure includes a different KSI_{asme}. At receipt of this message, the user verifies the freshness of the authentication vector by checking the received sequence number and the MAC value as described in technical specification TS 33.102 [13]. If these checks are successful, the user computes a response Res. The user then computes CK and IK, then use it to derive computing the local master key K_{asme}. Finally, the user equipment and the MME, after having accepted the session, have to compute session keys for NAS and RRC traffic.

New Session Keys Hierarchy.

The key hierarchy in EPS is considerably more elaborate than the UMTS one. Indeed, the new mobile network requires a network separation for the confidentiality and integrity protection in the access network. That implies a binding of session keys to the serving network identity, which is not the case with the keys CK and IK. Thus, instead of computing only two session keys CK and IK respectively for guaranteeing the confidentiality and integrity for all the communication in the access network, separated key-pairs are computed in 4G networks for each type

of communications. A first local master key $\mathsf{K_{asme}}$ is computed by the client and by the network which computes the authentication vectors. This key is distributed to the server MME. Re-using the local master key $\mathsf{K_{asme}}$ reduces the frequency with which authentication vectors need to be fetched from the operator. Finally, this key is less exposed since it is never transferred in the core network. Following the computation of $\mathsf{K_{asme}}$, a second master key called $\mathsf{K_{eNb}}$ is computed to the radio access network level. This key is computed by the server and distributed to the base station. To derive all the session keys used in the encryption and integrity algorithms, the client, MME, and eNodeB all use a key-derivation function KDF, typically instantiated as HMAC-SHA-256, as detailed in the technical specification TS 33.220 [6]. To derive the 256-bit key $\mathsf{K_{asme}}$, the user and the operator uses the KDF function taking in input the following mandatory parameters: CK, IK, and serving network identifier $\mathsf{ID_{SN}}$. The latter value consists of the encoded concatenation of the values MCC and MNC described. The key $\mathsf{K_{asme}}$ is computed by the following expression: $\mathsf{K_{asme}} = \mathsf{KDF}(\mathsf{CK}\|\mathsf{IK}, 0\text{x}10\|\mathsf{ID_{SN}}|||\mathsf{ID_{SN}}|||(\mathsf{Sqn} \oplus \mathsf{AK})|||(\mathsf{Sqn} \oplus \mathsf{AK})|)$, including thus the exclusive or of the sequence number (Sqn) and the anonymity key (AK) the value used during the related session of the EPS-AKA protocol. After having accepted the session, the user and the MME compute several session keys: $\mathsf{K_{eNB}}$, $\mathsf{K_{NAS_{int}}}$, $\mathsf{K_{NAS_{enc}}}$, $\mathsf{K_{UP_{enc}}}$, $\mathsf{K_{RRC_{int}}}$, $\mathsf{K_{RRC_{enc}}}$ and $\mathsf{K_{UP_{int}}}$, each protecting a different level of communications (traffic) between the server and the client. Since all these keys are derived (directly or not) from the key $\mathsf{K_{asme}}$, we can assume both parties use these key appropriately. Moreover, note that these keys are computed directly in the user equipment and not necessary in the USIM (as the session keys CK and IK). All the keys (except $\mathsf{K_{eNB}}$) are computed as following the following computation (more details TS 33.401 [8]): $\mathsf{K_{derived}} = \mathsf{KDF}(\mathsf{K}, 0\text{x}11\|\mathsf{P_0}|||\mathsf{P_0}|||\mathsf{P_1}|||\mathsf{P_1}|)$, with $\mathsf{P_0}$ the value of algorithm type distinguisher and $\mathsf{P_1}$ the EPS Integrity Algorithm (EIA). The key hierarchy is described as follows:

- Intermediate key $\mathsf{K_{eNB}}$: This eNodeB key is a key derived by UE and the MME from $\mathsf{K_{asme}}$ or by the ME and a target eNB as follows:
  $\mathsf{K_{eNB}} = \mathsf{KDF}(\mathsf{K_{asme}}, 0\text{x}11\|\mathsf{P_0}\|\mathsf{L_0})$, with $\mathsf{P_0}, \mathsf{L_0}$ two values depending the uplink NAS COUNT.

- Keys for NAS traffic: $\mathsf{K_{NAS_{int}}}$ (respectively $\mathsf{K_{NAS_{enc}}}$) is a key used to protect the NAS traffic particularly for providing integrity (respectively confidentiality). It is derived by the ME and MME from $\mathsf{K_{asme}}$ and an identifier for the integrity algorithm $\mathsf{P_0} = 0\text{x}02$ (respectively $\mathsf{P_0} = 0\text{x}01$).

- Keys for UP traffic: $\mathsf{K_{UP_{int}}}$ (respectively $\mathsf{K_{UP_{enc}}}$) is a key used to protect the UP traffic particularly for providing integrity (respectively confidentiality). This key is derived by ME and eNB from $\mathsf{K_{eNB}}$ and an identifier for the encryption algorithm $\mathsf{P_0} = 0\text{x}04$ (respectively $\mathsf{P_0} = 0\text{x}03$).

- Keys for RRC traffic: $\mathsf{K_{RRC_{int}}}$ (respectively $\mathsf{K_{RRC_{enc}}}$) is a key used to protect the RRC traffic particularly for providing integrity (respectively confidentiality). This key is derived by ME and eNB from $\mathsf{K_{eNB}}$, as well as an identifier for the encryption algorithm $\mathsf{P_0} = 0\text{x}06$ (respectively $\mathsf{P_0} = 0\text{x}05$).

Figure 2.11 details this session-keys hierarchy.

Reallocation of the Temporary Identifier.

Upon completing a successful run of the EPS-AKA protocol, a new GUTI must be reallocated to replace the one used in the current execution. The MME generates a new GUTI for that client, and sends it in ciphered mode. As detailed in Section 5.4.1 of the technical specification

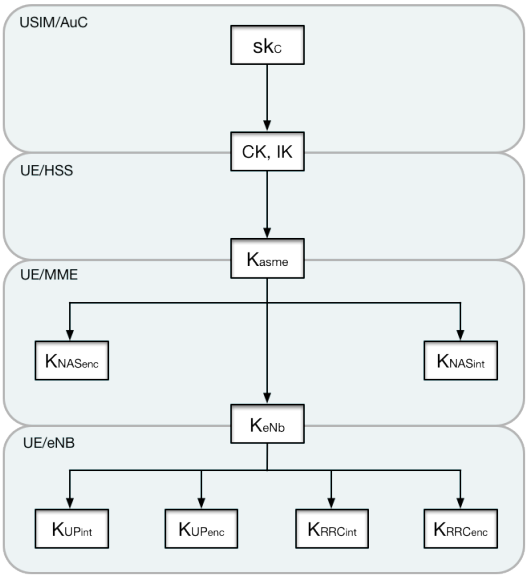


Figure 2.11: New session keys hierarchy in 4G networks.

TS 24.301 [15], the behaviour of the two entities is similar to that during a TMSI re-allocation procedure. Indeed, when the user receives a ciphered identifier, it recovers the new GUTI which is considered as valid and the old one becomes as invalid. After receiving the acknowledgement message, the MME considers the new GUTI as valid and the old one as invalid.

2.3.3 The Internal Cryptographic Primitives

Both AKA protocols are based on a set of seven functions which generate the cryptographic output necessary to attain the required security. In the original seven-algorithm proposal called MILENAGE [2], these functions relied on AES encryption. As an alternative to MILENAGE, another set of algorithms called TUAK [4] was proposed, the latter relying on a truncation of Keccak's internal permutation. The winner of the SHA-3 hash function competition, Keccak relies on the sponge construction [45], thus offering both higher performance, in hardware and software, than AES, and resistance to many generic attacks. While the TUAK algorithms, designed by the ETSI SAGE group, inherit Keccak's superior performance, they do not use the Keccak permutation in a usual, black box way. Instead the internal permutation is truncated, then used in a cascade, which makes it non-trivial to analyze. We cannot simply use the same assumptions for the truncated version as we would for the original permutation, either. Our analysis of the key security, as well as client- and respectively server-impersonation resistance of the protocol concerns both the MILENAGE based instantiations.

The MILENAGE Algorithms.

MILENAGE [2] is a set of algorithms which aims to achieve authentication and key generation properties. As opposed to TUAK which is based on Keccak's internal permutation, the MILENAGE algorithms are based on the Advanced Encryption Standard (AES). The functions $\mathcal{F}_1^*$, $\mathcal{F}_1$ and $\mathcal{F}_2$ must provide authentication, while $\mathcal{F}_3$, $\mathcal{F}_4$, $\mathcal{F}_5$ and $\mathcal{F}_5^*$ are used to derive key material in order to achieve confidentiality, integrity, and anonymity. The various parameters of these functions are:

- Inputs: $\mathsf{sk}_{\mathsf{Op}}$, sk_{C} two 128-bit long term credential keys that are fixed by the operator, a 128-bit random value R , a 48-bit sequence number Sqn, and a 16-bit authentication field management AMF chosen by the operator (the last two values are only used for the MAC

generation). We denote that the subscriber key $\mathsf{sk_{Op}}$ is a private key shared by all the subscribers of the same operator. The 128-bit subscriber key $\mathsf{sk_C}$ is shared out of band between the client and operator.

- Five 128-bit constants c_1,c_2,c_3,c_4,c_5 which are xored onto intermediate variables and are defined as follows:
  - $\mathsf{c}_1[i] = 0, \forall i \in \{0, 127\}$.
  - $\mathsf{c}_2[i] = 0, \forall i \in \{0, 127\}$, except that $\mathsf{c}_2[127] = 1$.
  - $\mathsf{c}_3[i] = 0, \forall i \in \{0, 127\}$, except that $\mathsf{c}_3[126] = 1$.
  - $\mathsf{c}_4[i] = 0, \forall i \in \{0, 127\}$, except that $\mathsf{c}_4[125] = 1$.
  - $\mathsf{c}_5[i] = 0, \forall i \in \{0, 127\}$, except that $\mathsf{c}_5[124] = 1$.
- Five integers r_1,r_2,r_3,r_4,r_5 in the range $\{0, 127\}$ which define amounts by which intermediate variables are cyclically rotated. They are defined as follows: $\mathsf{r}_1 = 64; \mathsf{r}_2 = 0; \mathsf{r}_3 = 32; \mathsf{r}_4 = 64; \mathsf{r}_5 = 96$.

The generation of all AKE output starts by initializing a value Top_C. To do so, one applies a first call of the AES function on inputs the operator and subscriber keys: $\mathsf{Top}_C = \mathsf{sk_{Op}} \oplus \mathsf{AES_{sk_C}}(\mathsf{sk_{Op}})$. We recall that $\mathsf{AES_K}(\mathsf{M})$ denotes the result of applying the Advanced Encryption Standard encryption algorithm to the 128-bit value M under the 128-bit key K. Then, we compute the following values taking as input Sqn, R, AMF and other constants:

- $\mathsf{Temp} = \mathsf{AES_{sk_C}}(\mathsf{R} \oplus \mathsf{Top}_C)$,
- $\mathsf{Out}_1 = \mathsf{AES_{sk_C}}(\mathsf{Temp} \oplus \mathsf{Rot_{r_1}}(\mathsf{Sqn} \| \mathsf{AMF} \| \mathsf{Sqn} \| \mathsf{AMF}) \oplus \mathsf{c}_1) \oplus \mathsf{Top}_C$,
- $\mathsf{Out}_2 = \mathsf{AES_{sk_C}}(\mathsf{Rot_{r_2}}(\mathsf{Temp} \oplus \mathsf{Top}_C) \oplus \mathsf{c}_2) \oplus \mathsf{Top}_C$,
- $\mathsf{Out}_3 = \mathsf{AES_{sk_C}}(\mathsf{Rot_{r_3}}(\mathsf{Temp} \oplus \mathsf{Top}_C) \oplus \mathsf{c}_3) \oplus \mathsf{Top}_C$,
- $\mathsf{Out}_4 = \mathsf{AES_{sk_C}}(\mathsf{Rot_{r_4}}(\mathsf{Temp} \oplus \mathsf{Top}_C) \oplus \mathsf{c}_4) \oplus \mathsf{Top}_C$,
- $\mathsf{Out}_5 = \mathsf{AES_{sk_C}}(\mathsf{Rot_{r_5}}(\mathsf{Temp} \oplus \mathsf{Top}_C, \mathsf{r}_5) \oplus \mathsf{c}_5) \oplus \mathsf{Top}_C$.

All the outputs of the MILENAGE algorithms are computed as follows:

- **Output** $\mathcal{F}_1$**:** $\mathsf{Mac_C} = \lfloor \mathsf{Out}_1 \rfloor_{0..63}$,
- **Output** $\mathcal{F}_1^*$**:** $\mathsf{Mac}^* = \lfloor \mathsf{Out}_1 \rfloor_{64..127}$,
- **Output** $\mathcal{F}_2$**:** $\mathsf{Mac_S} = \lfloor \mathsf{Out}_2 \rfloor_{64..127}$,
- **Output** $\mathcal{F}_3$**:** $\mathsf{CK} = \mathsf{Out}_3$,
- **Output** $\mathcal{F}_4$**:** $\mathsf{IK} = \mathsf{Out}_4$,
- **Output** $\mathcal{F}_5$**:** $\mathsf{AK} = \lfloor \mathsf{Out}_2 \rfloor_{0..47}$,
- **Output** $\mathcal{F}_5^*$**:** $\mathsf{AK}^* = \lfloor \mathsf{Out}_5 \rfloor_{0..47}$,

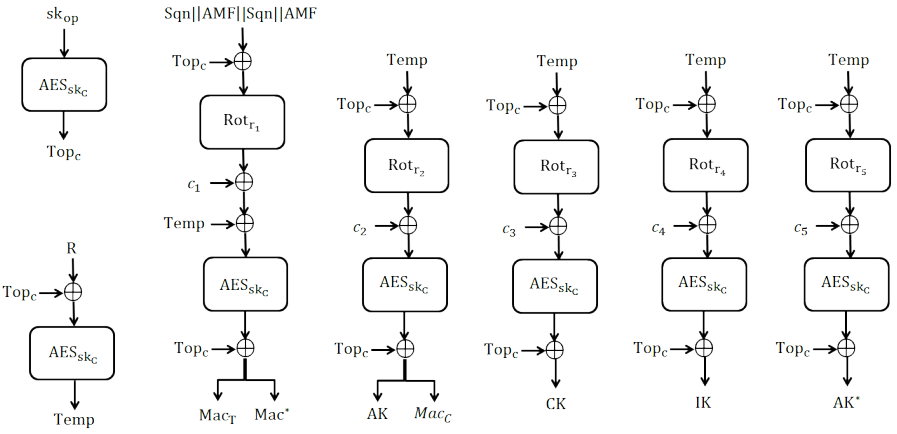


Figure 2.12: MILENAGE diagram.

This is also described in Figure 2.12.

The TUAK Algorithms.

TUAK [4] is a set of algorithms based on a truncation of the internal permutation function of Keccak; however, for efficiency reasons, only one or two iterations of the internal TUAK permutation are used. The goal of the TUAK functions is to provide secure authentication and key-exchange in the AKA protocols. In particular the TUAK functions $\mathcal{F}_1$ (respectively $\mathcal{F}_1^*$) and $\mathcal{F}_2$ must provide authentication, while $\mathcal{F}_3$, $\mathcal{F}_4$, and $\mathcal{F}_5$ (respectively $\mathcal{F}_5^*$) are used to derive the session keys used to attain confidentiality, integrity, and anonymity. The seven functions are parametrized by:

- Inputs: $\mathsf{sk_{Op}}$ a 256-bit long term operator key, a 128-bit random value R, a 48-bit sequence number Sqn, and a 16-bit authentication field management string AMF chosen by the operator (the last two values are only used for the MAC generation). Note that all subscribers to the same operator will share that operator's key $\mathsf{sk_{Op}}$. Additionally, a subscriber key $\mathsf{sk_C}$ is shared out of band between the HSS/AuC and ME/USIM which allows to initialize the value Key:
  - If $|\mathsf{sk_C}| = 128$ bits, then $\mathsf{Key} \leftarrow \mathsf{sk_C}[127..0]||0^{128}$.
  - If $|\mathsf{sk_C}| = 256$ bits, then $\mathsf{Key} \leftarrow \mathsf{sk_C}[255..0]$.
- Several public constants:
  - AN: a fixed 56-bit value $0x5455414B312E30$.
  - Inst and Inst' are fixed binary variables of 8 bits, specified in [4], which depend on the functions and the output sizes.

The generation of all AKE output starts once more by initializing Top_C. To do so, one applies a first $\mathsf{f_{Keccak}}$ permutation on a 1600-bit state Val_1 as $\mathsf{Val}_1 = \mathsf{sk_{Op}}||\mathsf{Inst}||\mathsf{AN}||0^{192}||\mathsf{Key}||\mathsf{Pad}||1||0^{512}$, where Pad is a bitstring output by a padding function. The value Top_C corresponds to the first 256 bits of this output. At this point, the behaviour of the functions $\mathcal{F}_1$ and $\mathcal{F}_1^*$ diverges from that of the other functions. To generate the MAC value of $\mathcal{F}_1$ and $\mathcal{F}_1^*$, we take as input Sqn, AMF, and R, three values chosen by the operator, and some constants. After the generation of Top_C, we initialize a second state $\mathsf{Val}_2 = \mathsf{Top}_C||\mathsf{Inst'}||\mathsf{AN}||\mathsf{R}||\mathsf{AMF}||\mathsf{Sqn}||\mathsf{Key}||\mathsf{Pad}||1||0^{512}$. Then, one applies the TUAK permutation on Val_2, using only the first 64 bits to compute $\mathsf{Mac_S}$. To generate the session keys and run $\mathcal{F}_2$, one initializes a second state for this function, namely,

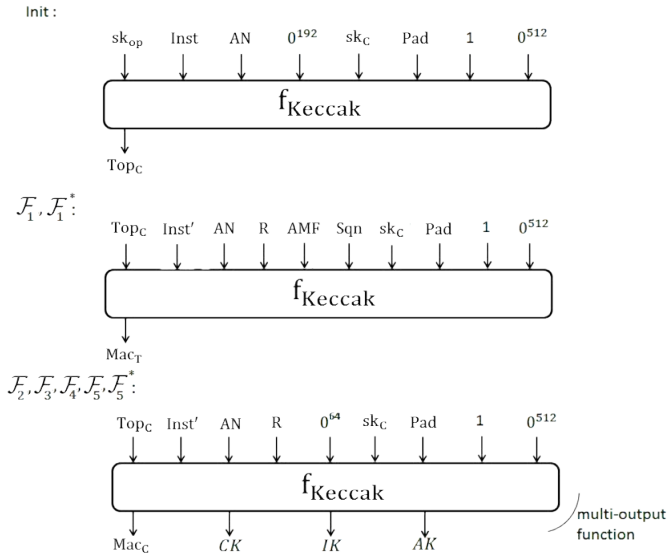


Figure 2.13: TUAK diagram.

$\mathsf{Val}_2 = \mathsf{Top}_C \| \mathsf{Inst}' \| \mathsf{AN} \| \mathsf{R} \| 0^{63} \| \mathsf{Key} \| \mathsf{Pad} \| 1 \| 0^{512}$. Then, the TUAK permutation is applied on Val_2 yielding Out, which in turn is used to compute the response $\mathsf{Mac_C}$ and the session keys:

$$\begin{aligned} \mathsf{Mac_C} &= \lfloor \mathsf{Out} \rfloor_{|\ell|-1..0}, \ell \in \{16, 32, 64, 128\}, \\ \mathsf{CK} &= \lfloor \mathsf{Out} \rfloor_{256..384} \text{ and } |\mathsf{CK}| = 128, \\ \mathsf{IK} &= \lfloor \mathsf{Out} \rfloor_{512..640} \text{ and } |\mathsf{IK}| = 128, \\ \mathsf{AK} &= \lfloor \mathsf{Out} \rfloor_{768..816} \text{ and } |\mathsf{AK}| = 48. \end{aligned}$$

This is also depicted in Figure 2.13. The way the output of the functions is truncated and used is the reason why TUAK is called a *multi-output function*. This is one of TUAK's chief differences from MILENAGE, and it has a non-negligible impact on its efficiency, as it saves a few calls to the internal function. However, this multi-output property can be an issue for the security of the master key, since during one session we can have as many as four calls to the same function with similar inputs (and a different truncation). Having different chunks of the same 1600-bit state (called Out in our description) can lead to recovering the long-term key $\mathsf{sk_C}$ by the reversibility of the TUAK permutation. The concatenation of all the different chunks used per session totals at most only 432 out of the 1600 output bits. Thus, though having multiple outputs can be hazardous in general, the Keccak-based construction of TUAK allows this without compromising the long-term parameters.

Chapter 3

Security analysis of the AKA protocols

Contents

The AKA protocols were designed for 3G and 4G networks, and are currently used to securely provide service to mobile clients on 3G and 4G networks. As a standardized, and highly used protocols, they are likely to become one of the main building blocks securing 5G communications. Despite its significance, the security of the AKA protocols is not well-understood to date. Several previous results indicate privacy flaws and propose quite radical modifications which are claimed to provide better privacy. We focus on the *actual security guarantees* of the unmodified AKA protocols, and conclude that it is vulnerable to server corruptions and to offline relays. We furthermore indicate a small modification, easily incorporated in the design of this protocol, which provides a much stronger security.

3.1 Security analysis

The security goals of the AKA protocols are: the secrecy of the established sessions keys against both passive and active MitM adversaries, as well as mutual authentication. In particular, these protocols cannot guarantee (perfect) strong secrecy, as it uses symmetric long-term keys, which, once compromised, can also endanger past session keys. We formalize these goals in terms of three properties: key-indistinguishability, client-impersonation resistance, and server-impersonation resistance. Moreover, the protocols implicitly requires (and addresses) security with respect to malicious servers. Indeed, since servers are sometimes untrusted (in the case of roaming), the AKA protocols must also protect clients with respect to these third parties. Considering this adversary, the additional security properties are required and are defined by: (1) the servers have no access to the long-term symmetric keys; (2) the (hence necessary) operator-server communication must be minimized in order to minimize costs. We formulate the following two *implicit* requirements by the *state confidentiality*, i.e the servers must no learn any client-related long-term state, and *soundness* which means clients must reject authentication-challenges not explicitly provided by the operator to the server.

3.2 Security Model

The security of the AKA protocols is defined by five security properties which are key-indistinguishability, client- and server-impersonation resistance, state-confidentiality and soundness. These five security notions cannot be trivially proved in the Bellare-Rogaway model variations, e.g. [40]. Though AKA may seem to be a typical – if stateful – symmetric key agreement protocol, its design is convoluted and includes several unusual features. The sequence numbers provide state to the protocol, and are tied to a resynchronization procedure. The server authentication step allows an unorthodox kind of relay attack, which permits a degree of server impersonation. Furthermore, clients registered with the same operator share that operator's key $\mathsf{sk}_{\mathsf{Op}}$, though not their individual client keys. An interesting fact regarding the operator key is that it is never stored in clear in the client's SIM card. Finally, clients and servers may become desynchronized, and the resynchronization procedure is required, introducing a further protocol step. Due to these features, we cannot use the classical Bellare-Rogaway [40] or Bellare-Pointcheval-Rogaway [39] model for our analysis, though we employ a modified version of it. Our model is robust with respect to multiple clients, multiple operators, and different types of corruptions, and do take into account re-synchronizations, sequence numbers and a more restricted Man-in-the-Middle server-impersonation attacker model. Note that, even if this implies an imperfect mutual authentication, the offline relay strategy we depict, has no impact on the secrecy (indistinguishability from random) of the sessions keys. We split the guarantee of mutual authentication, which implies client and server impersonation resistance, into two properties. This is because the AKA protocols offer different degrees of security with respect to impersonation attacks for clients and for servers.

Setup and participants. We consider a set $\mathcal{P}$ of honest participants, which are either servers S or mobile clients C of the type ME/USIM subscribing to operators Op. A participant is generically denoted as P. In all security games with respect to Man-in-the-Middle adversary, the operators Op are black-box algorithms within the server S. For the soundness and state-confidentiality properties with respect the malicious server adversary, the operators Op are oracles accessible from servers. We assume the existence of n_C clients, n_S servers and n_{Op} operators. If the operators are contained within the servers, we assume that all copies of the same operator are synchronized at all times. Each client C is associated with a *unique* identifier UID, two long term static secret keys: $\mathsf{sk}_{\mathsf{UID}}$ (subscriber key), and $\mathsf{sk}_{\mathsf{Op}}$ (operator key), which is common to all clients subscribing to a

specific operator. Clients also keep track of a long-term state $\mathsf{st}_{\mathsf{UID}}$[1]. In particular, we consider multiple operators, with the restriction that each user may only be registered to a single operator[2]. In our model, we also assume for simplicity that the key space of all operators is identical, noting that neither the key-indistinguishability, nor the mutual authentication properties are affected by the way operators choose their keys (the security of both the key exchange and the authentication properties rely just on the key *length*); the variation in the key space does, however, affect user privacy. Each operator is assumed to store $\mathsf{sk}_{\mathsf{UID}}$ as part of a database entry of the tuples $(\mathsf{UID}, \mathsf{sk}_{\mathsf{UID}}, \mathsf{sk}_{\mathsf{Op}}, \mathsf{st}_{\mathsf{Op},\mathsf{UID}})$, each tuple corresponding to a single user of this operator. The last entry $\mathsf{st}_{\mathsf{Op},\mathsf{UID}}$ of each tuple denotes the long term state of the operator associated with that user – which may in fact differ from the state of the user itself. For the AKA protocols, the state is in fact a sequence number, associated with each client. Finally, servers do not contain any secret information pertaining to the operator or the subscriber. In our model, each participant may run concurrent key-agreement executions of the protocol Π. We denote the j-th execution of the protocol by the party P as P_j. We tacitly associate each instance P_i with a session ID sid, a partner ID pid (consisting either of one or of multiple elements), and an accept/reject bit accept. As explained more in the proofs, the partner ID is set to either the server or to a user identifier UID, whereas the session ID includes three values: the user ID given by the client (thus tacitly also the key associated with that UID), the randomness generated by the server, and the sequence number used for the authentication. Finally, the accept/reject bit is initialized to 0 and turns to 1 at the successful termination of the key agreement protocol. We call this "terminating in an accepting state".

In the absence of an adversary, the protocol is always run between a client C and a server S. For the AKA protocols, the server begins the protocol by means of an ID request, and can thus be called its *initiator*, whereas the mobile client is the *respondent*. A successful termination of the protocol yields, for each party, a session key K (which for the UMTS-AKA protocol consists of two keys), the session identifier sid, and the partner identifier pid of the party identified as the interlocutor. In the AKA protocols the client is authenticated by means of a challenge-response type of query, where the response is computed as a pseudo-random function of the key and (a part of) the challenge. The server is equally authenticated by means of an authentication string, also a pseudo-random function of the key, the challenge, and the long-term state that the server associates with that client. In particular, the challenge strings sent by the server are authenticated. During the next sections, we detail and formalize the five security properties of the AKA protocols.

3.2.1 Key-Indistinguishability

The notion of *key-indistinguishability* refers to the session keys calculated as a result of the key-exchange protocol (rather than to the long-term keys held by each party), requiring that they be indistinguishable from random bitstrings of equal length. The MitM adversary $\mathcal{A}$ can access instances of honest parties by means of oracles acting as interfaces; furthermore, $\mathcal{A}$ can schedule message deliveries, send tampered messages, or interact arbitrarily with any party, by means of the oracles below. We note that in the key-indistinguishability model the adversary may also know the long-term state (in our case, the sequence number) of both users and the server. This will also be the case in the impersonation games. Since the state is updated in a probabilistic way, we give the adversary a means of always learning the updated state of a party without necessarily corrupting it (the latter may rule out certain interactions due to notions of freshness, see below). Corruption is allowed and implied the related party is considered as *adversarially controlled*. We

[1] The latter consists in practice of a sequence number $\mathsf{Sqn}_{\mathsf{UID}}$, which is updated at each successful authenticated key exchange.

[2] We note that this seems to extend naturally to a case in which a single client may be registered with multiple operators, as long as the key-generation process for each operator is such that the registration of a single client to two operators is equivalent to representing a two-operator-client as two independent clients.

use the same fundamental model, with similar oracles, also for the definitions of *client* and *server* impersonation. We consider a finite (and public) list of n_{Op} operators $\mathsf{Op}_1, \ldots \mathsf{Op}_{n_{\mathsf{Op}}}$, for which the keys $\mathsf{sk}_{\mathsf{Op}_1}, \ldots \mathsf{sk}_{\mathsf{Op}_{n_{\mathsf{Op}}}}$ are generated independently and uniformly at random $\mathcal{S}_{\mathsf{op}}$.

Oracles. The adversary interacts with the system by means of the following oracles, in addition to a function G, which we model as a PRF.

- CreateCl(Op)$\rightarrow$ ($\mathsf{UID}, \mathsf{st}_{\mathsf{UID}}$): This oracle creates a client with unique identifier UID. Then the client's secret key $\mathsf{sk}_{\mathsf{UID}}$ and the sequence number $\mathsf{st}_{\mathsf{UID}} := \mathsf{Sqn}_{\mathsf{UID}}$. The tuples ($\mathsf{UID}$, $\mathsf{sk}_{\mathsf{UID}}, \mathsf{sk}_{\mathsf{Op}}, \mathsf{Sqn}_{\mathsf{UID}}$) are associated with the client UID and with the corresponding operator Op (i.e. each "copy" of Op in each server does this). The operator sets $\mathsf{st}_{\mathsf{Op},\mathsf{UID}} := \mathsf{Sqn}_{\mathsf{UID}}$ and then keeps track of $\mathsf{st}_{\mathsf{Op},\mathsf{UID}}$. The adversary is given UID.

- NewInstance(P) $\rightarrow$ (P_j, m): this oracle instantiates a new instance P_j, of party P, which is either a client or a server. Furthermore, the oracle also outputs a message m, which is either the first message in an honest protocol session (if P is a server) or $\perp$ (if P is a client). The state st of this party is initiated to be the current state of P, i.e the current value of $\mathsf{st}_{\mathsf{UID}}$.

- Execute($\mathsf{P}, i, \mathsf{P}', j$) $\rightarrow \tau$: creates (fresh) instances P_i of a server P and P'_j of a client P', then runs the protocol between them. The adversary $\mathcal{A}$ receives the transcript of the protocol.

- Send(P, i, m) $\rightarrow m'$: simulates sending message m to instance P_i of P. The output is a response message m' (which is set to $\perp$ in case of an error or an abort).

- Reveal(P,i) $\rightarrow$ $\{\mathsf{K}, \perp\}$: if the party has not terminated in an accepting state, this oracle outputs $\perp$; else, it outputs the session keys computed by instance P_i.

- Corrupt(P) $\rightarrow \mathsf{sk}_{\mathsf{P}}$: if P is a client, this oracle returns the long-term client key sk_{P}, but not $\mathsf{sk}_{\mathsf{Op}}$ (in this we keep faithful to the implementation of the protocol, which protects the key even from the user himself). If P is a server, then this oracle returns the identifier S_i, giving the adversary access to a special oracle OpAccess. If P is corrupted, then this party (and all its instances, past, present, or future), are considered to be adversarially controlled.

- OpAccess(S, C) $\rightarrow m$: for a corrupted server S, this oracle gives the adversary a one-time-access to the server's local copy of all the operators, in particular returning the message that the operator Op would have output to the server on input a client C. If S is not corrupted, this oracle outputs $\perp$.

- StReveal($\mathsf{C}, i, \mathsf{bit}_{\mathsf{S}}$) $\rightarrow x$: for a client P, if $\mathsf{bit}_{\mathsf{S}} = 0$, then this oracle reveals the current state of C_i; else, if $\mathsf{bit}_{\mathsf{S}} = 1$, then the oracle returns the state the operator stores for C.

- $\mathsf{Test}^{\mathsf{K.Ind}}$(P,$i$) $\rightarrow \hat{\mathsf{K}}$: this oracle is initialized with a secret random bit b. It returns $\perp$ if the instance P_i is unfresh or if it has not terminated in an accepting state (with a session key K). If $b = 0$, then the oracle returns $\hat{\mathsf{K}} := \mathsf{K}$, else it returns $\hat{\mathsf{K}} := \mathsf{K}'$, which is a value drawn uniformly at random from the same space as K. We assume that the adversary makes a single $\mathsf{Test}^{\mathsf{K.Ind}}$ query (a standard hybrid argument can extend the notion to multiple queries). We may assume that the adversary makes only a single $\mathsf{Test}^{\mathsf{K.Ind}}()$ query since we can extend our model to the multi-query scenario by a standard hybrid argument.

We allow the adversaries to learn whether instances have terminated and whether they have accepted or rejected their partners. Indeed, the adversary can always use Send queries to verify the status of a session. We do not model the precise error messages received by the two parties on abort, as they seem to have no effect on the key-indistinguishability and impersonation properties

of the two parties respectively. We also assume that the adversary will learn the session and partner identifiers for any session in which the instance has terminated in an accepting state.

Correctness and Partners. Each instance of each party keeps track of a session ID string, denoted sid. For the AKA protocols, this value consists of a triple of values: a user ID UID (corresponding to a single client C), a session-specific random value, and the sequence number used for the authentication step. We define partners as party instances that share the same session ID. More formally:

Definition 1. [**Partners.**] *Two instances* P_i *and* P'_j *are* partnered *if the following statements hold:*

(i) One of the parties is a user and the other is the server.

(ii) The two instances terminate in an accepting state.

(iii) The instances share the same sid.

In this case, the partner ID of some party P *denotes its (intended) partner.*

We define the correctness of the protocol as follows.

Definition 2. [**Correctness.**] *An execution of the protocol* Π *between two instances is correct if the execution is untampered with and if the following conditions hold:*

(i) The two conversing instances share the same sid*, i.e. they are partnered.*

(ii) Both instances output the same session key(s) K.

(iii) The partner identifiers pid *of the instances are correct, i.e they correspond to the conversing entities.*

We consider two classes of adversaries, *weak* and *strong*, depending on whether the adversary may corrupt servers or not. We model three requirements with respect to MitM adversaries.

Key-indistinguishability. For the property of key-indistinguishability, i.e. the guarantee that the session keys of honest sessions are indistinguishable from random, we could consider two types of models. The simpler of these gives the adversary the ability of recovering the secret key of the operator, which considerably eases the simulation in our proof. However, we note that the operator keys are not easy to recovery by a client in real-world implementations, as they are never stored on the USIM card[3]. Thus, a more realistic model is the one we present above, in which only the client key is recovered upon corruption.

The key-indistinguishability game is played as follows. First the challenger generates the keys of all the n_{Op} operators and gives black-box access to the server S. The adversary is then allowed to query any of the oracles above. We implicitly assume that the $\mathsf{Test}^{\mathsf{K.Ind}}$ oracle keeps state and, once it is queried a first time, it will return $\perp$ on all subsequent queries (we only allow a single query). However, we do allow the adversary to interact with other oracles after the $\mathsf{Test}^{\mathsf{K.Ind}}$ query as well.

Eventually, the adversary $\mathcal{A}$ outputs a bit d, which is a guess for the bit b used internally in the $\mathsf{Test}^{\mathsf{K.Ind}}$ oracle. The adversary *wins* if and only if: $b = d$ and $\mathcal{A}$ has queried a fresh instance to the $\mathsf{Test}^{\mathsf{K.Ind}}$ oracle. We consider the following definition of a fresh instance for the key-indistinguishability. We note that this notion is classical in authenticated- key-exchange protocols.

[3]Instead, what *is* stored in the USIM card is an intermediate value, obtained after either a first Keccak truncated permutation or a call of AES algorithm; thus the operator key is easy to use, but hard to recover.

Definition 3. [**Key-indistinguishability.**] *An instance* P_i *is* fresh *if the following queries had been not previously executed on neither this instance, nor a partner of* P_i*,*

(i) ***Reveal(.)****, either on the instance* P_i*, or on of its partners.*

(ii) ***Corrupt(.)*** *on any instance, either of* P*, or of their partners.*

The advantage of $\mathcal{A}$ in winning the key-indistinguishability game is defined as: $\mathsf{Adv}_{\Pi}^{\mathsf{K.Ind}}(\mathcal{A}) := |\Pr[\mathcal{A}\ \mathsf{wins}] - 1/2|$. We quantify the adversary's maximal advantage as a function of his resources which are the running time t, the number q_{exec} of instantiated party instances, and the maximum number of allowed resynchronization attempts q_{res} per instantiated instance.

Definition 4. [**Weak/Strong Key-Indistinguishability.**] *A key-agreement protocol* Π *is* $(t, q_{\mathsf{exec}}, q_{\mathsf{res}}, q_{\mathsf{G}}, \epsilon)$*-weakly key-indistinguishable (resp.* $(t, q_{\mathsf{exec}}, q_{\mathsf{res}}, q_{\mathsf{s}}, q_{\mathsf{Op}}, q_{\mathsf{G}}, \epsilon)$*-strongly key-indistinguishable) if no adversary running in time* t*, creating at most* q_{exec} *party instances with at most* q_{res} *resynchronizations per instance, (corrupting at most* q_{s} *servers and making at most* q_{Op} $\mathsf{OpAccess}$ *queries per operator per corrupted server for strong security), and making at most* q_{G} *queries to function* G*, has an advantage* $\mathsf{Adv}_{\Pi}^{\mathsf{K.Ind}}(\mathcal{A}) > \epsilon$*.*

3.2.2 Client Impersonation Resistance

Though the AKA protocols claim to provide mutual authentication, its design introduces a vulnerability, leading to a subtle difference between the degree of *client*-impersonation resistance and *server*-impersonation resistance. In fact, as detailed in the paragraph below, the protocols allow the adversary to do a type of Man-in-the-Middle attack which resembles, but is not quite the same as, a relay attack. We have two choices in modelling the client and server impersonation guarantees. The classical Bellare-Rogaway model, using the notion of freshness, cannot differentiate well between client- and server-impersonation resistance. A consequence is that we would only be able to prove a weaker client-impersonation guarantee than the one provided by the protocol. The alternative is to give a more accurate model, which features time and can capture the difference between online and offline relays. This is the strategy we use here. In a style akin to the distance-bounding model of Dürholz et al. [96], we introduce a time variable with positive integer values, denoted clock, which increments by 1 both when a Send query is sent by the adversary, and when an honest party responds to this query. Running the Execute query increments clock by 1 for each other implicit Send and for each implicit response step. For client impersonation, the only attacks we rule out are online relay attacks, which are (somewhat simplistically) depicted in Figure 3.1. We thus need to propose a more subtle definition of a fresh instance as follows:

Definition 5. [**Freshness:** C.Imp **resistance.**] *A server-instance* S_i*, with session ID* sid *and partner ID* pid*, is* fresh *if: neither this instance nor a partner of* S_i *is adversarially-controlled; and there exists no instance* C_j *sharing session* sid *with the partner* $\mathsf{pid} = \mathsf{S}_i$ *(the related transcript is denoted as* $(\mathsf{m}, \mathsf{m}', \mathsf{m}'')$*) such that the following events occur:*

(i) The message m *is sent by the adversary* $\mathcal{A}$ *to* S_i *via a* $\mathsf{Send}(m)$ *query at time* $\mathsf{clock} = k$*, yielding message* m' *at time* $\mathsf{clock} = k + 1$*.*

(ii) The message m′ *is sent by* $\mathcal{A}$ *to* C_j *via a* $\mathsf{Send}(m')$ *query at time* $\mathsf{clock} = k' > k + 1$*, yielding message* m'' *at time* $\mathsf{clock} = k' + 1$*.*

(iii) The message m″ *is sent by* $\mathcal{A}$ *to* S_i *via a* $\mathsf{Send}(m'')$ *query at time* $\mathsf{clock} = k'' > k' + 1$*.*

We note that the messages need not be exactly sequential (i.e. the adversary could query other oracles in different sessions before returning to session sid). Furthermore, the notion of freshness

only refers to relays with respect to the partner client pid. We do not restrict the adversary from forwarding received messages to other server or client instances. The goal of a client-impersonation adversary is to make a fresh server instance terminate in an accepting state. In this game, the Test oracle is not used.

More formally, the game begins by generating the operator keys as before; then the adversary $\mathcal{A}$ gains access to all the oracles except $\mathsf{Test}^{\mathsf{K.Ind}}$. When $\mathcal{A}$ stops, he *wins* if there exists an instance S_i that ends in an accepting state and is fresh as described above. The advantage of the adversary is defined as his success probability, i.e. $\mathsf{Adv}^{\mathsf{C.Imp}}_{\Pi}(\mathcal{A}) := \Pr[\mathcal{A}\ \mathsf{wins}]$.

Definition 6. [**Weak/Strong Client-Impersonation security.**] *A key-agreement protocol* Π *is* $(t, q_{\mathsf{exec}}, q_{\mathsf{res}}, q_{\mathsf{G}}, \epsilon)$*-weak-client-impersonation-secure (resp.* $(t, q_{\mathsf{exec}}, q_{\mathsf{res}}, q_{\mathsf{s}}, q_{\mathsf{Op}}, q_{\mathsf{G}}, \epsilon)$*-strong-client-impersonation secure) if no adversary running in time t, creating at most* q_{exec} *party instances with at most* q_{res} *resynchronizations per instance, (corrupting at most* q_{s} *servers and making at most* q_{Op} OpAccess *queries per operator per corrupted server for strong security), and making at most* q_{G} *queries to the function G, has an advantage* $\mathsf{Adv}^{\mathsf{C.Imp}}_{\Pi}(\mathcal{A}) \geq \epsilon$.

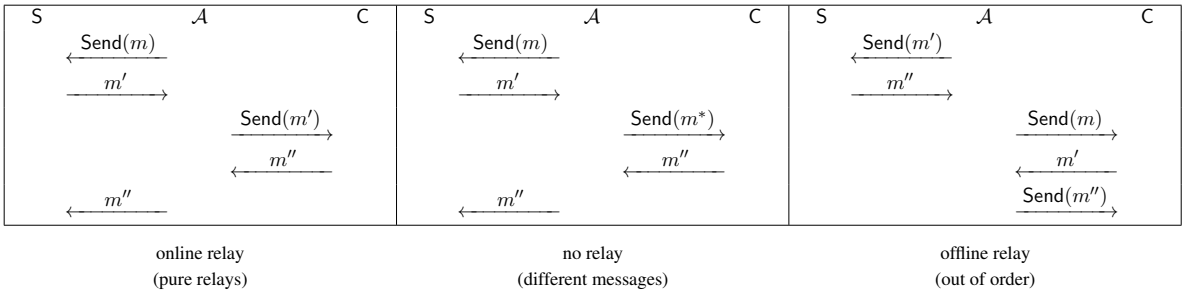


Figure 3.1: Examples of Online and Offline relays.

3.2.3 Server Impersonation Resistance

As we explain later in more detail in Section 3.5, it is possible to impersonate a server even if we rule out online relays. In particular, an adversary performs an offline (out of order) relay, as described in the third scenario of Figure 3.1. This is because the client's first message is the user id, which is always sent in clear (thus known to adversaries). This enables $\mathcal{A}$ to obtain, in a first session with the server, the server's authenticated challenge for a particular client UID, which it can replay to UID, in a separate (later) session. In essence, the adversary is relaying the messages, but this happens in two different, non-concurrent executions. This indicates a gap between the client impersonation and the server impersonation guarantees for the AKA protocols. Our server-impersonation model rules out both offline and online relays, redefining freshness as follows.

Definition 7. [**Freshness:** S.Imp **resistance.**] *An instance* C_i*, with session ID* sid *and partner ID* pid*, is* fresh *if: neither this instance nor a partner of* C_i *is adversarially-controlled; and there exists no instance* S_j *with session* sid *and partner* pid $=$ C_i *(the transcript of* sid *is denoted as* $(\mathsf{m}, \mathsf{m}', \mathsf{m}'')$*) such that the following events occur:*

(i) The message m *is sent by* $\mathcal{A}$ *to* S_j *via a* $\mathsf{Send}(m)$ *query yielding message* m'.

(ii) The message m′ *is sent by* $\mathcal{A}$ *to* C_i *via a* $\mathsf{Send}(m')$ *query yielding message* m''.

(iii) The message m″ *is sent by* $\mathcal{A}$ *to* S_j *via a* $\mathsf{Send}(m'')$ *query.*

The game is played as in the client impersonation case. When the adversary $\mathcal{A}$ stops, he *wins* if there exists a fresh instance C_i that ends in an accepting state. The advantage of the adversary is defined as its success probability, i.e. $\mathsf{Adv}_{\Pi}^{\mathsf{S.Imp}}(\mathcal{A}) := \Pr[\mathcal{A}\ \mathsf{wins}]$.

Definition 8. [**Weak/Strong Server-Impersonation security.**] *A key-agreement protocol* Π *is* $(t, q_{\mathsf{exec}}, q_{\mathsf{res}}, q_{\mathsf{G}}, \epsilon)$*-weak-server-impersonation-secure (resp.* $(t, q_{\mathsf{exec}}, q_{\mathsf{res}}, q_{\mathsf{s}}, q_{\mathsf{Op}}, q_{\mathsf{G}}, \epsilon)$*-strong-server-impersonation secure) if no adversary running in time* t*, creating at most* q_{exec} *party instances with at most* q_{res} *resynchronizations per instance, (corrupting at most* q_{s} *servers and making at most* q_{Op} OpAccess *queries per operator per corrupted server for strong security), and making at most* q_{G} *queries to the function* G*, has an advantage* $\mathsf{Adv}_{\Pi}^{\mathsf{S.Imp}}(\mathcal{A}) \geq \epsilon$.

3.2.4 Security with respect to Servers

In this section, we consider a new adversary which is a malicious, but legitimate server S. In the context of mobile networks, a (malicious) server cannot learn any secret data of the subscriber or operator, i.e the subscriber key sk_{C}, the operator key $\mathsf{sk}_{\mathsf{Op}}$, and the two related internal states. Moreover, the server must not be able to make a client accept the server's authentication (thus completing the key-derivation process), unless they are explicitly given authenticating information by a legitimate operator. We formalize these goals in terms of two new properties: *state-confidentiality* and *soundness*. This model is really similar as the previous one, and is based on the same participants which includes the adversary. For both properties, the adversary uses the UReg, NewInstance, Execute, Send, Reveal, StReveal oracles as described in the previous model. We additionally add two new oracles (including a new Corrupt oracle) as noted below:

- $\mathsf{Corrupt}(\mathsf{P}) \rightarrow S$: if P is a client, behave as before in the previous model. If P is an operator, returns $\mathsf{sk}_{\mathsf{Op}}$ and the list of tuples $S = (\mathsf{UID}, \mathsf{sk}_{\mathsf{UID}}, \mathsf{st}_{\mathsf{UID}}, \mathsf{st}_{\mathsf{Op,C}})$ for all clients C subscribing with that operator.

- $\mathsf{OpAccess}(\mathsf{C}) \rightarrow m$: this oracle gives the adversary one access to the server's own local copy of all the operators, in particular returning the message m that the client's operator Op would have output to the server on input a client C.

Unlike key-indistinguishability, which guarantees that *session* keys are indistinguishable from random with respect to MitM adversaries, the property of state-confidentiality demands that *long-term* client keys remain confidential with respect to *malicious servers*. This game begins by generating the material for n_{Op} operators and n_{C} clients. The adversary can then interact arbitrarily with these entities by using the oracles above. At the end of the game, the adversary must output a tuple: $(\mathsf{P}_i, \mathsf{sk}^*_{\mathsf{UID}}, \mathsf{sk}^*_{\mathsf{Op}}, \mathsf{st}^*_{\mathsf{UID}}, \mathsf{st}^*_{\mathsf{Op,UID}})$ such that UID is the long-term identifier of P and P_i is a fresh instance of P in the sense formalized below. The adversary wins if at least one of the values: $\mathsf{sk}^*_{\mathsf{UID}}, \mathsf{sk}^*_{\mathsf{Op}}, \mathsf{st}^*_{\mathsf{UID}}, \mathsf{st}^*_{\mathsf{Op,UID}}$ is respectively equal to $\mathsf{sk}_{\mathsf{UID}}, \mathsf{sk}_{\mathsf{Op}}, \mathsf{st}_{\mathsf{UID}}, \mathsf{st}_{\mathsf{Op,UID}}$, the real secret values of the fresh instance P_i.

Definition 9. [**Freshness:** St.Conf.] *An instance* P_i *is* fresh *if neither this instance, nor a partner of* P_i *is adversarially-controlled (its long-term key* sk_{P} *has not been corrupted) and the following queries were not previously executed:*

(i) StReveal*(.) on* any *instance of* P.

(ii) Corrupt*(.) on* any *instance of* P *or on the operator* Op *to which* P *subscribes.*

The advantage of the adversary is defined as: $\mathsf{Adv}_{\Pi}^{\mathsf{St.Conf}}(\mathcal{A}) := \Pr[\mathcal{A}\ \mathsf{wins}]$.

Definition 10. [**State-confidentiality.**] *A key-agreement protocol* Π *is* $(t, q_{\mathsf{exec}}, q_{\mathsf{res}}, q_{\mathsf{Op}}, q_{\mathsf{G}}, \epsilon)$*-state-confidential if no adversary running in time* t*, creating at most* q_{exec} *party instances with at most* q_{res} *resynchronizations per instance, making at most* q_{Op} $\mathsf{OpAccess}$ *queries and* q_{G} *queries to* G*, has an advantage* $\mathsf{Adv}_{\Pi}^{\mathsf{St.Conf}}(\mathcal{A}) \geq \epsilon$.

In the *Soundness* game, we demand that no server is able to make a fresh client instance terminate in an accepting state without help from the operator. This game resembles impersonation-security; however, this time the adversary is a legitimate server (not a MitM) and it has access to operators. The adversary may interact with oracles in the soundness game arbitrarily, but we only allow a maximum number of q_{Op} queries to the $\mathsf{OpAccess}$ oracle per client. The adversary wins if there exist $(q_{\mathsf{Op}} + 1)$ fresh client instances of a given client which terminated in an accepting state. Freshness is defined similarly as in the impersonation game with the same restriction due to the offline replays attacks:

Definition 11. [**Freshness:** Sound**.**] *An instance* C_i*, with session ID* sid *and partner ID* pid*, is* fresh *if: neither this instance, a partner of* C_i *nor their related operator* Op *is adversarially-controlled ;and there exists no instance* S_j *with session* sid *and partner* $\mathsf{pid} = \mathsf{C}_i$ *(the transcript of* sid *is denoted as* $(\mathsf{m}, \mathsf{m}', \mathsf{m}'')$*) such that the following events occur:*

(i) The message m *is sent by* $\mathcal{A}$ *to* S_j *via a* $\mathsf{Send}(m)$ *query yielding message* m'.

(ii) The message m' *is sent by* $\mathcal{A}$ *to* C_i *via a* $\mathsf{Send}(m')$ *query yielding message* m''.

(iii) The message m'' *is sent by* $\mathcal{A}$ *to* S_j *via a* $\mathsf{Send}(m'')$ *query.*

The advantage of the adversary is defined as: $\mathsf{Adv}_{\Pi}^{\mathsf{Sound}}(\mathcal{A}) := \Pr[\mathcal{A}\ \mathsf{wins}]$.

Definition 12. [Soundness.] *A key-agreement protocol* Π *is* $(t, q_{\mathsf{exec}}, q_{\mathsf{res}}, q_{\mathsf{Op}}, q_{\mathsf{G}}, \epsilon)$*-server-sound if no adversary running in time* t*, creating at most* q_{exec} *party instances with at most* q_{res} *resynchronizations per instance, making at most* q_{Op} *queries to any operator* Op *and at most* q_{G} *queries to the function* G*, has an advantage* $\mathsf{Adv}_{\Pi}^{\mathsf{Sound}}(\mathcal{A}) \geq \epsilon$.

3.3 Security Proofs of the UMTS-AKA Protocol

In this section, we analyse the UMTS-AKA protocol with respect to the five properties formalized in Section 3.2. The partner ID pid of an accepting client instance C_i is S (this reflects the lack of server identifiers); server instances S_i, have a pid corresponding to a unique UID. The session ID sid of each instance consists of: UID, R, and the value Sqn that is agreed upon during the session. In the absence of resynchronization, the session ID is $(\mathsf{UID}, \mathsf{R}, \mathsf{Sqn}_{\mathsf{Op,C}})$. During re-synchronization, the operator updates $\mathsf{Sqn}_{\mathsf{Op,C}}$ to the client's $\mathsf{Sqn}_{\mathsf{C}}$; this update is taken into account in the sid. Any two partners (same sid) with accepting states compute session keys ($\mathsf{CK}\|\mathsf{IK}$).

3.3.1 A Unitary Function G

We analyse the security of UMTS-AKA in two steps. First, we reduce the security of this protocol to the security (pseudorandomness) of an intermediate, unitary function G. This function models the suite of seven algorithms used in UMTS-AKA; each algorithm is a specific call to G. For the state-confidentiality property we also need to assume the pseudorandomness of the related unitary function G^*, which is the same as G, but we key it with the operator key $\mathsf{sk}_{\mathsf{Op}}$ rather than the client key sk_{C}. This first reduction gives a sufficient condition to provide UMTS-AKA security for any suite of algorithms intended to be used within it. As a second step, we prove that both current proposals for UMTS-AKA, i.e. TUAK and $\mathsf{MILENAGE}$, guarantee this property. We note

that the pseudorandomness of the unitary function G implies the pseudorandomness of each of the sub-algorithms of TUAK and MILENAGE. However, it is a strictly stronger property, which is necessary because we require the output of all the different functions to be independent.

MILENAGE **and** TUAK **as** G**.** In section 3.3.3, we prove that both the calls to the instantiations MILENAGE and TUAK algorithms can be modelled as the unitary function G that we use for our proofs. The last step in our proof is to show that both algorithm suites exhibit the PRF property we require for G, when instantiated with the key $\mathsf{sk_C}$ **and** with the operator key $\mathsf{sk_{Op}}$ (for the key-indistinguishability). However, as opposed to TUAK (whose symmetric design allows a lot more leeway), the MILENAGE algorithms require a stronger assumption to prove the PRF property when G is used with key $\mathsf{sk_{Op}}$. We note that the operator key causes a modelling problem. While common to all the clients of a certain operator, this key is only stored *implicitly* in the phone, and cannot be easily recovered (as we have detailed in appendix, it is based on the pseudorandomness of the function keyed with the operator key). We show that with or without operator-key corruptions, we can reduce the security of UMTS-AKA to the same underlying condition for the cryptographic algorithms.

Identities and reallocation. In our security analysis, we stick close to the original UMTS-AKA protocol. However, one simplification we make throughout this paper is associating each prover with a single, unique UID, which we consider public. In practice, this identifier is the user's IMSI, which can be requested by servers in case a TMSI value is not traceable to an IMSI. From the point of view of security, any attack initiated by mismatching TMSI values (i.e. replacing one value by another) is equivalent to doing the same thing with IMSI values. Another important part of the UMTS-AKA protocol that we abstract in this analysis is the TMSI reallocation. If the TMSI system were flawless (a newly-allocated TMSI is reliable and non-modifiable by an active MitM), then we could prove a stronger degree of server impersonation than in our current model. As we discuss in Section 3.2, an active MitM can inject false TMSI values, which make servers request an IMSI value; if the MitM reuses this value, it can use a type of offline relay to impersonate the server. In particular, the use of the TMSI is undone by the back door allowing servers to demand the IMSI; simultaneously, insecurities in using TMSIs translate to the identification by IMSI.

3.3.2 Provable Security Guarantees

The existing UMTS-AKA protocol only attains the weaker versions of key-indistinguishability, client-, and server-impersonation resistance. The protocol also guarantees state-confidentiality and soundness with respect to malicious servers. Denote by Π the UMTS-AKA protocol described in Section 2.3.1, but in which the calls to the internal cryptographic functions $\mathcal{F}_1, \ldots, \mathcal{F}_5, \mathcal{F}_1^*, \mathcal{F}_5^*$ are replaced by calls to the function $G : \{0,1\}^\kappa \times \{0,1\}^d \times \{0,1\}^t \times \{0,1\}^t \to \{0,1\}^n$, in which κ is a security parameter, d is a positive integer strictly larger than the size of the operator key, and t indicates the block size of an underlying pseudo-random permutation. Each input space is specified according the considered instantiations (MILENAGE or TUAK) detailed above. We denote by $\mathcal{S}_\mathsf{C} := \{0,1\}^\kappa$ the key-space for the client keys and by $\mathcal{S}_\mathsf{Op} := \{0,1\}^e$, the key space for operator keys, for some specified $e < d$ (in practice $e = 256$). Our system features $\mathsf{n_C}$ clients, $\mathsf{n_S}$ servers and n_Op operators.

Security statements. We proceed to give the five security statements with the respect to the UMTS-AKA protocol, in the following order: first, the notion of weak-key-indistinguishability, then strong-client-, and weak-server-impersonation resistance, soundness with respect to servers, and finally the state-confidentiality, which requires an additional assumption.

Theorem 1. **[**W.K.Ind**-resistance.]** *Let* $G : \{0,1\}^\kappa \times \{0,1\}^d \times \{0,1\}^t \times \{0,1\}^t \to \{0,1\}^n$ *be the function specified in section 3.3.1, and* Π *the modelled UMTS-AKA protocol described in section*

2.3.1. Consider a $(t, q_{\mathsf{exec}}, q_{\mathsf{res}}, q_G)$*-adversary* $\mathcal{A}$ *against the* W.K.Ind*-security of the protocol* Π*, running in time* t *and creating at most* q_{exec} *party instances with at most* q_{res} *resynchronizations per instance, and making* q_G *queries to the function* G*. Denote the advantage of this adversary as* $\mathsf{Adv}^{\mathsf{W.K.Ind}}_{\Pi}(\mathcal{A})$*. Then there exist a* $(t' \approx O(t), q' = q_G + q_{\mathsf{exec}}(2 + q_{\mathsf{res}}))$*-prf-adversary* $\mathcal{A}'$ *on* G *such that:*

$$\mathsf{Adv}^{\mathsf{W.K.Ind}}_{\Pi}(\mathcal{A}) \leq \mathsf{n_C} \cdot \left(\frac{q^2_{\mathsf{exec}}}{2^{|\mathsf{R}|}} + \mathsf{Adv}^{\mathsf{prf}}_{G}(\mathcal{A}') \right).$$

We show that this result holds (in fact with a simpler proof) even if operator key corruptions are possible. This is important, as it shows that even if an adversary knows all the operator keys, it is still unable to distinguish real session keys from random ones. This is also the case for the client- and server-impersonation statements below.

Proof. Our proof has the following hops.

Game $\mathbb{G}_0$: This game works as the W.K.Ind-game stipulated in our security model in section 3.2. The goal of the adversary $\mathcal{A}_{\mathbb{G}_0}$ is to distinguish, for a fresh instance that ends in an accepting state, the fresh session keys from random ones.

Game $\mathbb{G}_1$: We modify $\mathbb{G}_0$ by allowing $\mathcal{A}$ access to a new $\mathsf{Corrupt}(\mathsf{P}, \mathtt{type})$ oracle, depicted below. We note that this new query permits to consider the corruption of the key operator independently to the corruption of the subscriber keys. The new query behaves as follows:

$\mathsf{Corrupt}(\mathsf{P}, \mathtt{type})$: yields to the adversary the long-term keys of party $\mathsf{P} \neq \mathsf{S}$ (else, if the oracle takes as input $\mathsf{P} = \mathsf{S}$, then it behaves as usual calling the oracle $\mathsf{OpAccess}$). The output of the oracle depends on the value $\mathtt{type} \in \{\mathtt{sub}, \mathtt{op}, \mathtt{all}\}$. If $\mathtt{type} = \mathtt{sub}$, then the returned value is $\mathsf{sk_P}$. If $\mathtt{type} = \mathtt{op}$, then the oracle returns $\mathsf{sk_{Op}}$. Then, for $\mathtt{type} = \mathtt{all}$, we return the both values $\mathsf{sk_P}$, $\mathsf{sk_{Op}}$. If $\mathtt{type} \in \{\mathtt{sub}, \mathtt{all}\}$, then P (and all its instances, past, present, or future), are considered to be adversarially controlled.

We argue that given any adversary $\mathcal{A}$ playing the game $\mathbb{G}_1$ and winning with probability $\epsilon_{\mathcal{A}}$, the same adversary wins the game $\mathbb{G}_0$ with probability at least $\epsilon_{\mathcal{A}}$ (this is trivial since in game $\mathbb{G}_1$, $\mathcal{A}$ has more information).

$$\Pr[\mathcal{A} \text{ wins } \mathbb{G}_0] \quad \leq \quad \Pr[\mathcal{A} \text{ wins } \mathbb{G}_1].$$

Game $\mathbb{G}_2$: We modify $\mathbb{G}_1$ to only allow interactions with a single client (any future $\mathsf{CreateCl}$ calls for a client would be answered with an error symbol $\perp$). The challenger generates only a single operator key, which is associated with the operator chosen for the registered client, and chooses a bit $b \in \{0, 1\}$. We process as follows: for any adversary $\mathcal{A}_{\mathbb{G}_1}$ winning the game $\mathbb{G}_1$ with a non-negligible success probability $\epsilon_{\mathcal{A}_{\mathbb{G}_1}}$, we propose to construct an adversary $\mathcal{A}_{\mathbb{G}_2}$ winning the game $\mathbb{G}_2$ with a black-box access to the adversary $\mathcal{A}_{\mathbb{G}_1}$ as follows.

Adversary $\mathcal{A}_{\mathbb{G}_2}$ begins by choosing a single client C. For every user registration request that $\mathcal{A}_{\mathbb{G}_1}$ sends to its challenger, $\mathcal{A}_{\mathbb{G}_2}$ responds as follows: if the registered client is C, then it forwards the exact $\mathsf{CreateCl}$ query that $\mathcal{A}_{\mathbb{G}_1}$ makes to its own $\mathsf{CreateCl}$ oracle. Else, if $\mathcal{A}_{\mathbb{G}_1}$ registers any client $\mathsf{C}^* \neq \mathsf{C}$, $\mathcal{A}_{\mathbb{G}_2}$ simulates the registration, generating $\mathsf{sk_{C^*}}$ and $\mathsf{Sqn_{C^*}}$, returning the latter value. Adversary $\mathcal{A}_{\mathbb{G}_2}$ also generates $n_{\mathsf{Op}} - 1$ operator keys, and associates them with the clients as follows: the target client C is associated with the same operator given as input by $\mathcal{A}_{\mathbb{G}_1}$ to the $\mathsf{CreateCl}$ query (thus with the operator key $\mathsf{sk_{Op}}$ generated by the challenger of game $\mathbb{G}_2$). Let this target operator be denoted as Op. Adversary $\mathcal{A}_{\mathbb{G}_2}$ queries $\mathsf{Corrupt}(\mathsf{C}, \mathsf{op})$ and stores $\mathsf{sk_{Op}}$. We distinguish between two types of other clients. For all other clients C^* which are registered by $\mathcal{A}_{\mathbb{G}_1}$ with an operator $\mathsf{Op}^* \neq \mathsf{Op}$, adversary $\mathcal{A}_{\mathbb{G}_2}$ associates Op^* with one of its generated keys

$\mathsf{rsk}_{\mathsf{Op}^*}$. Recall that, since adversary $\mathcal{A}_{\mathbb{G}_1}$ plays the game in the presence of n_{Op} operators, there are $n_{\mathsf{Op}} - 1$ keys which will be used this way. We call all clients $\mathsf{C}^* \neq \mathsf{C}$ registered by $\mathcal{A}_{\mathbb{G}_0}$ with the target operator Op the *brothers* of the target client C. Adversary $\mathcal{A}_{\mathbb{G}_2}$ associates each brother of C with the corrupted key $\mathsf{sk}_{\mathsf{Op}}$ it learns from its challenger.

In the rest of the simulation, whenever $\mathcal{A}_{\mathbb{G}_1}$ makes a query to an instance of some party C^*, not a brother of C, the adversary $\mathcal{A}_{\mathbb{G}_2}$ simulates the response using the values $\mathsf{sk}_{\mathsf{C}^*}$, $\mathsf{rsk}_{\mathsf{Op}^*}$, and the current value of Sqn. For the brothers of C, the simulation is done with $\mathsf{sk}_{\mathsf{C}^*}$, $\mathsf{sk}_{\mathsf{Op}}$, and the current Sqn. For the target client C, any queries are forwarded by $\mathcal{A}_{\mathbb{G}_2}$ to its challenger.

Any corruption or reveal queries are dealt with in a similar way. Note that $\mathcal{A}_{\mathbb{G}_2}$ cannot query $\mathsf{Corrupt}$ to its adversary (this is a condition of freshness). The simulation is thus perfect up to the Test query.

In the Test query, $\mathcal{A}_{\mathbb{G}_1}$ chooses a *fresh session* and sends it to $\mathcal{A}_{\mathbb{G}_2}$ (acting as a challenger). Note that $\mathcal{A}_{\mathbb{G}_2}$ will be able to test whether this instance is fresh, as freshness is defined in terms of $\mathcal{A}_{\mathbb{G}_1}$'s queries. If $\mathcal{A}_{\mathbb{G}_1}$ queries Test with a client other than the target client C, then $\mathcal{A}_{\mathbb{G}_2}$ aborts the simulation, tests a random, fresh instance of the client C (creating one if necessary), and guesses the bit d, winning with probability at least $\frac{1}{2}$. Else, if $\mathcal{A}_{\mathbb{G}_1}$ queried a fresh instance of C, $\mathcal{A}_{\mathbb{G}_2}$ forwards this choice to its challenger and receives the challenger's input. The adversary $\mathcal{A}_{\mathbb{G}_2}$ forwards the input of the challenger to $\mathcal{A}_{\mathbb{G}_1}$ and then receives $\mathcal{A}$'s output d, which will be $\mathcal{A}_{\mathbb{G}_2}$'s own response to its own challenger.

Denote by E_1 the event that adversary tests C in game $\mathbb{G}_1$, while $\bar{\mathsf{E}_1}$ denotes the event that $\mathcal{A}_{\mathbb{G}_1}$ chooses to test $\mathsf{C}^* \neq \mathsf{C}$.

It holds that:

$$\begin{aligned}
\Pr[\mathcal{A}_{\mathbb{G}_2}\mathsf{wins}] &= \Pr[\mathcal{A}_{\mathbb{G}_2}\mathsf{wins} \mid \mathsf{E}_1] \cdot \Pr[\mathsf{E}_1] + \Pr[\mathcal{A}_{\mathbb{G}_2}\mathsf{wins} \mid \bar{\mathsf{E}_1}] \cdot \Pr[\bar{\mathsf{E}_1}] \\
&\geq \frac{1}{\mathsf{n_C}} \Pr[\mathcal{A}_{\mathbb{G}_1}\mathsf{wins}] + \frac{1}{2} \cdot \big(1 - \frac{1}{\mathsf{n_C}}\big) \\
&\geq \frac{1}{\mathsf{n_C}} \Pr[\mathcal{A}_{\mathbb{G}_0}\mathsf{wins}] + \frac{1}{2} \cdot \big(1 - \frac{1}{\mathsf{n_C}}\big).
\end{aligned}$$

Note that adversary $\mathcal{A}_{\mathbb{G}_2}$ makes one extra query with respect to $\mathcal{A}_{\mathbb{G}_1}$, since we need to learn the key of the target operator.

Game $\mathbb{G}_3$: We modify $\mathbb{G}_2$ to ensure that the random values sampled by honest server instances are always unique. This gives us a security loss (related to the respective collisions between the R in two different instances) of

$$\big|\Pr[\mathcal{A}_{\mathbb{G}_2}\mathsf{wins}] - \Pr[\mathcal{A}_{\mathbb{G}_3}\mathsf{wins}]\big| \leq \frac{q_{\mathsf{exec}}^2}{2^{|\mathsf{R}|}}.$$

Game $\mathbb{G}_4$: We modify $\mathbb{G}_3$ to replace outputs of the internal cryptographic functions by truly random, but consistent values (they are independent of the input, but the same input gives the same output). We argue that the security loss is precisely the advantage of the adversary $\mathcal{A}$ against the pseudorandomness of function G. Note that the total number of queries to the related functions is at most 2 G per honest instance(thus totalling at most $\mathsf{q}_G + q_{\mathsf{exec}}(2 + q_{\mathsf{res}})$ queries to the function G).

$$\big|\Pr[\mathcal{A}_{\mathbb{G}_3}\mathsf{wins}] - \Pr[\mathcal{A}_{\mathbb{G}_4}\mathsf{wins}]\big| \leq \mathsf{Adv}_G^{\mathsf{prf}}(\mathcal{A}).$$

Winning $\mathbb{G}_4$: At this point, the adversary plays a game in the presence of a single client C. The goal of this adversary is to distinguish a random session key to a fresh session key. But, in game $\mathbb{G}_4$, queries to G return truly random, consistent values. In this case, the adversary can do no better than guessing. Thus, we have: $\Pr[\mathcal{A}_{\mathbb{G}_4}\mathsf{wins}] = \frac{1}{2}$.

Security statement: This yields the following result:

$$\frac{1}{\mathsf{n_C}} \cdot \Pr[\mathcal{A}_{\mathbb{G}_0}\mathsf{wins}] + \frac{1}{2} \cdot (1 - \frac{1}{\mathsf{n_C}}) \leq \frac{q_{\mathsf{exec}}^2}{2^{|\mathsf{R}|}} + \mathsf{Adv}_G^{\mathsf{prf}}(\mathcal{A})$$
$$\Leftrightarrow \frac{1}{\mathsf{n_C}} \cdot \mathsf{Adv}_{\Pi}^{\mathsf{W.K.Ind}}(\mathcal{A}_{\mathbb{G}_0}) \leq \frac{q_{\mathsf{exec}}^2}{2^{|\mathsf{R}|}} + \mathsf{Adv}_G^{\mathsf{prf}}(\mathcal{A})$$
$$\Leftrightarrow \mathsf{Adv}_{\Pi}^{\mathsf{W.K.Ind}}(\mathcal{A}_{\mathbb{G}_0}) \leq \mathsf{n_C} \cdot (\frac{q_{\mathsf{exec}}^2}{2^{|\mathsf{R}|}} + \mathsf{Adv}_G^{\mathsf{prf}}(\mathcal{A}')).$$

This concludes the proof. □

Theorem 2. **[S.C.Imp-resistance.]** *Let $G : \{0,1\}^\kappa \times \{0,1\}^d \times \{0,1\}^t \times \{0,1\}^t \rightarrow \{0,1\}^n$ be the function specified in Section 3.3.1, and Π the modelled UMTS-AKA protocol specified in Section 2.3.1. Consider a $(t, q_{\mathsf{exec}}, q_{\mathsf{res}}, q_{\mathsf{s}}, q_{\mathsf{Op}}, q_G)$-adversary $\mathcal{A}$ against the S.C.Imp-security of the protocol Π, running in time t, and creating at most q_{exec} party instances with at most q_{res} resynchronizations per instance, corrupting at most q_{s} servers and making at most q_{Op} OpAccess queries per operator per corrupted server and making q_G queries to the function G. Denote the advantage of this adversary as $\mathsf{Adv}_{\Pi}^{\mathsf{S.C.Imp}}(\mathcal{A})$. Then there exists a $(t' \approx O(t), q' = 5 \cdot q_{\mathsf{Op}} \cdot q_{\mathsf{s}} + q_G + q_{\mathsf{exec}}(q_{\mathsf{res}} + 2))$-prf-adversary $\mathcal{A}'$ on G such that:*

$$\mathsf{Adv}_{\Pi}^{\mathsf{S.C.Imp}}(\mathcal{A}_{\mathbb{G}_0}) \leq \mathsf{n_C} \cdot \left(2 \cdot \mathsf{Adv}_G^{\mathsf{prf}}(\mathcal{A}') + \frac{(q_{\mathsf{exec}} + q_{\mathsf{s}} \cdot q_{\mathsf{Op}})^2}{2^{|\mathsf{R}|}} + \frac{q_{\mathsf{exec}} \cdot q_{\mathsf{res}}}{2^{|\mathsf{Res}|}} + \frac{1}{2^\kappa}\right).$$

Proof. Our proof has the following hops.

Game $\mathbb{G}_0$: This game works as the S.C.Imp-game: When the adversary $\mathcal{A}$ stops, it is said to win if there exists an instance S_i that ends in an accepting state with session and partner ID sid and pid such that: (a) pid is not adversarially controlled (its long-term key $\mathsf{sk}_{\mathsf{pid}}$ has not been corrupted), (b) no other instance C_i exists for $\mathsf{pid} = \mathsf{S}_i$ that ends in an accepting state, such that the both entities have the same session ID sid.

Game $\mathbb{G}_1$: This game works as the previous game $\mathbb{G}_0$, but including the new query Corrupt(P, type), i.e with the presence of operator keys corruption (as detailed in the previous proof). The reduction from the game $\mathbb{G}_0$ to the game $\mathbb{G}_1$ is the same as the proof of the Theorem 1. As before, it holds that:

$$\Pr[\mathcal{A}_{\mathbb{G}_0}\mathsf{wins}] \quad \leq \quad \Pr[\mathcal{A}_{\mathbb{G}_1}\mathsf{wins}].$$

Game $\mathbb{G}_2$: We modify $\mathbb{G}_1$ to only interact with a single client (any future CreateCl calls for a client would be answered with an error symbol $\perp$). The challenger only generates a single operator key, which is associated with the operator chosen for the registered client. As indicated before, the security loss is given by:

$$\Pr[\mathcal{A}_{\mathbb{G}_1}\mathsf{wins}] \quad \leq \quad \mathsf{n_C} \cdot \Pr[\mathcal{A}_{\mathbb{G}_2}\mathsf{wins}].$$

Game $\mathbb{G}_3$: We modify $\mathbb{G}_2$ to ensure that the random values sampled by any authentication challenge are always unique. This gives us a security loss (related to the collisions between the R in two different instances) of

$$\left|\Pr[\mathcal{A}_{\mathbb{G}_2}\mathsf{wins}] - \Pr[\mathcal{A}_{\mathbb{G}_3}\mathsf{wins}]\right| \leq \frac{(q_{\mathsf{exec}} + q_{\mathsf{s}} \cdot q_{\mathsf{Op}})^2}{2^{|\mathsf{R}|}}.$$

Game $\mathbb{G}_4$: This game behaves as the game $\mathbb{G}_3$ with the restriction to only interact with only

one server. As a consequence, the adversary loses the ability to obtain some authentication challenges from corrupted servers. We recall that the challenge is split in five parts: a random value, a masked version of the fresh sequence number (an one-time-pad based on an anonymity key generated by the function G), two mac computed with the function G and both session keys. Corrupted servers permit to obtain challenges based on the fresh sequence number but different random values. So the related security loss is given by the collision on two outputs of the same function G with two different inputs (the differences between both inputs are at least the value of the random) and by the indistinguishability of the function G which are guaranteed by the pseudorandomness of G. We recall that the Test Phase of the game can be only focus on a fresh server which is or was never corrupted. This give us a security loss

$$\left| \Pr[\mathcal{A}_{\mathbb{G}_4}\mathsf{wins}] - \Pr[\mathcal{A}_{\mathbb{G}_3}\mathsf{wins}] \right| \leq \mathsf{Adv}_G^{\mathsf{prf}}(\mathcal{A}).$$

Game $\mathbb{G}_5$: We modify $\mathbb{G}_4$ to replace outputs to calls to all the internal cryptographic functions by truly random, but consistent values (they are independent of the input, but the same input gives the same output). As detailed in the key-secrecy, we obtain:

$$\left| \Pr[\mathcal{A}_{\mathbb{G}_4}\mathsf{wins}] - \Pr[\mathcal{A}_{\mathbb{G}_5}\mathsf{wins}] \right| \leq \mathsf{Adv}_G^{\mathsf{prf}}(\mathcal{A}).$$

Winning $\mathbb{G}_5$: At this point, the adversary plays a game with a single client. A server instance S_i only accepts $\mathcal{A}_{\mathbb{G}_5}$, if this latter can generate a fresh an authentication response Res for some session sid. Assume that this happens against accepting instance S_i of the server, for some target session sid. Note that the value Res computed by C_i is purely random, but consistent. Thus, the adversary has three options for each of these values: (a) forwarding a value already received from the honest client for the same input values of which sk_C is unknown; (b) guessing the key sk_C; or (c) guessing the value. The first option yields no result, since it implies there exists a previous client instance with the same session id sid as the client.

The second option happens with a probability of $2^{-|\mathsf{sk}_\mathsf{C}|}$. The third option occurs with a probability of $2^{-|\mathsf{Res}|}$ per session (with or without resynchronization) per client, thus a total of $q_{\mathsf{exec}} \cdot 2^{-|\mathsf{Res}|}$. Thus,

$$\Pr[\mathcal{A}_{\mathbb{G}_5}\mathsf{wins}] = 2^{-|\mathsf{sk}_\mathsf{C}|} + q_{\mathsf{exec}} \cdot q_{\mathsf{res}} \cdot (2^{-|\mathsf{Res}|}).$$

Security statement: This yields the following result:

$$\mathsf{Adv}_\Pi^{\mathsf{S.C.Imp}}(\mathcal{A}_{\mathbb{G}_0}) \leq \mathsf{n}_\mathsf{C} \cdot (2 \cdot \mathsf{Adv}_G^{\mathsf{prf}}(\mathcal{A}') + \frac{(q_{\mathsf{exec}} + q_\mathsf{s} \cdot q_{\mathsf{Op}})^2}{2^{|\mathsf{R}|}} + \frac{q_{\mathsf{exec}} \cdot q_{\mathsf{res}}}{2^{|\mathsf{Res}|}} + \frac{1}{2^{|\mathsf{sk}_\mathsf{C}|}}).$$

□

Theorem 3. **[W.S.Imp-resistance.]** *Let $G : \{0,1\}^\kappa \times \{0,1\}^d \times \{0,1\}^t \times \{0,1\}^t \to \{0,1\}^n$ be the function specified in Section 3.3.1 and Π our protocol specified in Section 2.3.1. Consider a $(t, q_{\mathsf{exec}}, q_{\mathsf{res}}, q_\mathsf{G})$-adversary $\mathcal{A}$ against the W.S.Imp-security of the protocol Π, running in time t, creating at most q_{exec} party instances, running at most q_{res} re-synchronizations per each instance, and making at most q_G queries to the function G. Denote the advantage of this adversary as $\mathsf{Adv}_\Pi^{\mathsf{W.S.Imp}}(\mathcal{A})$. Then there exists a $(t' \approx t, q = q_{\mathsf{exec}} \cdot (q_{\mathsf{res}} + 2) + q_\mathsf{G})$-adversary $\mathcal{A}$' with an advantage $\mathsf{Adv}_G^{\mathsf{prf}}(\mathcal{A}')$ of winning against the pseudorandomness of the function G, such that:*

$$\mathsf{Adv}_\Pi^{\mathsf{W.S.Imp}}(\mathcal{A}) \leq \mathsf{n}_\mathsf{C} \cdot \left(\mathsf{Adv}_G^{\mathsf{prf}}(\mathcal{A}') + \frac{q_{\mathsf{exec}} \cdot q_{\mathsf{res}}}{2^{|\mathsf{Macs}|}} + \frac{1}{2^\kappa} \right).$$

Proof. We prove this statement in three steps, similarly to the previous W.K.Ind proof. We recall that the adversary cannot corrupt the server.

Game $\mathbb{G}_0$: This game works as the S.Imp-game stipulated in section 3.2.

Game $\mathbb{G}_1$: This game works as the previous game $\mathbb{G}_0$ but including the new query $\mathsf{Corrupt}(\mathsf{P}, \mathtt{type})$. This game is the same as the game $\mathbb{G}_0$ in the proof of the weak key-indistinguishability theorem. As before, it holds that:

$$\Pr[\mathcal{A}_{\mathbb{G}_0}\mathsf{wins}] \;\leq\; \Pr[\mathcal{A}_{\mathbb{G}_1}\mathsf{wins}].$$

Note that adversary $\mathcal{A}_{\mathbb{G}_1}$ makes no extra query.

Game $\mathbb{G}_2$: We modify $\mathbb{G}_1$ to only allow interactions with a single client. The challenger generates only a single operator key, which is associated with the operator chosen for the registered client. As indicated before the security loss is given by:

$$\Pr[\mathcal{A}_{\mathbb{G}_1}\mathsf{wins}] \leq \mathsf{n_C} \cdot \Pr[\mathcal{A}_{\mathbb{G}_2}\mathsf{wins}].$$

Game $\mathbb{G}_3$: We modify $\mathbb{G}_2$ to replace outputs to calls to the function G by truly random, but consistent values (they are independent of the input, but the same input gives the same output). As before, it holds that:

$$\big|\Pr[\mathcal{A}_{\mathbb{G}_2}\mathsf{wins}] - \Pr[\mathcal{A}_{\mathbb{G}_3}\mathsf{wins}]\big| \leq \mathsf{Adv}^{\mathsf{prf}}_{G}(\mathcal{A}').$$

Winning $\mathbb{G}_3$: At this point, the adversary plays a game against a single client C, which only accepts $\mathcal{A}_{\mathbb{G}_3}$, if $\mathsf{Mac_S}$ is verified for some session sid. Assume that this happens against accepting instance C_i of the target client, for some target session sid. Note that the MAC value $\mathsf{Mac_S}$ computed by C_i is purely random, but consistent. Thus, the adversary has three options: (a) forwarding a value already received from the honest server for the same input values $\mathsf{R}, \mathsf{Sqn}, \mathsf{sk_{Op}}, \mathsf{sk_C}$, of which $\mathsf{sk_C}$ is unknown; (b) guessing the key $\mathsf{sk_C}$; or (c) guessing the vector. The former option yields no result, since it implies a server instance with the same session id sid as the client. The second option happens with a probability of $2^{-|\mathsf{sk_C}|}$. The third option occurs with a probability of $2^{-|\mathsf{Mac_S}|}$ per session (which is to say per instance and per re-synchronization), thus a total of $q_{\mathsf{exec}} \cdot q_{\mathsf{res}} \cdot 2^{-|\mathsf{Mac_S}|}$. Thus,

$$\Pr[\mathcal{A}_{\mathbb{G}_3}\mathsf{wins}] = 2^{-|\mathsf{sk_C}|} + q_{\mathsf{exec}} \cdot q_{\mathsf{res}} \cdot 2^{-|\mathsf{Mac_S}|}.$$

Security statement: This yields the following result:

$$\mathsf{Adv}^{\mathsf{W.S.Imp}}_{\Pi}(\mathcal{A}_{\mathbb{G}_0}) \leq \mathsf{n_C} \cdot \big(2^{-|\mathsf{sk_C}|} + q_{\mathsf{exec}} \cdot q_{\mathsf{res}} \cdot 2^{-|\mathsf{Mac_S}|} + \mathsf{Adv}^{\mathsf{prf}}_{G}(\mathcal{A}')\big).$$

□

Theorem 4. **[Sound-security.]** *Let* $G : \{0,1\}^{\kappa} \times \{0,1\}^{d} \times \{0,1\}^{t} \times \{0,1\}^{t} \to \{0,1\}^{n}$ *be our specified function in Section 3.3.1, and* Π *the protocol specified in Section 2.3.1. Consider a* $(t, q_{\mathsf{exec}}, q_{\mathsf{res}}, q_{\mathsf{Op}}, q_{\mathsf{G}}, \epsilon)$*-server-sound-adversary* $\mathcal{A}$ *against the soundness of the protocol* Π*, running in time* t*, creating at most* q_{exec} *party instances with at most* q_{res} *resynchronizations per instance, making at most* q_{Op} *queries to any operator* Op *and at most* q_{G} *queries to the function* G*. Denote the advantage of this adversary as* $\mathsf{Adv}^{\mathsf{Sound}}_{\Pi}(\mathcal{A})$*. Then there exists a* $(t' \approx t, q' = 5 \cdot q_{\mathsf{Op}} + q_G + q_{\mathsf{exec}}(2 + q_{\mathsf{res}}))$*-adversary* $\mathcal{A}'$ *with an advantage* $\mathsf{Adv}^{\mathsf{prf}}_{G}(\mathcal{A}')$ *of winning against the pseudorandomness of the function* G*, such that:*

$$\mathsf{Adv}^{\mathsf{Sound}}_{\Pi}(\mathcal{A}) \leq \mathsf{n_C} \cdot \left(2 \cdot \mathsf{Adv}^{\mathsf{prf}}_{G}(\mathcal{A}') + \frac{q_{\mathsf{exec}} \cdot q_{\mathsf{res}}}{2^{|\mathsf{Mac_S}|}} + \frac{1}{2^{\kappa}}\right).$$

Proof. Our proof has the following hops.

Game $\mathbb{G}_0$: This game works as the game Sound-game stipulated in our security model. The goal of this adversary $\mathcal{A}_{\mathbb{G}_0}$ is similar as that of the S.Imp-game but with a different adversary; indeed in the S.Imp-game is a MitM adversary and in the Sound-game, we have a *legitimate-but-malicious* server-adversary.

Game $\mathbb{G}_1$: We consider the game $\mathbb{G}_1$ as the S.S.Imp-game (as previously detailed) but including the new query $\mathsf{Corrupt}(\mathsf{P}, \mathtt{type})$, i.e. allowing the corruption of operator keys, which we allowed in previous proofs. We proceed to show that, for any adversary $\mathcal{A}_{\mathbb{G}_0}$ winning the game $\mathbb{G}_0$ with an advantage $\mathsf{Adv}^{\mathsf{Sound}}_{\Pi}(\mathcal{A}_{\mathbb{G}_0})$, there exists an adversary $\mathcal{A}_{\mathbb{G}_1}$ with black-box access to the adversary $\mathcal{A}_{\mathbb{G}_0}$ which wins game $\mathbb{G}_1$. Both adversaries play their related games with oracle access to: Send, CreateCl, Init, Execute, Reveal, and StReveal, but with distinct corruption queries. For each former queries made by $\mathcal{A}_{\mathbb{G}_0}$, the adversary $\mathcal{A}_{\mathbb{G}_1}$ forwards these queries to its own challenger and sends to $\mathcal{A}_{\mathbb{G}_0}$ the related answers. Now focus on the two last oracles which can be used by the adversary $\mathcal{A}_{\mathbb{G}_0}$: OpAccess and Corrupt.

First, recall that the OpAccess oracle in game $\mathbb{G}_0$ takes in input a client identifier and outputs, for our protocol, an authentication vector composed by the tuple $\mathsf{AV} = (\mathsf{R}, \mathsf{Autn}, \mathsf{Mac}_{\mathsf{C}}, \mathsf{CK}, \mathsf{IK})$. To simulate the answer of the oracle $\mathsf{OpAccess}(\mathsf{C}_i)$, $\mathcal{A}_{\mathbb{G}_1}$ uses the query $\mathsf{Execute}(S, \mathsf{C}_i)$ (with the server related to the *legitimate-but-malicious* adversary) and $\mathsf{Reveal}(\mathsf{C}, i)$.

Now, focus on the simulation of the Corrupt queries response. We recall that we have two possible inputs: a client or an operator. If the Corrupt oracle takes in input a client, the adversary $\mathcal{A}_{\mathbb{G}_1}$ uses its own Corrupt oracle to obtain the related answer. If the input is an operator, $\mathcal{A}_{\mathbb{G}_1}$ needs to provide the following values: the operator key $\mathsf{sk}_{\mathsf{Op}}$, and for each client of this operator the tuple $(\mathsf{UID}, \mathsf{sk}_{\mathsf{UID}}, \mathsf{st}_{\mathsf{Op},\mathsf{C}})$. To simulate such an answer, $\mathcal{A}_{\mathbb{G}_1}$ uses its own $\mathsf{Corrupt}(\mathsf{C})$ and $\mathsf{StReveal}(\mathsf{C}, i, 1)$ for each client C of this operator. So at this point, the adversary $\mathcal{A}_{\mathbb{G}_1}$ can simulate any query from the adversary $\mathcal{A}_{\mathbb{G}_0}$. At the end of the simulation, the adversary $\mathcal{A}_{\mathbb{G}_1}$ replays the impersonation's attempt from the adversary $\mathcal{A}_{\mathbb{G}_0}$. Thus, we have:

$$\Pr[\mathcal{A}_{\mathbb{G}_0}\mathsf{wins}] = \Pr[\mathcal{A}_{\mathbb{G}_1}\mathsf{wins}].$$

Winning game $\mathbb{G}_1$: This game follows the game $\mathbb{G}_1$ described in the reduction proof of the theorem S.S.Imp. Thus, we have:

$$\mathsf{Adv}^{\mathsf{S.S.Imp}}_{\Pi}(\mathcal{A}_{\mathbb{G}_1}) \leq \mathsf{n}_{\mathsf{C}} \cdot (2 \cdot \mathsf{Adv}^{\mathsf{prf}}_{G}(\mathcal{A}') + \frac{q_{\mathsf{exec}} \cdot q_{\mathsf{res}}}{2^{|\mathsf{Mac_S}|}} + \frac{1}{2^{\kappa}}).$$

□

Theorem 5. **[St.Conf-resistance.]** *Let G and G^* be the functions specified in Section 3.3.1, and Π our modelled UMTS-AKA protocol specified in Section 2.3.1. Consider a $(t, q_{\mathsf{exec}}, q_{\mathsf{res}}, q_{\mathsf{Op}}, q_G, q_{G^*})$-adversary $\mathcal{A}$ against the St.Conf-security of the protocol Π, running in time t and creating at most q_{exec} party instances with at most q_{res} resynchronizations per instance, making at most q_{Op} queries to oracle OpAccess, and q_G (resp. q_{G^*}) queries to the function G (resp. G^*). Denote the advantage of this adversary as $\mathsf{Adv}^{\mathsf{St.Conf}}_{\Pi}(\mathcal{A})$. Then there exist a $(t' \approx O(t), q' = q_G + q_{\mathsf{exec}}(5 + q_{\mathsf{res}}))$-adversary $\mathcal{A}_1$ and $(t' \approx O(t), q' = q_{G^*})$-adversary $\mathcal{A}_2$ against respectively the pseudorandomness of the functions G and G^* such that:*

$$\mathsf{Adv}^{\mathsf{St.Conf}}_{\Pi}(\mathcal{A}) \leq \mathsf{n}_{\mathsf{C}} \cdot \left(\frac{1}{2^{|\mathsf{sk_C}|}} + \frac{1}{2^{|\mathsf{sk_{Op}}|}} + \frac{2}{2^{|\mathsf{Sqn}|}} + \mathsf{Adv}^{\mathsf{prf}}_{G}(\mathcal{A}_1) + \mathsf{Adv}^{\mathsf{prf}}_{G^*}(\mathcal{A}_2) \right).$$

Proof. Our proof has the following hops.

Game $\mathbb{G}_0$: This game works as the St.Conf-game stipulated in our security model. The goal

of the adversary $\mathcal{A}_{\mathbb{G}_0}$ is to recover at least one secret value, i.e. the subscriber key $\mathsf{sk_C}$, the operator key $\mathsf{sk_{Op}}$, or the subscriber sequence number $\mathsf{Sqn_C}$ for a fresh instance.

Game $\mathbb{G}_1$: We modify $\mathbb{G}_0$ to only allow interactions with one operator. The challenger related to the game $\mathbb{G}_1$ only generates a single operator key, which is associated with the operator chosen for the registered client. We proceed as follows: for any adversary $\mathcal{A}_{\mathbb{G}_0}$ winning the game $\mathbb{G}_0$ with a no-negligible success probability $\epsilon_{\mathcal{A}_{\mathbb{G}_0}}$, we propose to construct an adversary $\mathcal{A}_{\mathbb{G}_1}$ winning the game $\mathbb{G}_1$ with a black-box access to the adversary $\mathcal{A}_{\mathbb{G}_0}$.

Adversary $\mathcal{A}_{\mathbb{G}_1}$ begins by choosing a single operator Op. It generates $n_{\mathsf{Op}} - 1$ operator keys, denoted $\mathsf{rsk_{Op^*}}$. Then, for every user registration request that $\mathcal{A}_{\mathbb{G}_0}$ sends to its challenger, $\mathcal{A}_{\mathbb{G}_1}$ responds as follows: if the request $\mathsf{CreateCl}(.)$ takes in input the operator Op, then it forwards the same query to its own oracle. Else, if $\mathcal{A}_{\mathbb{G}_0}$ sends a registration request based on any operator $\mathsf{Op^*} \neq \mathsf{Op}$, $\mathcal{A}_{\mathbb{G}_1}$ simulates the registration, generating a subscriber key $\mathsf{sk_{C^*}}$ and a sequence number $\mathsf{Sqn_{C^*}}$, returning the latter value. Moreover, each new client registered with the operator Op (resp. any $\mathsf{Op^*}$) is associated with the related operator key $\mathsf{sk_{Op}}$(resp. $\mathsf{rsk_{Op^*}}$).

We distinguish between two types of clients: we denote $\mathsf{C^*}$ the clients which are registered with an operator $\mathsf{Op^*} \neq \mathsf{Op}$, and C the ones with the operator Op.

In the rest of the simulation, whenever $\mathcal{A}_{\mathbb{G}_0}$ makes a query to an instance of some party $\mathsf{C^*}$ (from any operator except Op), the adversary $\mathcal{A}_{\mathbb{G}_1}$ simulates the response using the values $\mathsf{sk_{C^*}}$, $\mathsf{rsk_{Op^*}}$, and the current value of $\mathsf{Sqn_{C^*}}$. For the other clients, the query is forwarded by $\mathcal{A}_{\mathbb{G}_1}$ to its own challenger. Any corruption or reveal queries are dealt with in a similar way. Note that $\mathcal{A}_{\mathbb{G}_1}$ cannot query $\mathsf{Corrupt}$ to its adversary (this is a condition of freshness). The simulation is thus perfect up to the Test query.

In the Test query, $\mathcal{A}_{\mathbb{G}_0}$ chooses a $\mathsf{fresh\ instance}$ and sends it to $\mathcal{A}_{\mathbb{G}_1}$ (acting as a challenger). Note that $\mathcal{A}_{\mathbb{G}_1}$ will be able to test whether this instance is fresh, as freshness is defined in terms of $\mathcal{A}_{\mathbb{G}_0}$'s queries. If $\mathcal{A}_{\mathbb{G}_0}$ queries an instance C_i^* for the Test query ,then $\mathcal{A}_{\mathbb{G}_1}$ aborts the simulation, tests a random tuple about any fresh instance of the client C (creating one if necessary), winning with probability $\frac{1}{2^{|\mathsf{sk_C}|}} + \frac{1}{2^{|\mathsf{sk_{Op}}|}} + \frac{1}{2^{|\mathsf{Sqn_C}|}} + \frac{1}{2^{|\mathsf{Sqn_{Op,C}}|}}$. Else, if $\mathcal{A}_{\mathbb{G}_0}$ sends a tuple of a fresh instance of C_i, $\mathcal{A}_{\mathbb{G}_1}$ forwards this choice to its challenger and receives the challenger's output which contains the result of this game.

Denote by E_1 the event that adversary $\mathcal{A}_{\mathbb{G}_0}$ tests an instance C_i (from the chosen operator Op), while $\bar{\mathsf{E}_1}$ denotes the event that $\mathcal{A}_{\mathbb{G}_0}$ chooses to test C_i^*.

It holds that:

$$\begin{aligned}\Pr[\mathcal{A}_{\mathbb{G}_1}\mathsf{wins}] &= \Pr[\mathcal{A}_{\mathbb{G}_1}\mathsf{wins} \mid \mathsf{E}_1] \cdot \Pr[\mathsf{E}_1] + \Pr[\mathcal{A}_{\mathbb{G}_1}\mathsf{wins} \mid \bar{\mathsf{E}_1}] \cdot \Pr[\bar{\mathsf{E}_1}] \\ &\geq \frac{1}{n_{\mathsf{Op}}} \Pr[\mathcal{A}_{\mathbb{G}_0}\mathsf{wins}] + \big(1 - \frac{1}{n_{\mathsf{Op}}}\big) \cdot \big(\frac{1}{2^{|\mathsf{sk_C}|}} + \frac{1}{2^{|\mathsf{sk_{Op}}|}} + \frac{2}{2^{|\mathsf{Sqn}|}}\big).\end{aligned}$$

That implies: $\Pr[\mathcal{A}_{\mathbb{G}_0}\mathsf{wins}] \leq n_{\mathsf{Op}} \cdot \Pr[\mathcal{A}_{\mathbb{G}_1}\mathsf{wins}]$.

Game $\mathbb{G}_2$: We modify $\mathbb{G}_1$ to only allow interactions with a single client (any future $\mathsf{CreateCl}$ (Op) calls for a client would be answered with an error symbol $\perp$). We recall that the two adversaries $\mathcal{A}_{\mathbb{G}_1}$ and $\mathcal{A}_{\mathbb{G}_2}$ interact with clients from a single operator key, denoted Op, which is associated with the operator key $\mathsf{sk_{Op}}$. We proceed as follows: for any adversary $\mathcal{A}_{\mathbb{G}_1}$ winning the game $\mathbb{G}_2$ with a no-negligible success probability $\epsilon_{\mathcal{A}_{\mathbb{G}_1}}$. we propose to construct an adversary $\mathcal{A}_{\mathbb{G}_2}$ winning the game $\mathbb{G}_2$ with a black-box access to the adversary $\mathcal{A}_{\mathbb{G}_1}$. Adversary $\mathcal{A}_{\mathbb{G}_2}$ begins by choosing a single client C. For every user registration request that $\mathcal{A}_{\mathbb{G}_1}$ sends to its challenger, $\mathcal{A}_{\mathbb{G}_2}$ responds as follows: for a new client $\mathsf{C^*} \neq \mathsf{C}$ it generates $\mathsf{sk_{C^*}}$ and $\mathsf{Sqn_{C^*}}$, returning the latter value. In the rest of the simulation, whenever $\mathcal{A}_{\mathbb{G}_1}$ makes a query to an instance of some party $\mathsf{C^*}$, the adversary $\mathcal{A}_{\mathbb{G}_2}$ simulates the response using the oracle of the function G^* and the values $\mathsf{sk_{C^*}}$ and the current value of $\mathsf{Sqn_{C^*}}$. For the target client C, any queries are forwarded by $\mathcal{A}_{\mathbb{G}_2}$ to its challenger. Any corruption or reveal queries are dealt with in a similar way. Note that $\mathcal{A}_{\mathbb{G}_2}$

cannot query Corrupt to its adversary (this is a condition of freshness). The simulation is thus perfect up to the Test query. In the Test query, $\mathcal{A}_{\mathbb{G}_1}$ chooses a fresh instance and sends it to $\mathcal{A}_{\mathbb{G}_2}$ (acting as a challenger). Note that $\mathcal{A}_{\mathbb{G}_2}$ will be able to test whether this instance is fresh, as freshness is defined in terms of $\mathcal{A}_{\mathbb{G}_1}$'s queries. If $\mathcal{A}_{\mathbb{G}_1}$ queries Test with a client other than the target client C, then $\mathcal{A}_{\mathbb{G}_2}$ aborts the simulation, tests a random tuple as the previous reduction. Else, if $\mathcal{A}_{\mathbb{G}_1}$ queried a fresh instance of C, $\mathcal{A}_{\mathbb{G}_2}$ forwards this choice to its challenger and receives the challenger's which contains the result of this game. It holds that:

$$\Pr[\mathcal{A}_{\mathbb{G}_1}\mathsf{wins}] \leq \mathsf{n}_{\mathsf{C},\mathsf{Op}} \cdot \Pr[\mathcal{A}_{\mathbb{G}_2}\mathsf{wins}],$$

with $\mathsf{n}_{\mathsf{C},\mathsf{Op}}$ denoting the maximum number of clients per operator.

Game $\mathbb{G}_3$: We modify $\mathbb{G}_2$ to replace outputs of the internal cryptographic functions by truly random, but consistent values (they are independent of the input, but the same input gives the same output). We argue that the security loss is precisely the advantage of the adversary $\mathcal{A}$ against the pseudorandomness of functions G and G^*. Note that the total number of queries to the related functions are at most $\mathsf{q}_G + q_{\mathsf{exec}}(5 + q_{\mathsf{res}})$ queries to the function G.

$$\left|\Pr[\mathcal{A}_{\mathbb{G}_3}\mathsf{wins}] - \Pr[\mathcal{A}_{\mathbb{G}_2}\mathsf{wins}]\right| \leq \mathsf{Adv}^{\mathsf{prf}}_{G}(\mathcal{A}) + \mathsf{Adv}^{\mathsf{prf}}_{G^*}(\mathcal{A}).$$

Winning Game $\mathbb{G}_3$: At this point, the adversary plays a game with an uncorruptible single client C_i in a protocol including truly but consistent values. She wins if she can output a tuple $(\mathsf{C}_i, \mathsf{sk}^*_{\mathsf{C}}, \mathsf{sk}^*_{\mathsf{Op}}, \mathsf{Sqn}_{\mathsf{C}}{}^*, \mathsf{Sqn}_{\mathsf{Op},\mathsf{C}}{}^*)$ such as at least one of these values corresponds to the real related secret value of the instance C_i. Thus, the adversary has only one choice to win this game: guessing each value. So the probability that the adversary $\mathcal{A}_{\mathbb{G}_3}$ wins is as follows:

$$\Pr[\mathcal{A}_{\mathbb{G}_3}\mathsf{wins}] = \frac{1}{2^{|\mathsf{sk}_{\mathsf{C}}|}} + \frac{1}{2^{|\mathsf{sk}_{\mathsf{Op}}|}} + \frac{2}{2^{|\mathsf{Sqn}|}}.$$

□

3.3.3 The Pseudorandomness of the Internal Cryptographic Functions

Two sets of algorithms have been standardized to be used in the AKA protocols: MILENAGE and TUAK. In this section, we prove the pseudorandomness of TUAK and MILENAGE algorithms.

The security of TUAK **algorithms.** In order to prove the pseudorandomness of the TUAK algorithms, we assume that the truncated keyed internal Keccak permutation is a good pseudorandom function. We propose two generic constructions to model the TUAK algorithms: a first one, denoted G_{tuak} when the secret is based on the subscriber key sk_{C} and a second one, denoted G^*_{tuak} when the function is keyed with the operator key $\mathsf{sk}_{\mathsf{Op}}$. It is worth noting that the construction of the TUAK functions is reminiscent of the Merkle-Damgård construction, where the output of function f is an input of the next iteration of the function f. This is in contradiction with the Sponge construction used in the Keccak hash function, given the internal permutation $\mathsf{f}_{\mathsf{Keccak}}$. Thus, we clarify that this security proof does not directly imply an innovation on Keccak construction. We model the truncated keyed internal permutation of Keccak by functions f and f^*:

$$f(K, x\|y, i, j) = \lfloor \mathsf{f}_{\mathsf{Keccak}}(x\|K\|y)\rfloor_{i..j}, f^*(K^*, x^*\|y^*, i, j) = \lfloor \mathsf{f}_{\mathsf{Keccak}}(K^*\|x^*\|y^*)\rfloor_{i..j},$$

with $x \in \{0,1\}^{512}$, $K, K^* \in \{0,1\}^{\kappa}$, $y \in \{0,1\}^{1088-\kappa}$, $x^* \in \{0,1\}^{512+\kappa}$, $y^* \in \{0,1\}^{1088}$ and $i, j \in \{0,1\}^t$ with $\log_2(t-1) < 1600 \leq \log_2(t)$. We note that $\forall K, x, x^*, y, y^*, i, j$ such as $x = K^*\|x^*$ and $y^* = K\|y$, we have $f(K, x\|y, i, j) = f^*(K^*, x^*\|y^*, i, j)$. The input x (resp. x^*) can be viewed as the chaining variable of the cascade construction of G_{tuak} given f (resp. f^*),

y (resp. y^*) is an auxiliary input of the function, and i and j define the size of the truncation. The construction G_{tuak} and G^*_{tuak} act as a generalization of the TUAK algorithms:

$$\begin{aligned}
\mathcal{F}_1(\mathsf{sk_{Op}}, \mathsf{sk_C}, \mathsf{R}, \mathsf{Sqn}, \mathsf{AMF}) &= G_{\mathsf{tuak}}(\mathsf{sk_C}, \mathsf{inp}_1, 0, 127) = G^*_{\mathsf{tuak}}(\mathsf{sk_{Op}}, {\mathsf{inp}^*}_1, 0, 127),\\
\mathcal{F}^*_1(\mathsf{sk_{Op}}, \mathsf{sk_C}, \mathsf{R}, \mathsf{Sqn}, \mathsf{AMF}) &= G_{\mathsf{tuak}}(\mathsf{sk_C}, \mathsf{inp}_2, 0, 127) = G^*_{\mathsf{tuak}}(\mathsf{sk_{Op}}, \mathsf{inp}^*_2, 0, 127),\\
\mathcal{F}_2(\mathsf{sk_{Op}}, \mathsf{sk_C}, \mathsf{R}) &= G_{\mathsf{tuak}}(\mathsf{sk_C}, \mathsf{inp}_3, 0, 127) = G^*_{\mathsf{tuak}}(\mathsf{sk_{Op}}, \mathsf{inp}^*_3, 0, 127),\\
\mathcal{F}_3(\mathsf{sk_{Op}}, \mathsf{sk_C}, \mathsf{R}) &= G_{\mathsf{tuak}}(\mathsf{sk_C}, \mathsf{inp}_3, 256, 383) = G^*_{\mathsf{tuak}}(\mathsf{sk_{Op}}, \mathsf{inp}^*_3, 256, 383),\\
\mathcal{F}_4(\mathsf{sk_{Op}}, \mathsf{sk_C}, \mathsf{R}) &= G_{\mathsf{tuak}}(\mathsf{sk_C}, \mathsf{inp}_3, 512, 639) = G^*_{\mathsf{tuak}}(\mathsf{sk_{Op}}, \mathsf{inp}^*_3, 512, 639),\\
\mathcal{F}_5(\mathsf{sk_{Op}}, \mathsf{sk_C}, \mathsf{R}) &= G_{\mathsf{tuak}}(\mathsf{sk_C}, \mathsf{inp}_3, 768, 815) = G^*_{\mathsf{tuak}}(\mathsf{sk_{Op}}, \mathsf{inp}^*_3, 768, 815),\\
\mathcal{F}^*_5(\mathsf{sk_{Op}}, \mathsf{sk_C}, \mathsf{R}) &= G_{\mathsf{tuak}}(\mathsf{sk_C}, \mathsf{inp}_4, 768, 815) = G^*_{\mathsf{tuak}}(\mathsf{sk_{Op}}, \mathsf{inp}^*_4, 768, 815),
\end{aligned}$$

with:

$\mathsf{inp}_1 = \mathsf{sk_{Op}}\|\mathsf{cst}_1\|\mathsf{cst}_5$,, $\mathsf{inp}_2 = \mathsf{sk_{Op}}\|\mathsf{cst}_1\|\mathsf{cst}_5$,
$\mathsf{inp}_3 = \mathsf{sk_{Op}}\|\mathsf{cst}_3\|\mathsf{cst}_5$,, $\mathsf{inp}_4 = \mathsf{sk_{Op}}\|\mathsf{cst}_4\|\mathsf{cst}_5$,
$\mathsf{inp}^*_1 = \mathsf{cst}_1\|\mathsf{keys}\|\mathsf{cst}_5$,, $\mathsf{inp}^*_2 = \mathsf{cst}_1\|\mathsf{keys}\|\mathsf{cst}_5$,
$\mathsf{inp}^*_3 = \mathsf{cst}_3\|\mathsf{keys}\|\mathsf{cst}_5$,, $\mathsf{inp}^*_4 = \mathsf{cst}_4\|\mathsf{keys}\|\mathsf{cst}_5$,
$\mathsf{cst}_1 = \mathsf{Inst}\|\mathsf{AN}\|0^{192}\|(\mathsf{Inst}'\|\mathsf{AN}\|\mathsf{R}\|\mathsf{AMF}\|\mathsf{Sqn})$,
$\mathsf{cst}_2 = \mathsf{Inst}\|\mathsf{AN}\|0^{192}\|(\mathsf{Inst}^{(2)}\|\mathsf{AN}\|\mathsf{R}\|\mathsf{AMF}\|\mathsf{Sqn})$,
$\mathsf{cst}_3 = \mathsf{Inst}\|\mathsf{AN}\|0^{192}\|(\mathsf{Inst}'\|\mathsf{AN}\|\mathsf{R}\|0^{64})$,
$\mathsf{cst}_4 = \mathsf{Inst}\|\mathsf{AN}\|0^{192}\|(\mathsf{Inst}^{(3)}\|\mathsf{AN}\|\mathsf{R}\|0^{64})$,
$\mathsf{cst}_5 = \mathsf{Pad}\|1\|0^{192}$,

We define the cascade construction G_{tuak} based on the function f as follows:

$$\begin{aligned}
G_{\mathsf{tuak}}(K, \mathsf{val}, i, j) &= f(K, f(K, \mathsf{val}_1\|\mathsf{val}_3, 0, 256)\|\mathsf{val}_2\|\mathsf{val}_3, i, j),\\
G^*_{\mathsf{tuak}}(K^*, \mathsf{val}^*, i, j) &= f^*(f^*, \mathsf{val}^*_1\|\mathsf{val}^*_3, 0, 256), \mathsf{val}^*_2\|\mathsf{val}^*_3, i, j),
\end{aligned}$$

with G_{tuak} and G^*_{tuak} defined on the domain $\{0,1\}^\kappa \times \{0,1\}^d \times \{0,1\}^t \times \{0,1\}^t$ with range $\{0,1\}^n$, $\mathsf{val} = (\mathsf{val}_1\|\mathsf{val}_2)\|\mathsf{val}_3 \in \{0,1\}^{512} \times \{0,1\}^{256} \times \{0,1\}^{(832-\kappa)}$, $\mathsf{val}^* = (\mathsf{val}^*_1\|\mathsf{val}^*_2)\|\mathsf{val}^*_3 \in \{0,1\}^{256} \times \{0,1\}^{256} \times \{0,1\}^{(1088-\kappa)}$ two known values with $n = j-i$, $d = 1600-\kappa$, $\kappa = |K|$ and $\log_2(t-1) < 1600 \leq \log_2(t)$, K a secret value and $0 \leq i \leq j \leq 1600$. We express the required security properties of the generalization G_{tuak} (resp. G^*_{tuak}) in terms of the pseudorandomness of the function f (resp. f^*). Since the construction of the two functions, while we cannot prove the latter property, we can conjecture that the advantage of a prf-adversary would be of the form:

$$\mathsf{Adv}^{\mathsf{prf}}_{f^*}(\mathcal{A}) = \mathsf{Adv}^{\mathsf{prf}}_{f}(\mathcal{A}) \leq \mathsf{c}_1 \cdot \frac{t/T_f}{2^{|K|}} + \mathsf{c}_2 \cdot \frac{q \cdot t/T_f}{2^{1600-m}},$$

for any adversary $\mathcal{A}$ running in time t and making at most q queries to its challenger. Here, m is the output's size of our function f and T_f is the time to do one f computation on the fixed RAM model of computation. Furthermore, c_1 and c_2 are two constants depending only on this model. In other words, we assume that the best attacks are either a exhaustive key search or a specific attack on this construction. This attack uses the fact that the permutation is public and can be easily inverted. Even if the protocol truncates the permutation, if the output values are large, and an exhaustive search on the missing bits is performed, it is possible to invert the permutation and recover the inputs. Since the secret key is one of the inputs, together with some known values, it is possible to determine which guesses of the exhaustive search are correct. Finally, if the known inputs are shorter than the truncation, false positives can happen due to collisions and we have to

be filtered. However, if the number of queries is large enough, it is possible to filter these bad guesses and uniquely recover the keys.

Pseudorandomness of TUAK algorithms. We begin by reducing the pseudorandomness of G_{tuak} to the prf-security of the function f. This implies the pseudorandomness of each TUAK algorithm. Recall that our main assumption is that the function f is pseudorandom if the Keccak permutation is a good random permutation.

Theorem 6. [prf-security for G^*_{tuak}.] *Let $G^*_{\mathsf{tuak}} : \{0,1\}^\kappa \times \{0,1\}^e \times \{0,1\}^{d-e} \times \{0,1\}^t \times \{0,1\}^t \to \{0,1\}^n$ and $f^* : \{0,1\}^\kappa \times \{0,1\}^e \times \{0,1\}^d \times \{0,1\}^t \times \{0,1\}^t \to \{0,1\}^m$ be the two functions specified above. Consider a (t,q)-adversary $\mathcal{A}$ against the prf-security of the function G^*_{tuak}, running in time t and making at most q queries to its challenger. Denote the advantage of this adversary as $\mathsf{Adv}^{\mathsf{prf}}_{G^*_{\mathsf{tuak}}}(\mathcal{A})$. Then there exists a $(t' \approx O(t), q' = q)$-adversary $\mathcal{A}$' with an advantage $\mathsf{Adv}^{\mathsf{prf}}_{f^*}(\mathcal{A}')$ of winning against the pseudorandomness of f^* such that:*

$$\mathsf{Adv}^{\mathsf{prf}}_{G^*_{\mathsf{tuak}}}(\mathcal{A}) = \mathsf{Adv}^{\mathsf{prf}}_{f^*}(\mathcal{A}'),$$

Proof. We construct the adversary $\mathcal{A}_{f^*}$ using a prf-adversary $\mathcal{A}_{G^*}$. The latter uses $\mathcal{A}_{f^*}$ as a challenger for a prf-game $\mathbb{G}_{\mathsf{prf}(f^*)}$ and can only communicate with $\mathcal{A}_{f^*}$ whereas $\mathcal{A}_{f^*}$ has access to a challenger for f^*. To begin with, the challenger $\mathcal{C}^{\mathsf{prf}}_{f^*}$ chooses a bit b and a private $\mathsf{sk_{Op}} \in \{0,1\}^\kappa$. If $b = 0$, it assigns f^* to a random function and if $b = 1$, it assigns f^* to the specific internal function. The adversary $\mathcal{A}_{f^*}$ expects queries of the form (m,a,b) from $\mathcal{A}_{G^*}$, with $m = m^{(1)}\|m^{(2)}\|m^{(3)} \in \{0,1\}^d$, and $a,b \in \{0,1\}^t$. Then, $\mathcal{A}_{f^*}$ responds as follows:

- It queries its challenger $\mathcal{C}^{\mathsf{prf}}_{f^*}$ on input $(m^{(1)}\|m^{(3)}, 0, 256)$, and receives the value Out_1.
- Then, it computes $\mathsf{Out}_2 = f^*(\mathsf{Out}_1, m^{(2)}\|m^{(3)}, a, b)$.
- It returns the value Out_2.

We note that the operations related to the two first bullets allows the adversary to generate $G^*(\mathsf{sk_{Op}}, m, a, b) = \mathsf{Out}_2$. This step is repeated up to a total of q queries from $\mathcal{A}_{G^*}$, with a and b fixed. At some point, $\mathcal{A}_{G^*}$ halts and outputs a guess d of the bit b. The prf-adversary $\mathcal{A}_{f^*}$ chooses its guess b' as $b' = d$ and forwards it to $\mathcal{C}^{\mathsf{prf}}_{f^*}$, which verifies if $b = b'$. We analyze this simulation. Recall that the challenger responded either with a random value (if its internal bit b was set to 0) or with the output of the function $f^*(\mathsf{sk_{Op}}, m^{(1)}\|m^{(3)}, 0, 256)$ (if its internal bit was set as 1). Thus, the output Out_2 matches either the output of a random function or the output of the function $G^*(\mathsf{sk}, m, a, b)$. So the prf-adversary $\mathcal{A}_{f^*}$ simulates perfectly a prf-challenger of G. Thus, we have:

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{prf}}_{f^*}(\mathcal{A}_{f^*}) &= \big|\Pr[\mathcal{A}_{f^*} \to 1 \mid b = 1] - \Pr[\mathcal{A}_{f^*} \to 1 \mid b = 0]\big| \\
&= \big|\Pr[b = b' | b = 1] - \Pr[b = b' | b = 0]\big| \\
&= \big|\Pr[b = d | b = 1] - \Pr[b = d | b = 0]\big| \\
&= \big|\Pr[d' = d | b = 1] - \Pr[d' = d | b = 0]\big| \\
&= \big|\Pr[\mathcal{A}_{G^*} \to 1 \mid b = 1] - \Pr[\mathcal{A}_{G^*} \to 1 \mid b = 0]\big| \\
&= \big|\mathsf{Adv}^{\mathsf{prf}}_{G^*}(\mathcal{A}_{G^*})\big|.
\end{aligned}$$

□

Theorem 7. **[prf-security for G_{tuak}.]** *Let $G_{\mathsf{tuak}} : \{0,1\}^{\kappa} \times \{0,1\}^{e} \times \{0,1\}^{d-e} \times \{0,1\}^{t} \times \{0,1\}^{t} \rightarrow \{0,1\}^{n}$ and $f : \{0,1\}^{\kappa} \times \{0,1\}^{e} \times \{0,1\}^{d} \times \{0,1\}^{t} \times \{0,1\}^{t} \rightarrow \{0,1\}^{m}$ be the two functions specified above. Consider a (t,q)-adversary $\mathcal{A}$ against the* prf*-security of the function G_{tuak}, running in time t and making at most q queries to its challenger. Denote the advantage of this adversary as $\mathsf{Adv}^{\mathsf{prf}}_{G_{\mathsf{tuak}}}(\mathcal{A})$. Then there exist: (i) a $(t' \approx 2 \cdot t, q' = 2 \cdot q)$-adversary $\mathcal{A}$' with an advantage $\mathsf{Adv}^{\mathsf{prf}}_{f}(\mathcal{A}')$ of winning against the pseudorandomness of f, such that:*

$$\mathsf{Adv}^{\mathsf{prf}}_{G_{\mathsf{tuak}}}(\mathcal{A}) = \mathsf{Adv}^{\mathsf{prf}}_{f}(\mathcal{A}').$$

Proof. We construct the adversary $\mathcal{A}_f$ using a prf-adversary $\mathcal{A}_{G_{\mathsf{tuak}}}$ The latter uses $\mathcal{A}_f$ as a challenger for a prf-game $\mathbb{G}_{\mathsf{prf}(f)}$ and can only communicate with $\mathcal{A}_f$ whereas $\mathcal{A}_f$ has access to a challenger for f. To begin with, the challenger $\mathcal{C}^{\mathsf{prf}}_f$ chooses a bit b and a private $\mathsf{sk} \in \{0,1\}^{\kappa}$. If $b = 0$, it assigns f to a random function and if $b = 1$, it assigns f to the specific internal function. The adversary $\mathcal{A}_f$ expects queries of the form (m, a, b) from $\mathcal{A}_{G_{\mathsf{tuak}}}$, with $m = m^{(1)} \| m^{(2)} \| m^{(3)} \in \{0,1\}^{d}$, and $a, b \in \{0,1\}^{t}$. Then, $\mathcal{A}_f$ responds as follows:

- It queries its challenger $\mathcal{C}^{\mathsf{prf}}_f$ on input $(m^{(1)} \| m^{(3)}, 0, 256)$, and receives the value Out_1.
- Then, it queries $\mathcal{C}^{\mathsf{prf}}_f$ for inputs $(\mathsf{Out}_1 \| m^{(2)}, a, b)$ and receives the value Out_2.
- It returns the value Out_2.

We note that the two operations allows the adversary to generate $G_{\mathsf{tuak}}(\mathsf{sk}, m, a, b)$ computing $f(\mathsf{sk},\ f(\mathsf{sk}, m^{(1)} \| m^{(3)}, 0, 256) \| m^{(2)}, a, b)$. This step is repeated up to a total of q queries from $\mathcal{A}_{G_{\mathsf{tuak}}}$, with a and b fixed. At some point, $\mathcal{A}_{G_{\mathsf{tuak}}}$ halts and outputs a guess d of the bit b. The prf-adversary $\mathcal{A}_f$ chooses its guess b' as $b' = d$ and forwards it to $\mathcal{C}^{\mathsf{prf}}_f$, which verifies if $b = b'$. We analyze this simulation. Recall that the challenger responded either with a random value (if its internal bit b was set to 0) or with the output of the function $f(\mathsf{sk}, m^{(1)} \| m^{(3)}, 0, 256)$ (if its internal bit was set as 1). Thus, the output Out_2 matches either the output of a random function or the output of the function $G_{\mathsf{tuak}}(\mathsf{sk}, m, a, b)$. So the prf-adversary $\mathcal{A}_f$ simulates perfectly a prf-challenger of G_{tuak}. Thus, we have:

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{prf}}_{f}(\mathcal{A}_f) &= \big|\Pr[\mathcal{A}_f \rightarrow 1 \mid b = 1] - \Pr[\mathcal{A}_f \rightarrow 1 \mid b = 0]\big| \\
&= \big|\Pr[b = b' | b = 1] - \Pr[b = b' | b = 0]\big| \\
&= \big|\Pr[b = d | b = 1] - \Pr[b = d | b = 0]\big| \\
&= \big|\Pr[d' = d | b = 1] - \Pr[d' = d | b = 0]\big| \\
&= \big|\Pr[\mathcal{A}_{G_{\mathsf{tuak}}} \rightarrow 1 \mid b = 1] - \Pr[\mathcal{A}_{G_{\mathsf{tuak}}} \rightarrow 1 \mid b = 0]\big| \\
&= \big|\mathsf{Adv}^{\mathsf{prf}}_{G_{\mathsf{tuak}}}(\mathcal{A}')\big|.
\end{aligned}$$

□

The security of MILENAGE **algorithms.** In order to prove the prf-security of the MILENAGE algorithms, we assume that the AES permutation is a good pseudorandom function. We note that this set of algorithms has been previously studied by Gilbert [70]. While Gilbert provides an out-of-context security proof for the MILENAGE algorithms [70], showing they operate as a kind of counter mode in deriving key materials (MILENAGE runs AES multiple times as a one block to many blocks expansion function), it is unclear whether the proved indistinguishablilty properties are strictly useful to guarantee the security of the full UMTS-AKA protocol. By contrast, we begin by analyzing the security of the UMTS-AKA scheme, and are able to show that the security

statements hold when it is instantiated both with MILENAGE and with TUAK. We model the AES algorithm by the function f and a keyed version of a classic Davies-Meyer by the function f^*: $f(K,x) = \mathsf{AES}_K(x)$, $f^*(K,x) = K \oplus \mathsf{AES}_x(K)$, with $x \in \{0,1\}^{128}$, $K \in \{0,1\}^{\kappa}$. Contrary to the TUAK algorithms, the MILENAGE algorithms have not the same behaviour. Let the construction G_{mil1} (resp. G^*_{mil1}), the generalization of the functions $\mathcal{F}_1$ and $\mathcal{F}_1^*$ and G_{mil2} (resp. G^*_{mil2}) the generalization of the functions $\mathcal{F}_2$, $\mathcal{F}_3$, $\mathcal{F}_4$, $\mathcal{F}_5$, $\mathcal{F}_5^*$ which are keyed with the subscriber key $\mathsf{sk_C}$ (resp. with the operator key $\mathsf{sk_{Op}}$):

$$
\begin{aligned}
\mathcal{F}_1(\mathsf{sk_{Op}},\mathsf{sk_C},\mathsf{R},\mathsf{Sqn},\mathsf{AMF}) &= G_{\mathsf{mil1}}(\mathsf{sk_C},\mathsf{inp}_1,0,63) = G^*_{\mathsf{mil1}}(\mathsf{sk_{Op}},\mathsf{inp}^*_1,0,63),\\
\mathcal{F}_1^*(\mathsf{sk_{Op}},\mathsf{sk_C},\mathsf{R},\mathsf{Sqn},\mathsf{AMF}) &= G_{\mathsf{mil1}}(\mathsf{sk_C},\mathsf{inp}_2,64,127) = G^*_{\mathsf{mil1}}(\mathsf{sk_{Op}},\mathsf{inp}^*_2,64,127),\\
\mathcal{F}_2(\mathsf{sk_{Op}},\mathsf{sk_C},\mathsf{R}) &= G_{\mathsf{mil2}}(\mathsf{sk_C},\mathsf{inp}_2,64,127) = G^*_{\mathsf{mil2}}(\mathsf{sk_{Op}},\mathsf{inp}^*_2,64,127),\\
\mathcal{F}_3(\mathsf{sk_{Op}},\mathsf{sk_C},\mathsf{R}) &= G_{\mathsf{mil2}}(\mathsf{sk_C},\mathsf{inp}_3,0,127) = G^*_{\mathsf{mil2}}(\mathsf{sk_{Op}},\mathsf{inp}^*_3,0,127),\\
\mathcal{F}_4(\mathsf{sk_{Op}},\mathsf{sk_C},\mathsf{R}) &= G_{\mathsf{mil2}}(\mathsf{sk_C},\mathsf{inp}_4,0,127) = G^*_{\mathsf{mil2}}(\mathsf{sk_{Op}},\mathsf{inp}^*_4,0,127),\\
\mathcal{F}_5(\mathsf{sk_{Op}},\mathsf{sk_C},\mathsf{R}) &= G_{\mathsf{mil2}}(\mathsf{sk_C},\mathsf{inp}_2,0,47) = G^*_{\mathsf{mil2}}(\mathsf{sk_{Op}},\mathsf{inp}^*_2,0,47),\\
\mathcal{F}_5^*(\mathsf{sk_{Op}},\mathsf{sk_C},\mathsf{R}) &= G_{\mathsf{mil2}}(\mathsf{sk_C},\mathsf{inp}_5,0,47) = G^*_{\mathsf{mil2}}(\mathsf{sk_{Op}},\mathsf{inp}^*_5,0,47),
\end{aligned}
$$

with: $\mathsf{inp}_1 = \mathsf{sk_{Op}}\|\mathsf{R}\|(\mathsf{Sqn}\|\mathsf{AMF})\|\mathsf{c}_1\|\mathsf{r}_1\|0^{128}$, $\mathsf{inp}^*_1 = \mathsf{sk_C}\|\mathsf{R}\|(\mathsf{Sqn}\|\mathsf{AMF})\|\mathsf{c}_1\|\mathsf{r}_1\|0^{128}$,
$\forall i \in \{2,...,5\}, \mathsf{inp}_i = \mathsf{sk_{Op}}\|\mathsf{R}\|\mathsf{c}_i\|\mathsf{r}_i\|0^{128}$, $\mathsf{inp}^*_i = \mathsf{sk_C}\|\mathsf{R}\|\mathsf{c}_i\|\mathsf{r}_i\|0^{128}$.

Then, these constructions are constructed as follows:

$$
\begin{aligned}
G_{\mathsf{mil1}}(K,\mathsf{val}^{(1)},a,b) &= \lfloor \mathsf{Top}_C \oplus f(K,\mathsf{val}_4 \oplus f(K,\mathsf{Top}_C \oplus \mathsf{val}_2 \oplus \mathsf{val}_6)\oplus\\
&\qquad \mathsf{Rot}_{\mathsf{val}_5}(\mathsf{Top}_C \oplus (\mathsf{val}_3\|\mathsf{val}_3)))\rfloor_{a..b},\\
G_{\mathsf{mil2}}(K,\mathsf{val}^{(2)},a,b) &= \lfloor \mathsf{Top}_C \oplus f(K,\mathsf{val}_4\oplus\\
&\qquad \mathsf{Rot}_{\mathsf{val}_5}(\mathsf{Top}_C \oplus f(K,\mathsf{Top}_C \oplus \mathsf{val}_2 \oplus \mathsf{val}_6)))\rfloor_{a..b},\\
G^*_{\mathsf{mil1}}(K^*,\mathsf{val}^{*(1)},a,b) &= \lfloor \mathsf{Top}_C \oplus f(\mathsf{val}^*_1,\mathsf{val}^*_4 \oplus f(\mathsf{val}^*_1,\mathsf{Top}_C \oplus \mathsf{val}^*_2 \oplus \mathsf{val}^*_6)\oplus\\
&\qquad \mathsf{Rot}_{\mathsf{val}^*_5}(\mathsf{Top}_C \oplus (\mathsf{val}^*_3\|\mathsf{val}^*_3)))\rfloor_{a..b},\\
G^*_{\mathsf{mil2}}(K^*,\mathsf{val}^{*(2)},a,b) &= \lfloor \mathsf{Top}_C \oplus f(\mathsf{val}^*_1,\mathsf{val}^*_4\oplus\\
&\qquad \mathsf{Rot}_{\mathsf{val}^*_5}(\mathsf{Top}_C \oplus f(\mathsf{val}^*_1,\mathsf{Top}_C \oplus \mathsf{val}^*_2 \oplus \mathsf{val}^*_6)))\rfloor_{a..b},
\end{aligned}
$$

with G_{mil1} (resp. G^*_{mil1}): $\{0,1\}^{\kappa}\times\{0,1\}^{d_1}\times\{0,1\}^t\times\{0,1\}^t \to \{0,1\}^n$, G_{mil2} (resp. G^*_{mil2}): $\{0,1\}^{\kappa} \times \{0,1\}^{d_2} \times \{0,1\}^t \times \{0,1\}^t \to \{0,1\}^n$, $\mathsf{val}^{(1)} = \mathsf{val}_1\|\ \mathsf{val}_2\|\ \mathsf{val}_3\|\ \mathsf{val}_4\|\ \mathsf{val}_5\|\|\mathsf{val}_6$, $\mathsf{val}^{(2)} = \mathsf{val}_1\|\ \mathsf{val}_2\|\ \mathsf{val}_4\|\ \mathsf{val}_5\|\mathsf{val}_6$, val_1,val_2, $\mathsf{val}_4, \mathsf{val}_6 \in \{0,1\}^{128}$,$\mathsf{val}_3 \in \{0,1\}^{64}$, $\mathsf{val}_5 \in \{0,1\}^7$, and $\mathsf{val}^{*(1)} = \mathsf{val}^*_1\|\ \mathsf{val}^*_2\|\ \mathsf{val}^*_3\|\ \mathsf{val}^*_4\|\ \mathsf{val}^*_5\ \|\mathsf{val}^*_6$, $\mathsf{val}^{(2)}_* = \mathsf{val}^*_1\|\ \mathsf{val}^*_2\|\ \mathsf{val}^*_4\|\ \mathsf{val}^*_5\ \|\mathsf{val}^*_6$, val^*_1,val^*_2, $\mathsf{val}^*_4, \mathsf{val}^*_6 \in \{0,1\}^{128}$,$\mathsf{val}^*_3 \in \{0,1\}^{64}$, $\mathsf{val}^*_5 \in \{0,1\}^7$ and $\mathsf{Top}_C = \mathsf{val}_1 \oplus f(K,\mathsf{val}_1) = K^* \oplus f^*(\mathsf{val}^*_1, K^*)$.

We express the security property of the generalizations G_{mil1} and G_{mil2} (resp. G^*_{mil1} and G^*_{mil2}) under the prf-security of the function f (resp. f^*). While we cannot prove the latter property, we can conjecture that the advantage of a prf-adversary would be of the form:

$$\mathsf{Adv}^{\mathsf{prf}}_f(\mathcal{A}) \leq \mathsf{c}_1 \cdot \frac{t/T_f}{2^{128}} + \mathsf{c}_2 \cdot \frac{q^2}{2^{128}},$$

for any adversary $\mathcal{A}$ running in time t and making at most q queries at its challenger. Here, m is the output's size of our function f and T_f is the time to do one f computation on the fixed RAM model of computation. Moreover, c_1 and c_2 are two constants depending only on this model. In other words, we assume that the best attacks are either a exhaustive key search or a linear cryptanalysis. We also conjecture that the advantage of a prf-adversary on f^* is negligible.

Pseudorandomness of MILENAGE algorithms. We begin by reducing the prf-security of G_{mil1} and G_{mil2} to the prf-security of the function f. This implies the prf-security of each MILENAGE

algorithm. In the following, we recall the properties of the functions G_{mil1} and G_{mil2} and prove them.

Theorem 8. **[prf-security of G_{mil1}]** *Let $G_{\mathsf{mil1}} : \{0,1\}^\kappa \times \{0,1\}^d \times \{0,1\}^t \times \{0,1\}^t \to \{0,1\}^n$ and $f : \{0,1\}^\kappa \times \{0,1\}^{128} \times \{0,1\}^{128} \to \{0,1\}^{128}$ be the two functions specified above. Consider a (t,q)-adversary $\mathcal{A}$ against the prf-security of the function G_{mil1}, running in time t and making at most q queries to its challenger. Denote the advantage of this adversary as $\mathsf{Adv}^{\mathsf{prf}}_{G_{\mathsf{mil1}}}(\mathcal{A})$. Then there exists: (i) a $(t' \approx 3 \cdot t, q' = 3 \cdot q)$-adversary $\mathcal{A}$' with an advantage $\mathsf{Adv}^{\mathsf{prf}}_{f}(\mathcal{A}')$ of winning against the pseudorandomness of f such that:*

$$\mathsf{Adv}^{\mathsf{prf}}_{G_{\mathsf{mil1}}}(\mathcal{A}) = \mathsf{Adv}^{\mathsf{prf}}_{f}(\mathcal{A}'),$$

Proof. We construct the adversary $\mathcal{A}_f$ using a prf-adversary $\mathcal{A}_{G_{\mathsf{mil1}}}$. The latter uses $\mathcal{A}_f$ as a challenger for a prf-game $\mathbb{G}_{\mathsf{prf}(f)}$ and can only communicate with $\mathcal{A}_f$ whereas $\mathcal{A}_f$ has access to a challenger for f. To begin with, the challenger $\mathcal{C}^{\mathsf{prf}}_f$ chooses a bit b and a private $\mathsf{sk} \in \{0,1\}^\kappa$. If $b = 0$, it assigns f to a random function and if $b = 1$, it assigns f to the specific internal function. The adversary $\mathcal{A}_f$ waits for queries from $\mathcal{A}_{G_{\mathsf{mil1}}}$ of the form (m, a, b), with $m = m^{(1)}\|m^{(2)}\|m^{(3)}\|m^{(4)}\|m^{(5)} \in \{0,1\}^d$, and $a, b \in \{0,1\}^t$ and responds as follows:

- It queries its challenger $\mathcal{C}^{\mathsf{prf}}_f$ for inputs $m^{(1)}$ and receives the value Out_1.
- Then, it computes $\mathsf{Top}_C = m^{(1)} \oplus \mathsf{Out}_1$ and it queries $\mathcal{C}^{\mathsf{prf}}_f$ for inputs $(\mathsf{Out}_1 \oplus m^{(2)})$ and receives the value Out_2.
- It queries $(m^{(4)} \oplus \mathsf{Out}_2 \oplus rot(\mathsf{Out}_1 \oplus (m^{(3)}\|m^{(3)}), m^{(5)}))$ and receives the value Out_3.
- It returns the value $\lfloor \mathsf{Out}_1 \oplus \mathsf{Out}_3 \rfloor_{a,b}$.

This step is repeated up to a total of q queries from $\mathcal{A}_{G_{\mathsf{mil1}}}$, with a and b fixed. At some point, $\mathcal{A}_{G_{\mathsf{mil1}}}$ halts and outputs a guess d of the bit b. The prf-adversary $\mathcal{A}_f$ chooses its guess b' as $b' = d$ and forwards it to $\mathcal{C}^{\mathsf{prf}}_f$, which verifies if $b = b'$. We analyze this simulation. Recall that the challenger responded either with a random value (if its internal bit b was set to 0) or with the output of the function $f(\mathsf{sk}_{,}., .)$ (if its internal bit was set as 1). Thus, the output Out_3 matches either the output of the output of the function $G_{\mathsf{mil1}}(\mathsf{sk}_{,}m, a, b)$ or a random function (indeed, the combination of two random functions by a boolean addition gives a random function). So the prf-adversary $\mathcal{A}_f$ simulates perfectly a prf-challenger of G_{mil1}. Thus, we have:

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{prf}}_{f}(\mathcal{A}_f) &= \big|\Pr[\mathcal{A}_f \to 1 \mid b = 1] - \Pr[\mathcal{A}_f \to 1 \mid b = 0]\big| \\
&= \big|\Pr[b = b'|b = 1] - \Pr[b = b'|b = 0]\big| \\
&= \big|\Pr[b = d|b = 1] - \Pr[b = d|b = 0]\big| \\
&= \big|\Pr[d' = d|b = 1] - \Pr[d' = d|b = 0]\big| \\
&= \big|\Pr[\mathcal{A}_{G_{\mathsf{mil1}}} \to 1 \mid b = 1] - \Pr[\mathcal{A}_{G_{\mathsf{mil1}}} \to 1 \mid b = 0]\big| \\
&= \mathsf{Adv}^{\mathsf{prf}}_{G_{\mathsf{mil1}}}(\mathcal{A}').
\end{aligned}$$

□

Theorem 9. **[prf-security for G^*_{mil1}]** *Let $G^*_{\mathsf{mil1}} : \{0,1\}^\kappa \times \{0,1\}^d \times \{0,1\}^t \times \{0,1\}^t \to \{0,1\}^n$ and $f^* : \{0,1\}^\kappa \times \{0,1\}^{128} \to \{0,1\}^{128}$ be the two functions specified above. Consider a $(t$, $q)$-adversary $\mathcal{A}$ against the prf-security of the function G^*_{mil1}, running in time t and making at most q queries to its challenger. Denote the advantage of this adversary as $\mathsf{Adv}^{\mathsf{prf}}_{G^*_{\mathsf{mil1}}}(\mathcal{A})$. Then*

there exists a $(t' \approx O(t), q' = q)$*-adversary* $\mathcal{A}$*' with an advantage* $\mathsf{Adv}^{\mathsf{prf}}_{f^*}(\mathcal{A}')$ *of winning against the pseudorandomness of* f^* *such that:*

$$\mathsf{Adv}^{\mathsf{prf}}_{G^*_{\mathsf{mil1}}}(\mathcal{A}) = \mathsf{Adv}^{\mathsf{prf}}_{f^*}(\mathcal{A}').$$

Proof. We construct the adversary $\mathcal{A}_{f^*}$ using a prf-adversary $\mathcal{A}_{G^*_{\mathsf{mil1}}}$. The latter uses $\mathcal{A}_{f^*}$ as a challenger for a prf-game $\mathbb{G}_{\mathsf{prf}(f^*)}$ and can only communicate with $\mathcal{A}_{f^*}$ whereas $\mathcal{A}_{f^*}$ has access to a challenger for f^*. To begin with, the challenger $\mathcal{C}^{\mathsf{prf}}_{f^*}$ chooses a bit b and a private $\mathsf{sk} \in \{0,1\}^\kappa$. If $b = 0$, it assigns f^* to a random function and if $b = 1$, it assigns f^* to the specific internal function. The adversary $\mathcal{A}_{f^*}$ waits for queries from $\mathcal{A}_{G^*_{\mathsf{mil1}}}$ of the form (m, a, b), with $m^* = m^{(1)} \| \ m^{(2)} \| \ m^{(3)} \| \ m^{(4)} \| \ m^{(5)} \in \{0,1\}^d$, and $a, b \in \{0,1\}^t$ and responds as follows:

- It queries its challenger $\mathcal{C}^{\mathsf{prf}}_{f^*}$ for inputs $m^{(1)}$ and receives the value $\mathsf{Top}_C = \mathsf{Out}_1$.
- It computes $\mathsf{Out}_3 = \mathsf{Out}_1 \oplus f(m^*_1, m^*_4 \oplus f(m^*_1, \mathsf{Out}_1 \oplus m^*_2) \oplus \mathsf{Rot}_{m^*_5}(\mathsf{Out}_1 \oplus (m^*_3 \| m^*_3)))$.
- It returns the value $\lfloor \mathsf{Out}_3 \rfloor_{a,b}$.

This step is repeated up to a total of q queries from $\mathcal{A}_{G_{\mathsf{mil1}}}$, with a and b fixed. At some point, $\mathcal{A}_{G^*_{\mathsf{mil1}}}$ halts and outputs a guess d of the bit b. The prf-adversary $\mathcal{A}_{f^*}$ chooses its guess b' as $b' = d$ and forwards it to $\mathcal{C}^{\mathsf{prf}}_{f^*}$, which verifies if $b = b'$. We analyze this simulation. Recall that the challenger responded either with a random value (if its internal bit b was set to 0) or with the output of the function $f^*(\mathsf{sk}, .)$ (if its internal bit was set as 1). Thus, the output Out_3 matches either the output of the function $G^*_{\mathsf{mil1}}(\mathsf{sk}_{\mathsf{Op}}, m, a, b)$ or a random function (indeed, the combination of two random functions by a boolean addition gives a random function). So the prf-adversary $\mathcal{A}_{f^*}$ simulates perfectly a prf-challenger of G^*_{mil1}. Thus, we have:

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{prf}}_{f^*}(\mathcal{A}_{f^*}) &= \big|\Pr[\mathcal{A}_{f^*} \to 1 \mid b = 1] - \Pr[\mathcal{A}_{f^*} \to 1 \mid b = 0]\big| \\
&= \big|\Pr[b = b' | b = 1] - \Pr[b = b' | b = 0]\big| \\
&= \big|\Pr[b = d | b = 1] - \Pr[b = d | b = 0]\big| \\
&= \big|\Pr[d' = d | b = 1] - \Pr[d' = d | b = 0]\big| \\
&= \big|\Pr[\mathcal{A}_{G^*_{\mathsf{mil1}}} \to 1 \mid b = 1] - \Pr[\mathcal{A}_{G^*_{\mathsf{mil1}}} \to 1 \mid b = 0]\big| \\
&= \mathsf{Adv}^{\mathsf{prf}}_{G^*_{\mathsf{mil1}}}(\mathcal{A}').
\end{aligned}$$

□

We have the same reduction for the function G_{mil2} and G^*_{mil2} under the same parameters which can be proved in the same way. Thus, it holds that:

Theorem 10. **[prf-security of** G_{mil2}**]** *Let* $G_{\mathsf{mil2}} : \{0,1\}^\kappa \times \{0,1\}^d \times \{0,1\}^t \times \{0,1\}^t \to \{0,1\}^n$ *and* $f : \{0,1\}^\kappa \times \{0,1\}^{128} \times \{0,1\}^{128} \to \{0,1\}^{128}$ *be the two functions specified above. Consider a* (t, q)*-adversary* $\mathcal{A}$ *against the* prf*-security of the function* G_{mil2}*, running in time* t *and making at most* q *queries to its challenger. Denote the advantage of this adversary as* $\mathsf{Adv}^{\mathsf{prf}}_{G_{\mathsf{mil2}}}(\mathcal{A})$*. Then there exists: a* $(t' \approx 3 \cdot t, q' = 3 \cdot q)$*-adversary* $\mathcal{A}$*' with an advantage* $\mathsf{Adv}^{\mathsf{prf}}_{f}(\mathcal{A}')$ *of winning against the pseudorandomness of* f *such that:*

$$\mathsf{Adv}^{\mathsf{prf}}_{G_{\mathsf{mil2}}}(\mathcal{A}) = \mathsf{Adv}^{\mathsf{prf}}_{f}(\mathcal{A}').$$

Theorem 11. **[prf-security for G^*_{mil2}]** *Let $G^*_{\mathsf{mil2}} : \{0,1\}^\kappa \times \{0,1\}^d \times \{0,1\}^t \times \{0,1\}^t \to \{0,1\}^n$ and $f^* : \{0,1\}^\kappa \times \{0,1\}^{128} \to \{0,1\}^{128}$ be the two functions specified above. Consider a (t, q)-adversary $\mathcal{A}$ against the* prf*-security of the function G^*_{mil2}, running in time t and making at most q queries to its challenger. Denote the advantage of this adversary as $\mathsf{Adv}^{\mathsf{prf}}_{G^*_{\mathsf{mil2}}}(\mathcal{A})$. Then there exists a $(t' \approx O(t), q' = q)$-adversary $\mathcal{A}$' with an advantage $\mathsf{Adv}^{\mathsf{prf}}_{f^*}(\mathcal{A}')$ of winning against the pseudorandomness of f^* such that:*

$$\mathsf{Adv}^{\mathsf{prf}}_{G^*_{\mathsf{mil2}}}(\mathcal{A}) = \mathsf{Adv}^{\mathsf{prf}}_{f^*}(\mathcal{A}').$$

3.4 Does EPS-AKA Protocol Guarantee more Security than UMTS-AKA Protocol?

In the GSM and UMTS architecture, the session-key management is simple: the derived session keys, from running AKA protocol, are used one for encryption (CK) and one for integrity (IK) (just the cipher key CK for GSM architectures). These keys are directly used to protect the radio link. In the EPS architecture, the key hierarchy is considerably more elaborate. A local master key $\mathsf{K_{asme}}$ at the core network level is computed by both the client and operator. As 3G USIM is still in use, and 3G and 4G networks are interoperable, the EPS key is computed from CK and IK. Then, the new requirement of cryptographic network separation as defined in [8], implies the computation of the session key by the serving network, using its identity. Moreover, EPS allows for the generation of a batch of session keys, which are desired to separately handle the NAS, RRC and UP protocols. Finally, another desirable effect of such a construction is the key $\mathsf{K_{asme}}$ is not directly used to protect the radio link (contrary to the keys CK and IK in the 2G and 3G networks). That permits to increase the lifetime of the derived key and to avoid frequent updates the derived key as often as in the 3G and 4G networks.

Communications over 4G networks aim to guarantee similar security properties to those in 3G networks. The EPS-AKA protocol essentially aims to guarantee both server- and client-impersonation, and the key-indistinguishability of the session key $\mathsf{K_{asme}}$. In addition, state-confidentiality and soundness must hold in the presence of malicious servers. These properties are formalized in the security model defined in Section 3.2.

Security analysis: We note that the new key management featured by EPS-AKA protocol do not affect client- and server-impersonation resistance, nor state-confidentiality or soundness. As a consequence the same bounds hold for these properties.

Theorem 12. **[S.C.Imp-resistance]** *For the protocol Π using the unitary function G specified in Section 3.3.1, for any $(t, q_{\mathsf{exec}}, q_{\mathsf{res}}, q_{\mathsf{s}}, q_{\mathsf{Op}}, q_G)$-adversary $\mathcal{A}$ against the* S.C.Imp*-security of Π, winning with advantage $\mathsf{Adv}^{\mathsf{S.C.Imp}}_{\Pi}(\mathcal{A})$, there exists a $(t' \approx O(t), q' = 5 \cdot q_{\mathsf{s}} \cdot q_{\mathsf{Op}} + q_G + q_{\mathsf{exec}}(q_{\mathsf{res}} + 2))$-adversary $\mathcal{A}'$ against the pseudorandomness of G such that:*

$$\mathsf{Adv}^{\mathsf{S.C.Imp}}_{\Pi}(\mathcal{A}) \leq \mathsf{n_C} \cdot \left(2 \cdot \mathsf{Adv}^{\mathsf{prf}}_{G}(\mathcal{A}') + \frac{(q_{\mathsf{exec}} + q_{\mathsf{s}} \cdot q_{\mathsf{Op}})^2}{2^{|\mathsf{R}|}} + \frac{q_{\mathsf{exec}} \cdot q_{\mathsf{res}}}{2^{|\mathsf{Res}|}} + \frac{1}{2^\kappa}\right).$$

Theorem 13. **[W.S.Imp-resistance]** *For the protocol Π using the unitary function G specified in Section 3.3.1, for any $(t, q_{\mathsf{exec}}, q_{\mathsf{res}}, q_{\mathsf{G}})$-adversary $\mathcal{A}$ against the* W.S.Imp*-security of Π, winning with advantage $\mathsf{Adv}^{\mathsf{W.S.Imp}}_{\Pi}(\mathcal{A})$, there exists a $(t' \approx t, q = q_{\mathsf{exec}} \cdot (q_{\mathsf{res}} + 2) + q_{\mathsf{G}})$-adversary $\mathcal{A}'$ against the pseudorandomness of G such that:*

$$\mathsf{Adv}^{\mathsf{W.S.Imp}}_{\Pi}(\mathcal{A}) \leq \mathsf{n_C} \cdot \left(\mathsf{Adv}^{\mathsf{prf}}_{G}(\mathcal{A}') + \frac{q_{\mathsf{exec}} \cdot q_{\mathsf{res}}}{2^{|\mathsf{Mac_S}|}} + \frac{1}{2^\kappa}\right).$$

Theorem 14. **[Sound-resistance]** *For the protocol* Π *using the unitary function* G *specified in Section 3.3.1, for any* $(t, q_{\mathsf{exec}}, q_{\mathsf{res}}, q_{\mathsf{Op}}, q_{\mathsf{G}}, \epsilon)$*-adversary* $\mathcal{A}$ *against the soundness of* Π*, winning with advantage* $\mathsf{Adv}^{\mathsf{Sound}}_{\Pi}(\mathcal{A})$*, there exists a* $(t' \approx t, q' = 5 \cdot q_{\mathsf{Op}} + q_G + \mathsf{n_C} \cdot q_{\mathsf{exec}}(2 + q_{\mathsf{res}}))$*-adversary* $\mathcal{A}'$ *against the pseudorandomness of* G *such that:*

$$\mathsf{Adv}^{\mathsf{Sound}}_{\Pi}(\mathcal{A}) \leq \mathsf{n_C} \cdot \left(2 \cdot \mathsf{Adv}^{\mathsf{prf}}_{G}(\mathcal{A}') + \frac{q_{\mathsf{exec}} \cdot q_{\mathsf{res}}}{2^{|\mathsf{Mac_S}|}} + \frac{1}{2^{\kappa}} \right).$$

Theorem 15. **[St.Conf-resistance]** *For the protocol* Π *using the unitary functions* G, G^* *specified in Section 3.3.1, for any* $(t, q_{\mathsf{exec}}, q_{\mathsf{res}}, q_{\mathsf{Op}}, q_G, q_{G^*})$*-adversary* $\mathcal{A}$ *against the* St.Conf*-security of* Π*, winning with advantage* $\mathsf{Adv}^{\mathsf{St.Conf}}_{\Pi}(\mathcal{A})$*, there exist: a* $(t' \approx O(t), q' = q_G + q_{\mathsf{exec}}(5 + q_{\mathsf{res}}))$*-prf-adversary* $\mathcal{A}_1$ *on* G *and* $(t' \approx O(t), q' = q_{G^*})$*-prf-adversary* $\mathcal{A}_2$ *on* G^* *such that:*

$$\mathsf{Adv}^{\mathsf{St.Conf}}_{\Pi}(\mathcal{A}) \leq \mathsf{n_C} \cdot \left(\frac{1}{2^{|\mathsf{sk_C}|}} + \frac{1}{2^{|\mathsf{sk_{Op}}|}} + \frac{2}{2^{|\mathsf{Sqn}|}} + \mathsf{Adv}^{\mathsf{prf}}_{G}(\mathcal{A}_1) + \mathsf{Adv}^{\mathsf{prf}}_{G^*}(\mathcal{A}_2) \right).$$

Additional security offered by EPS-AKA: Since all the EPS-AKA session keys are based on a new key $\mathsf{K_{asme}}$ featured in servers, EPS-AKA can guarantee a better security in terms of session key indistinguishability than UMTS-AKA used the 3G networks but still owns the lack of the strong server-impersonation.

As detailed in the description of the EPS-AKA protocol, the authentication vectors are generated by the home network and sent to the serving network MMEs. Each of these networks is managed differently, in particular with respect to the security and privacy guarantees they can offer for sensitive data. Zhang and Fang [**?**] pointed out that the possiblity of corrupting servers (belonging especially to a server network distinct from the home network) notably implies an active attack permitting to impersonate an uncorrupted server. When a server is corrupted, it can request some authentication vectors (using an auth. vectors request) wherever the mobile client is located. From a corrupted (or malicious) server, a False Base Station (FBS) attack can be mounted against legitimate mobile clients located in a uncorrupted network. Indeed, the authentication challenge is not characteristic of the current location of the mobile client.

Such an FBS can reuse the obtained vectors to impersonate any (fresh) server. Thus, the classic AKA protocol cannot guarantee strong server-impersonation since in that model servers can be corrupted. Contrary to the UMTS-AKA used in the 3G architecture, corruption does not, however, imply an active attack against strong key-indistinguishability. As proved before, the new key hierarchy permits to guarantee this property since the main session key is computed using the unique identifier of the user. Thus, two servers using the same authentication vector request, i.e. using the same R, Sqn, would not receive the same key $\mathsf{K_{asme}}$ given to the MME. Moreover, the two pre-session keys CK and IK are never given to the MME. We note that if these keys were included in the authentication vector (as for 3G networks) the strong-key-indistinguishability property would not hold.

Thus, we can propose a security proof of the strong key-indistinguishability property as follows:

Theorem 16. **[S.K.Ind-resistance]** *For the protocol* Π *using the unitary function* G *specified in Section 3.3.1, for any* $(t, q_{\mathsf{exec}}, q_{\mathsf{res}}, q_{\mathsf{Op}}, q_{\mathsf{S}}, q_G)$*-adversary* $\mathcal{A}$ *against the* S.K.Ind*-security of* Π *winning with advantage* $\mathsf{Adv}^{\mathsf{S.K.Ind}}_{\Pi}(\mathcal{A})$ *there exists* $(t' \approx O(t), q' = 5 \cdot q_{\mathsf{Op}} \cdot q_{\mathsf{S}} + q_G + q_{\mathsf{exec}}(2 + q_{\mathsf{res}}))$*-adversary* $\mathcal{A}'$ *and* $(t'' \approx O(t), q'' = q_{\mathsf{KDF}} + q_{\mathsf{exec}})$*-adversary* $\mathcal{A}''$ *against respectively the pseudorandomness of* G *and the* prf*-security of* KDF*with:*

$$\mathsf{Adv}^{\mathsf{S.K.Ind}}_{\Pi}(\mathcal{A}) \leq \mathsf{n_C} \cdot \mathsf{n}_{\mathcal{S}} \left(\frac{(q_{\mathsf{exec}} + q_{\mathsf{Op}} \cdot q_{\mathsf{S}})^2}{2^{|\mathsf{R}|}} + 2 \cdot \mathsf{Adv}^{\mathsf{prf}}_{G}(\mathcal{A}') + 2 \cdot \mathsf{Adv}^{\mathsf{kdf}}_{\mathsf{KDF}}(\mathcal{A}'') + \frac{q^2_{\mathsf{exec}}}{2^{|\mathsf{Sqn}|}} \right).$$

Proof. We use the following game hops:

Game $\mathbb{G}_0$: The adversary in this game is a classic active MitM between the client and the server, whose goal is to distinguish the session key $\mathsf{K_{asme}}$ of a fresh client instance P_i from a random key of the same length. Contrary to the weak-key-indistinguishability property, we allow server corruptions.

Game $\mathbb{G}_2$: In this double reduction ($\mathbb{G}_0 \Rightarrow \mathbb{G}_2$), we use the same argument as applied in the proof of the weak-key-indistinguishability property detailed before. At this step, this game is reduced to the original game played with a single client.

Game $\mathbb{G}_3$: In this game, we abort the game if two honest server instances (or server-corruption/ operator access) output the same random value R of the token/challenge. The games $\mathbb{G}_3$ and $\mathbb{G}_4$ are equivalent up to a collision term $\frac{(q_{\mathsf{exec}}+q_{\mathsf{Op}}\cdot q_{\mathsf{S}})^2}{2^{\{\mathsf{R}\}}}$.

Game $\mathbb{G}_4$: In this game $\mathbb{G}_4$, we restrict the adversary to interact with only one server $\mathcal{S}$. With such a restriction, the adversary loses the ability to corrupt server $\mathcal{S}^* \neq \mathcal{S}$ and operator access. As a consequence the adversary loses its access to legitimately-obtained authentication vectors stored by $\mathcal{S}^*$, which can be used to break the server-impersonation resistance of the protocol. In addition, the server $\mathcal{S}^*$ will also store, apart from the authentication vectors, other values such as CK, IK, which could potentially leak information on $\mathsf{sk_C}$ or $\mathsf{sk_{Op}}$. We show, in fact that the adversary gains nothing by accessing the state of servers $\mathcal{S}^* \neq \mathcal{S}$. For the second condition, the security loss is given by the pseudorandomness of the functions G and KDF. For the first point, we recall that these authentication vectors are composed by the following values: two MAC values $\mathsf{Mac_S}$ and $\mathsf{Mac_C}$, a random value R, a constant AMF, the masked sequence number $\mathsf{Sqn} \oplus \mathsf{AK}$ and the session key $\mathsf{K_{asme}}$. The last value is the only one value computed from a server identifier. Thus, depending the network where the authentication challenge is used, the related session key $\mathsf{K_{asme}}$ computed will be not the same since we consider each server owns a different identifier. So, replaying this vector permits only to impersonate a server $\mathcal{S}$ (the mac value is not featured by the local network) and not to obtain the correct client' session key $\mathsf{K_{asme}}$ per a fresh session. The security loss is given by the collision probability between two outputs of the KDF functions taking two different inputs, i.e the pseudorandomnes of the KDF function.

Game $\mathbb{G}_5$: In this game $\mathbb{G}_5$, we replace all the outputs of the functions G and KDF by truly and consistent (i.e independent of the input but the same input implies the same output) values. That restrict notably the adversary to obtain some information about the long-term keys sk and $\mathsf{sk_{Op}}$ from some couples (input, output). We argue that the security loss is exactly the pseudorandomness of the function G and KDF.

Game $\mathbb{G}_6$: In this game $\mathbb{G}_6$, we consider that the outputs of the function G are unique. That implied that two keys CK and IK are indistinguishable to real random values and these values are unique for each session. We recall that the two keys are computed from the function G taking at input some constants and a unique no-replayable random value R. Moreover, these keys cannot be obtained directly to the adversary without corrupt the related entity (client and operator) since the servers cannot computed and received them, and the output of the key derivation function is randomized by the previous reduction. Thus, the security loss is pseudorandomness of the function G.

Game $\mathbb{G}_7$: In the game $\mathbb{G}_7$, we consider that the outputs of the KDF function are unique. Forethat, we consider that all the inputs of the KDF function permitting to compute the session

key $\mathsf{K_{asme}}$ are unique (different in each server instances) and the pseudorandomness of the function KDF. We recall that the session key is computed from the three main values: a predictible unique server identifier $\mathsf{Id_S}$, the masked value of the sequence number $\mathsf{Sqn} \oplus \mathsf{AK}$. Thus, the unicity of the input depends to the value of the masked sequence number and the network identifier. Since the random value R cannot be repeated during several instances and two challenges cannot be based on the same sequence number (the operator increments its sequence number after each generation of a challenge), the input is not the same in two different session except if we obtain a collision in the computation of the value $\mathsf{Sqn} \oplus \mathsf{AK}$. So the security loss is given by the kdf-security of the key derivation function KDF and the probability to have a collision in the computation of the obfuscated sequence number with the tested session.

Winning $\mathbb{G}_7$: At this point, the adversary plays a game in the presence of a single server, and a single client. The goal of this adversary is to distinguish the real random key to a fresh session key, considering that all the functions used in a session returns truly random. The outputs of the function KDF is indistinguishable to real random values since all the outputs are unique and randomized by the previous reductions. Thus, the adversary can do no better than guessing the answer.

□

Moreover, since the key $\mathsf{K_{asme}}$ is guaranteed to be indistinguishable from random since we assure the use of a secure KDF function. All the session keys of the EPS-AKA can be proved secure as the $\mathsf{K_{asme}}$ using the same reductions because the same function KDF (with different predictable input) is used to derive all the session keys, as described in Section 2.3.2.

3.5 Vulnerabilities of the AKA Protocols

Before our study, few papers gave a security analysis of the AKA protocols, especially when instantiated with MILENAGE. The closest results to a security proof use automated (formal) verification. While this approach has many advantages, and an automated proof is a good first step towards a thorough security analysis, one important disadvantage is that automated verification does not give an exact reduction to the security of the underlying primitives; thus, the proof statement is not easy to quantify, making it hard to approximate the tightness of the proof and the size of the parameters. In this sense, our results are stronger.

Some papers clearly focus on the mutual authentication and key-exchange properties of AKA. A first one [109] points out some problems with using sequence numbers and outlines a corrupted-network redirection attack, but does not attempt a security proof. Note that the AKA protocols are *à priori* designed with the assumption that the operator trusts the network; the network attack described by [109] falls outside the scope of our study as we do not consider how networks and operators implement the protocol. Furthermore, note that adding a simple network-specific constant in the computation of the MAC algorithms should prevent such attacks. By removing the sequence number, the same authors propose a stateless variant called AP-AKA, with a security proof based on Shoup's formal model [99].

A second paper [30] refers to the authentication and key security of AKA, but focuses mainly on client privacy. They attempt to do an automated verification proof for the UMTS-AKA protocol, using ProVerif [53]; however, they are only able to assess a *modified* version, which randomizes the sequence number. Since this modification is fundamental, their results cannot be applied to the original protocol. In order to better model the true sequence number updates in the protocol, we

have used an extension of ProVerif called StatVerif [32], which was proposed by Arapinis et al. to handle automatic verification for protocols with global state.

Tsay and Mjølsnes have provided a computational analysis of AKA protocols in [88, 102], using the computational prover CryptoVerif [52]. Additionally to prove the intended authentication and secrecy properties, they provided a new client-impersonation attack considering a malicious client adversary; however, this attack cannot be considered as a practical attack and it is out of the scope of our computational model, since they did not consider the session identifiers which make impossible the session mixup. These identifiers are essential to consider the real playing of successive sessions.

Now, we detail why the UMTS-AKA protocol cannot guarantee the strong key-indistuishability and strong server-impersonation properties.

3.5.1 The Lack of some Requirements

In the three-party mobile setting, the server is authenticated by the client if it presents credentials (authentication vectors) generated by the client's operator. The properties of state-confidentiality and soundness, which the AKA protocols guarantee, indicate that servers cannot learn the client's long-term data, and that they cannot authenticate without the operator-generated data.

However, Zhang [109] and Zhang and Fang [**?**] pointed out that once a server is corrupted, it can obtain legitimate authentication data from the client's operator, and then use this data to set up a False Base Station (FBS), which can lead to a malicious, unauthorised server authenticating to the client. As a result, the UMTS-AKA protocol does not guarantee strong key-indistinguishability, nor strong server-impersonation resistance. Such an attack only implies the lack of strong-server-impersonation resistance for the EPS-AKA protocol.

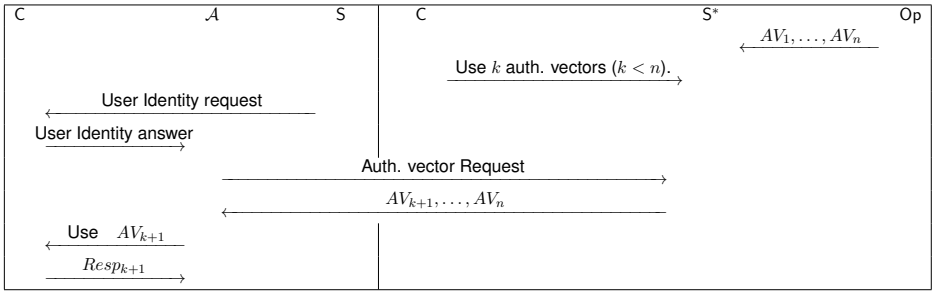


Figure 3.2: The attack of Zhang and Fang. On the right hand side, the client is in the vulnerable network, interacting with the server S^*. The server uses up authentication vectors $\mathsf{AV}_1, \ldots \mathsf{AK}_k$. Then, the server S^* is corrupted, and the adversary $\mathcal{A}$ learns $\mathsf{AV}_{k+1}, \ldots, \mathsf{AV}_n$, which it uses in a second attack phase (on the left).

The main attack strategy is also depicted in Figure 3.2. In a first step, the client C is assumed to be in the area (as designated by the LAI) corresponding to a server S^*, which will later be corrupted. The server receives a batch of authentication vectors ($\mathsf{AV}_1, \ldots, \mathsf{AV}_n$), using some of them (vectors $\mathsf{AV}_1, \ldots, \mathsf{AV}_k$) to provide service to that client (and learn what services this client has been provided with, etc.). Subsequently, the client moves to a different LAI, outside the corrupted network's area. The adversary $\mathcal{A}$ has corrupted the server S^* and learned the remaining vectors $\mathsf{AV}_{k+1}, \ldots, \mathsf{AV}_n$; this adversary then uses this authentication data to authenticate to the client, *in its new location*. This immediately breaks the server-impersonation guarantee. Moreover, since authentication vectors also contain the short-term session keys, key-indistinguishability is breached, too. This attack is particularly dangerous since a single server corruption can affect a very large number of clients. Moreover, server corruption is easily practiced in totalitarian regimes,

in which mobile providers are subject to the state, and partial data is furthermore likely to be leaked upon using backdoored algorithms.

Such attack do not, however, affect client-impersonation resistance, since the server cannot use an authentication vector from the server to respond to a freshly-generated authentication challenge (the random value for the two authentication vectors is different).

The authentication vectors considered in the previous attack, are only useful either if the current authentication challenge is based on the same random value as one of the recovered authentication vectors (that implies the MAC value Mac_C and the session keys CK and IK are the same) or if the cryptographic outputs reveal information about long-term secret data (i.e.the long-term keys sk_C and $\mathsf{sk}_{\mathsf{Op}}$). Such vectors could be dangerous if they were obtained from passive attacks, for example by eavesdropping on the core network communication between the server and the operator. We consider such an attack impossible (practical expensive) even if in the roaming case it is hard to consider this channel secure.

In addition, we denote that a such active attack can permit to desynchronize any mobile client as many as it has fresh authentication challenge. We note that desynchronization between client's and operator' states which implies a resynchronization procedure, is problematic only for the user privacy we will detail in the next chapter. Moreover, since the limited number of the 48-bit sequence number, this shortens the lifetime of the user sequence number.

In conclusion, it is important to propose counter measures to this attack (for example adding a public value characterizing the serving network) to such attacks, since a single corrupted network may jeopardize the entire system.

Security issue with the migration of authentication vectors. In the AKA protocols, the server requires a batch of authentication vectors (and not only one). Using these authentication vectors, a sever can execute an accepted session of the AKA protocols without requiring assistance from the home network. Even if this procedure is allowed in 3GPP specifications, that represents a failure in our notion of soundness. We require that servers never be able to authenticate without the operator's permission. Thus, we propose either to remove this procedure or at least to restrict it to only exchanging the temporary identifier. In particular, we require the exchange of only one authentication vector (i.e. the restriction of the batch to one element) during the AKA protocols.

Operational difficulties with sequence number. The AKA protocols are stateful, i.e. the subscriber and its home network maintain a sequence number, which is updated after each execution of this protocol. Since the sequence number is updated in the network even if the session of the protocol is aborted without specific reasons, an adversary can desynchronize the home network by dropping some authentication challenges. This reduces the lifetime of the sequence number.

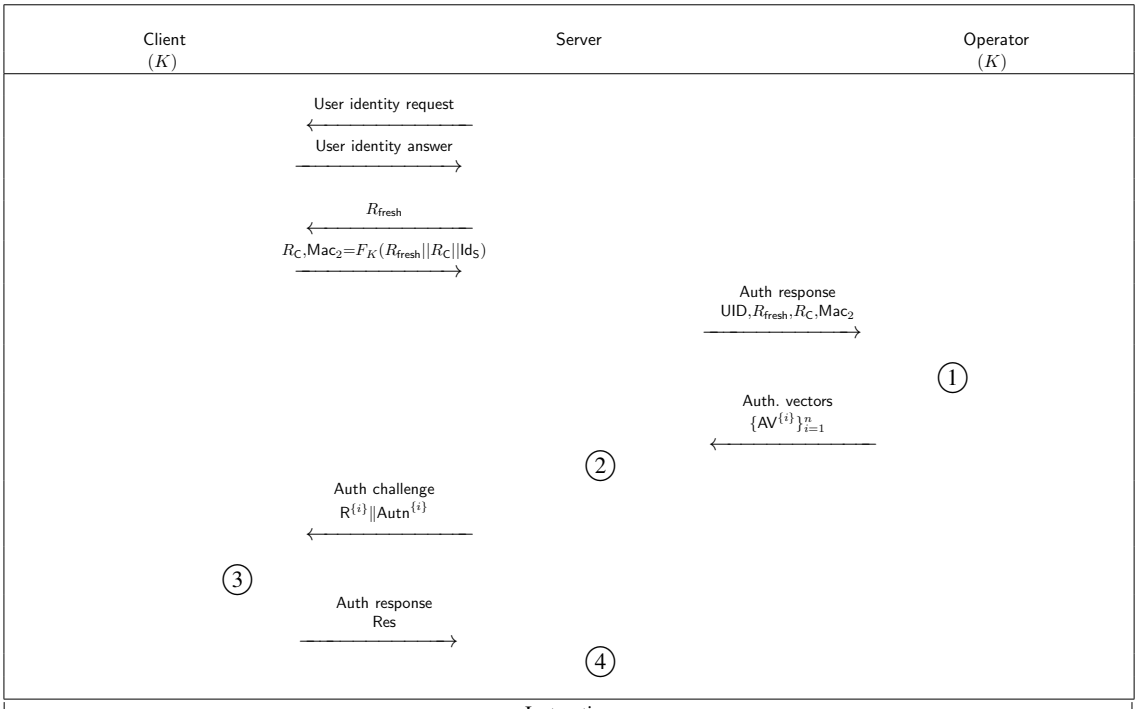
Moreover, when the client checks the freshness of the received sequence number, it does not only verify its equality to its own sequence number is equal to the received one, but only whether it is in the correct range with respect to a static value Δ defined differently by each operator. Thus, the ability to guess the value of a sequence number is unegligible ($\Delta/2^{48}$) since either the size of the sequence number is not increased or the verification is not restricted to the equality.

In order to fix all of these weaknesses, a variant has been proposed in [109]. We analyse this variant and explain why it (still) does not offer sufficient security and efficiency.

3.5.2 Why AP-AKA cannot replace UMTS-AKA?

In 2003, Zhang proposed a variant of UMTS-AKA he called AP-AKA, which we depict in Figure 3.3 (although we use a syntax closer to our own variant, to facilitate a comparison). Instead of the suite of seven cryptographic algorithms specified for the UMTS-AKA protocol, Zhang only uses three independent functions F, G, H, which are all keyed with a key K. The authors do not specify what this key is in the UMTS-AKA scenario, but considering the design of this protocol,

it must be a function of the two keys $\mathsf{sk_C}$ and $\mathsf{sk_{Op}}$. We assume $K = \mathsf{sk_C}||\mathsf{sk_{Op}}$ in this case. For the security of the protocol, G must be a pseudorandom function (PRF), while F and H must be unforgeable MACs.



Instructions:

Client	Server	Operator
③: Recover $\mathsf{R}^{\{i\}}$, and $\mathsf{Autn}^{\{i\}}$. If $R_{\mathsf{C}}^{\{i\}}$ was used before by S: abort. Else: Find $1 \leq i \leq n$ s.t. $R_{\mathsf{C}}^{\{i\}} = H_K(i\|R_{\mathsf{C}})$ If no such i exists: abort; Else: check $\mathsf{Mac_S}$ for $i, \mathsf{R}^{\{i\}}, R_{\mathsf{C}}^{\{i\}}$ (or abort) Compute $\mathsf{Res} = F_K(\mathsf{R}^{\{i\}})$ Compute: $(\mathsf{CK}\|\mathsf{IK}) = G_K(\mathsf{R}^{\{i\}})$.	②: Store $\{\mathsf{AV}^{\{i\}}\}_{i=1}^{n}$. Choose next $\mathsf{AV}^{\{i\}}$, send related challenge ④: $\mathsf{Res} \stackrel{?}{=} \mathsf{Mac_C}$.	①: Check Mac_2 w.r.t. $R_{\mathsf{fresh}}, R_{\mathsf{C}}$. If failure, send reject notice to S and abort; Else: for $i = 1, \ldots, n$, compute: $\mathsf{Mac_C} = F_K(\mathsf{R}^{\{i\}})$ $(\mathsf{CK}\|\mathsf{IK}) = G_K(\mathsf{R}^{\{i\}})$ $R_{\mathsf{C}}^{\{i\}} = H_K(i\|R_{\mathsf{C}})$ $\mathsf{Mac_S} = F_K(\mathsf{R}^{\{i\}}\|i\|R_{\mathsf{C}}^{\{i\}})$ $\mathsf{Autn}^{\{i\}} \leftarrow i\|R_{\mathsf{C}}^{\{i\}}\|\mathsf{Mac_S}$. $\mathsf{AV}^{\{i\}} := (\mathsf{R}^{\{i\}}, \mathsf{CK}, \mathsf{IK}, \mathsf{Autn}^{\{i\}}, \mathsf{Mac_C})$. End For.

Figure 3.3: The AP-AKA Variant.

We first describe this protocol. The procedure begins by the same identification phase as the regular UMTS-AKA procedure shown in Section 2.3.1. Namely, the server sends an identification request, to which the client responds with either the permanent identifier IMSI or with a tuple consisting of the temporary identifier TMSI and the local area identifier LAI of the server that issued the TMSI.

The first modification made with respect to the classical UMTS-AKA is extending the authentication vector request phase, which takes place between two parties in the original scheme, to three parties. The server/client communication takes place across an *insecure channel*, whereas the channel between the server and the operator is *secure*. Zhang [109] adds a message-exchange to the protocol every time the server needs fresh authentication vectors. This exchange is a typical challenge-response authentication: the server sends a fresh nonce R_{fresh}, and the client generates a fresh R_{C}, computing a MAC (namely the function F) keyed with the key K, on input the concatenation of R_{fresh}, R_{C}, and a unique server identifier $\mathsf{Id_S}$. The authentication vectors are similar

to those in the original UMTS-AKA, but they implicitly rely on the client's random value R_{C} and on fresh randomness $\mathsf{R}^{\{i\}}$ generated by the operator for $i = 1, \ldots, n$ (here n is the batch size). An initial step is to generate numbers $R_{\mathsf{C}}^{\{i\}}$ for $i = 1, \ldots, n$; these are generated by using the MAC function H on input i and R_{C}. The server authentication string $\mathsf{Mac}_{\mathsf{S}}$ is computed as the function F on input the operator's randomness $\mathsf{R}^{\{i\}}$, the current index i, and a nonce $R_{\mathsf{C}}^{\{i\}}$ generated from R_{C}. The values $i, R_{\mathsf{C}}^{\{i\}}$, and $\mathsf{Mac}_{\mathsf{S}}$ are grouped as $\mathsf{Autn}^{\{i\}}$ for each i. Each authentication vector $\mathsf{AV}^{\{i\}}$ consists of: the randomness $\mathsf{R}^{\{i\}}$, the authentication string $\mathsf{Autn}^{\{i\}}$, the session keys $(\mathsf{CK}, \mathsf{IK})$ which are derived as a result of the PRF G, keyed with K, on input the randomness $\mathsf{R}^{\{i\}}$, and the expected client-authentication string $\mathsf{Mac}_{\mathsf{C}}$. Finally, a batch of n authentication vectors are sent to the server.

The remainder of the protocol proceeds analogously to the original UMTS-AKA procedure, with the modifications imposed by the way the authentication vectors are generated. In particular, upon receiving the randomness $\mathsf{R}^{\{i\}}$ and the authentication string $\mathsf{Autn}^{\{i\}}$, the client verifies, in order: (1) that it has never received the same string $R_{\mathsf{C}}^{\{i\}}$; (2) that this value is consistent with the randomness R_{C}; (3) that the authentication $\mathsf{Mac}_{\mathsf{S}}$ is correct with respect to this randomness. If any of these verifications fail, the client aborts. Else, it computes the session keys and its own authentication string Res, sending the latter to the server.

Stateful vs. Stateless. Zhang presented his variant as eliminating "dynamic states". We note that, while his work ensured that no sequence number is necessary, the protocol is not entirely stateless. In particular, the client must keep track of a list, which is dynamically updated, of already seen randomness $R_{\mathsf{C}}^{\{i\}}$ for a given nonce R_{C}. Although this makes the protocol stateful, it does eliminate the need to resynchronize the state of the two parties.

Security Problems. A first problem is the fact that the three functions F, G, and H use the same key. In particular, the values $\mathsf{Mac}_{\mathsf{S}}$, $\mathsf{Mac}_{\mathsf{C}}$, and $R_{\mathsf{C}}^{\{i\}}$, which are computed using the key K, are sent across an insecure channel. Since F and H are MACs, the confidentiality of the key K is not fully guaranteed; thus the guarantee of pseudorandomness of G is not sufficient to guarantee the indistinguishability from random of the keys CK, IK. This weakness is remedied if all three functions are assumed to have pseudorandom output.

A more serious problem is a network-corruption attack, which is harder to prevent, and which originates in these two facts: (1) the new procedure to request authentication vectors originates from the server, not from the client (indeed, the client is not aware of whether the server still has pertinent authentication tokens or not); (2) the network-specific identifier Id_{S} is only used in the Mac_2 value. In particular, the attack proceeds as follows:

1. The client C arrives in a vulnerable area (for which the server S^* will be corrupted). This server is authorized to request authentication tokens from the operator and does so. Then, S^* may use several such tokens with the client (but not all). We assume that there will be at least a single authentication vector AV which was not used. The adversary $\mathcal{A}$ then corrupts the server S^*, thereby also retrieving the vector AV.

2. The client leaves the vulnerable area to enter a non-vulnerable one (with an honest server S). The adversary $\mathcal{A}$ acts as a Man-in-the-Middle (MitM) between C and S. It blocks the message R_{fresh} and any other message sent by S, sending instead $\mathsf{R}, \mathsf{Autn}$ from the authentication vector AV.

3. The verifications on the client side pass as long as the client still retains its past value R_{C}. This is not specified exactly in the original paper [109], but considering that it is the *server* that initiates this exchange, it is likely that the client will not automatically replace R_{C} unless prompted by the server.

A possible countermeasure to this vulnerability is to ensure that once the client is aware of having moved from the area associated with S^*, it discards the old R_{C} and aborts unless it is asked to generate a new one.

Practical Aspects. The protocol presented by Zhang [109] is not fully specified, but it does not follow closely the practical constraints of mobile networks. The protocol forces a lot of complexity on the client, which has to verify the uniqueness of the nonce $R_{\mathsf{C}}^{\{i\}}$, to search exhaustively for the correct index which gives an (honestly generated) $R_{\mathsf{C}}^{\{i\}}$, and to generate the randomness R_{C}. Since $R_{\mathsf{C}}^{\{i\}}$ is generated given the secret key K, it must be computed securely: it is not possible to delegate this computation to the phone (which is a more powerful, but untrusted tool). We reiterate that the rationale of the initial UMTS-AKA design was that the client's SIM card could not generate its own randomness.

Another concern is the lack of specificity with respect to the client and operator keys, which are replaced by the generic key K. The fact that this key is used in MAC functions exposes the keys to attackers, as explained in the first attack above. In our results, we prove that for both TUAK and MILENAGE the seven cryptographic algorithms are PRFs with respect to both the client and the operator keys; this is a much stronger result. The same lack of specificity affects the keys CK, IK, which are generically denoted as a secret key SK in the paper of Zhang. We note that the latter is a more sounder cryptographic design, since in the UMTS-AKA protocol, the two keys are output by *different* PRFs, but on the same input. In particular, a stronger property is required than merely the pseudorandomness of the two concerned algorithms: the two values must be independent even for adversarially-controlled input.

3.6 Our Fix Variant of AKA Protocols: SecAKA

In this section, we propose a new fix variant of the AKA protocols, called SecAKA. We only describe this fix variant of the UMTS-AKA protocol since the EPS-AKA protocol can be fixed similarly.

3.6.1 Description

The main reason server-corruption attacks are effective is that servers associated with a specific geographic area (like a country, a region, etc.) can re-use authentication vectors given by the operator in a different geographic area, impersonating the legitimate server associated with that area. This vulnerability, however, is easily fixed as long as the client's device is aware of its geographical location. Our solution is to add a unique server identifier, denoted Id_{S}, to the input of each of the cryptographic functions, thus making any leftover authentication tokens un-replayable in the wrong area. We stress that this is a minor modification to the protocol, as servers are already associated with a unique LAI identifier.

We also show below 3.6.3 how to include Id_{S} in the computation of each of the cryptographic algorithms. These updated algorithms achieves as required the pseudorandomness 3.6.3. We present our modified protocol in Figure 4.5.

3.6.2 Additional Achieved Security Properties of SecAKA Protocol

This modification still (trivially) preserves the properties of strong client-impersonation resistance, soundness, and state-confidentiality. However, the modification yields in addition strong key-indistinguishability and server-impersonation resistance, as we detail below. In this section, we consider the function G and G^* as the unitary functions of the suite of the updated seven algorithms

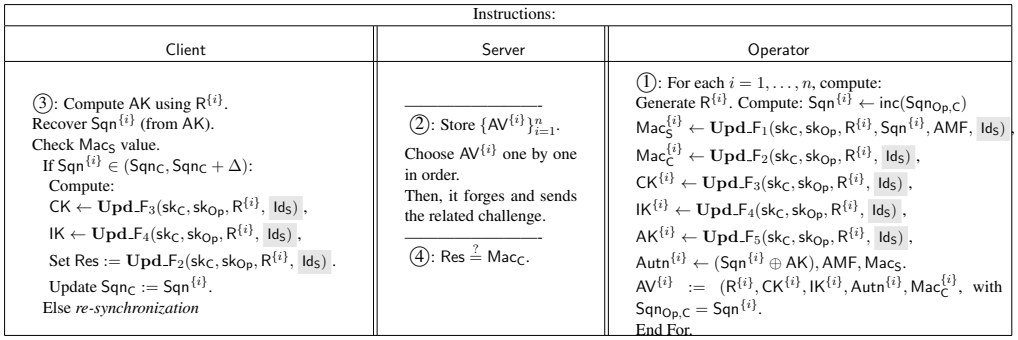


Figure 3.4: The modified instructions of the fixed UMTS-AKA Procedure.

used in the Sec-AKA protocol. The generalisation of these algorithms described in 3.6.3 is really similar to the one did in Section 3.3.1.

Theorem 17. **[S.K.Ind-resistance.]** *Let* $G : \{0,1\}^{\kappa} \times \{0,1\}^{d} \times \{0,1\}^{t} \times \{0,1\}^{t} \rightarrow \{0,1\}^{n}$ *be the unitary function of the suite of the updated seven algorithms specified in Section 3.6.3, and* Π *the fixed* AKA *protocol specified in section 3.6.1. Consider a* $(t, q_{\mathsf{exec}}, q_{\mathsf{res}}, q_{\mathsf{s}}, q_{\mathsf{Op}}, q_G)$*-adversary* $\mathcal{A}$ *against the* S.K.Ind*-security of the protocol* Π*, running in time* t *and creating at most* q_{exec} *party instances with at most* q_{res} *resynchronizations per instance, corrupting at most* q_{s} *servers, making at most* q_{Op} *queries per operator per corrupted server and making* q_G *queries to the function* G*. Denote the advantage of this adversary as* $\mathsf{Adv}_{\Pi}^{\mathsf{S.K.Ind}}(\mathcal{A})$*. Then there exists a* $(t' \approx O(t), q' = 5 \cdot q_{\mathsf{Op}} + q_G + q_{\mathsf{exec}}(q_{\mathsf{res}} + 2))$*-prf-adversary* $\mathcal{A}'$ *on* G *such that:*

$$\mathsf{Adv}_{\Pi}^{\mathsf{S.K.Ind}}(\mathcal{A}) \leq \mathsf{n_C} \cdot \left(\frac{(q_{\mathsf{exec}} + q_{\mathsf{s}} \cdot q_{\mathsf{Op}})^2}{2^{|\mathsf{R}|}} + 2 \cdot \mathsf{Adv}_{G}^{\mathsf{prf}}(\mathcal{A}') \right).$$

Proof. Our proof has the following hops.

Game $\mathbb{G}_0$: This game works as the S.K.Ind-game stipulated in our security model in Section 3.2.

Game $\mathbb{G}_1$: We modify $\mathbb{G}_0$ to only consider the new query $\mathsf{Corrupt}(\mathsf{P}, \mathtt{type})$ but keeping the same goal. We note that this new query permits to consider the corruption of the key operator independently to the corruption of the subscriber keys. This new query behaves as follows:

$\mathsf{Corrupt}(\mathsf{P}, \mathtt{type})$: yields to the adversary the long-term keys of party $\mathsf{P} \neq \mathsf{S}$ (else, if the oracle takes as input $\mathsf{P} = \mathsf{S}$, then it behaves as usual calling the oracle OpAccess). The output of the oracle depends on the value $\mathtt{type} \in \{\mathtt{sub}, \mathtt{op}, \mathtt{all}\}$. If $\mathtt{type} = \mathtt{sub}$, then the returned value is $\mathsf{sk_P}$. If $\mathtt{type} = \mathtt{op}$, then the oracle returns $\mathsf{sk_{Op}}$. Then, for $\mathtt{type} = \mathtt{all}$, we return the both values $\mathsf{sk_P}$, $\mathsf{sk_{Op}}$. If $\mathtt{type} \in \{\mathtt{sub}, \mathtt{all}\}$, then P (and all its instances, past, present, or future), are considered to be adversarially controlled.

We argue that given any adversary $\mathcal{A}$ playing the game $\mathbb{G}_1$ and winning with probability $\epsilon_{\mathcal{A}}$, the same adversary wins the game $\mathbb{G}_0$ with probability at least $\epsilon_{\mathcal{A}}$ (this is trivial since in game $\mathbb{G}_1$, $\mathcal{A}$ has more information).

$$\Pr[\mathcal{A} \text{ wins } \mathbb{G}_0] \quad \leq \quad \Pr[\mathcal{A} \text{ wins } \mathbb{G}_1].$$

Game $\mathbb{G}_2$: We modify $\mathbb{G}_1$ to only allow interactions with a single client. The challenger generates only a single operator key, which is associated with the operator chosen for the registered client. As indicated before, the security loss is given by:

$$\Pr[\mathcal{A}_{\mathbb{G}_2}\mathsf{wins}] \quad \geq \quad \frac{1}{\mathsf{n_C}} \Pr[\mathcal{A}_{\mathbb{G}_0}\mathsf{wins}] + \frac{1}{2} \cdot \left(1 - \frac{1}{\mathsf{n_C}}\right).$$

Game $\mathbb{G}_3$: We modify $\mathbb{G}_2$ to ensure that the random value sampled by honest server instances is always unique. This gives us a security loss (related to the respective collisions between the R in two different instances) of

$$\left| \Pr[\mathcal{A}_{\mathbb{G}_2}\mathsf{wins}] - \Pr[\mathcal{A}_{\mathbb{G}_3}\mathsf{wins}] \right| \leq \frac{(q_{\mathsf{exec}} + q_{\mathsf{s}} \cdot q_{\mathsf{Op}})^2}{2^{|\mathsf{R}|}}.$$

Game $\mathbb{G}_4$: This game behaves as the game $\mathbb{G}_3$ with the restriction to only interact with only one server. The benefices loss is the ability to obtain some authentication challenges from uncorrupted servers. Such authentication challenges can be either to give information about the used sequence number and the long term keys or to forge a fresh challenge replaying some parts of these challenges. We recall that the challenge is split into five parts: a random value, a masked version of the fresh sequence number (an one-time-pad based on an anonymity key generated by the function G), two mac computed with the function G and both session keys. Moreover, we note that all the calls of the function G take in input a specific value of the related server $\mathsf{Id_S}$. Thus, the two session keys can not directly reuse since the random value R is never reused (see previous reduction). So, the session keys will be always different in each session (except if some collisions of the function G appear). The related security loss is given by the collision on two outputs of the same function G with two different inputs (the only differences between the both inputs are at least the value of the network identifier) and by the indistinguishability of the function G which are both guaranteed by the pseudorandomness of G. We recall that the Test Phase of the game can be only focus on a network which is or was never corrupted. This give us a security loss

$$\left| \Pr[\mathcal{A}_{\mathbb{G}_4}\mathsf{wins}] - \Pr[\mathcal{A}_{\mathbb{G}_3}\mathsf{wins}] \right| \leq \mathsf{Adv}^{\mathsf{prf}}_{G}(\mathcal{A}).$$

Game $\mathbb{G}_5$: We modify $\mathbb{G}_4$ to replace outputs of the internal cryptographic functions by truly random, but consistent values (they are independent of the input, but the same input gives the same output). As indicated before, the security loss is given by:

$$\left| \Pr[\mathcal{A}_{\mathbb{G}_4}\mathsf{wins}] - \Pr[\mathcal{A}_{\mathbb{G}_5}\mathsf{wins}] \right| \quad \leq \quad \mathsf{Adv}^{\mathsf{prf}}_{G}(\mathcal{A}).$$

Winning $\mathbb{G}_5$: At this point, the adversary plays a game in the presence of a single client C. The goal of this adversary is to distinguish a random session key to a fresh session key. But, in game $\mathbb{G}_5$, queries to G return truly random, consistent values. In this case, the adversary can do no better than guessing. Thus, we have:

$$\Pr[\mathcal{A}_{\mathbb{G}_5}\mathsf{wins}] = \frac{1}{2}.$$

Security statement: This yields the following result:

$$\mathsf{Adv}^{\mathsf{S.K.Ind}}_{\Pi}(\mathcal{A}_{\mathbb{G}_0}) \leq \mathsf{n_C} \cdot \left(\frac{(q_{\mathsf{exec}} + q_{\mathsf{s}} \cdot q_{\mathsf{Op}})^2}{2^{|\mathsf{R}|}} + 2 \cdot \mathsf{Adv}^{\mathsf{prf}}_{G}(\mathcal{A}) \right).$$

This concludes the proof. □

Theorem 18. **[S.S.Imp-resistance.]** *Let* $G : \{0,1\}^{\kappa} \times \{0,1\}^{d} \times \{0,1\}^{t} \times \{0,1\}^{t} \to \{0,1\}^{n}$ *be the unitary function of the suite of the updated seven algorithms specified in Section 3.6.3 and* Π *our*

fixed variant of the AKA *protocol specified in section 3.6.1. Consider a* $(t, q_{\mathsf{exec}}, q_{\mathsf{res}}, q_{\mathsf{s}}, q_{\mathsf{Op}}, q_{\mathsf{G}}, \epsilon)$*-adversary* $\mathcal{A}$ *against the* S.S.Imp*-security of the protocol* Π*, running in time* t *and creating at most* q_{exec} *party instances with at most* q_{res} *resynchronizations per instance, corrupting at most* q_{s} *servers, making at most* q_{Op} *queries per operator per corrupted server and making* q_G *queries to the function* G*. Denote the advantage of this adversary as* $\mathsf{Adv}^{\mathsf{S.S.Imp}}_{\Pi}(\mathcal{A})$*. Then there exists a* $(t' \approx O(t), q' = 5 \cdot q_{\mathsf{s}} \cdot q_{\mathsf{Op}} + q_G + q_{\mathsf{exec}}(2 + q_{\mathsf{res}}))$*-prf-adversary* $\mathcal{A}'$ *on* G *such that:*

$$\mathsf{Adv}^{\mathsf{S.S.Imp}}_{\Pi}(\mathcal{A}_{\mathbb{G}_0}) \leq \mathsf{n_C} \cdot \left(\frac{q_{\mathsf{exec}} \cdot q_{\mathsf{res}}}{2^{|\mathsf{Mac_S}|}} + \frac{1}{2^{\kappa}} + 2 \cdot \mathsf{Adv}^{\mathsf{prf}}_{G}(\mathcal{A}') \right).$$

Proof. Our proof has the following hops.

Game $\mathbb{G}_0$: This game works as the S.S.Imp-game detailed in Section 3.2.

Game $\mathbb{G}_1$: This game works as the previous game $\mathbb{G}_0$ but including the new query Corrupt(P, type), i.e with the presence of operator keys corruption. The reduction from the game $\mathbb{G}_0$ to the game $\mathbb{G}_1$ is the same as the security proof of the theorem 17. As before, it holds that:

$$\Pr[\mathcal{A}_{\mathbb{G}_0}\mathsf{wins}] \quad \leq \quad \Pr[\mathcal{A}_{\mathbb{G}_1}\mathsf{wins}].$$

Note that adversary $\mathcal{A}_{\mathbb{G}_1}$ makes no extra query.

Game $\mathbb{G}_2$: We modify $\mathbb{G}_1$ to only allow the adversary to interact with a single client (any future CreateCl calls would be answered with an error symbol $\perp$). The challenger only generates a single operator key, which is associated with the operator chosen for the registered client. As indicated before, the security loss is given by:

$$\Pr[\mathcal{A}_{\mathbb{G}_1}\ \mathsf{wins}] \leq \mathsf{n_C} \cdot \Pr[\mathcal{A}_{\mathbb{G}_2}\ \mathsf{wins}].$$

Game $\mathbb{G}_3$: This game behaves as the game $\mathbb{G}_2$ with the restriction to only interact with only one server. The adversary loses the ability to obtain authentication challenges from uncorrupted servers. As detailed in the proof of the strong key-indistinguishability, the related security loss is given by the pseudorandomness of the function G. We recall that the Test Phase of the game can only apply to a server which is not corrupted. This gives us a security loss of:

$$\left| \Pr[\mathcal{A}_{\mathbb{G}_2}\mathsf{wins}] - \Pr[\mathcal{A}_{\mathbb{G}_3}\mathsf{wins}] \right| \leq \mathsf{Adv}^{\mathsf{prf}}_{G}(\mathcal{A}).$$

Game $\mathbb{G}_4$: We modify $\mathbb{G}_3$ to replace outputs to calls to all the internal cryptographic functions by truly random, but consistent values (they are independent of the input, but the same input gives the same output). As detailed in the key-indistinguishability, we obtain:

$$\left| \Pr[\mathcal{A}_{\mathbb{G}_3}\mathsf{wins}] - \Pr[\mathcal{A}_{\mathbb{G}_4}\mathsf{wins}] \right| \leq \mathsf{Adv}^{\mathsf{prf}}_{G}(\mathcal{A}).$$

Winning $\mathbb{G}_4$: At this point, the adversary plays a game with a single client C_i, which only accepts $\mathcal{A}_{\mathbb{G}_4}$ if the authentication challenge is verified for some session sid. Assume that this happens against accepting instance $\mathsf{C_i}$ of the target client, for some target session sid. Note that the MAC value $\mathsf{Mac_S}$ computed by C_i is purely random, but consistent. Thus, the adversary has three options: (a) forwarding a value already received from a honest server for the same input values R; $\mathsf{Id_S}$; Sqn; $\mathsf{sk_{Op}}$; $\mathsf{sk_C}$, of which $\mathsf{sk_C}$ is unknown; (b) guessing the key $\mathsf{sk_C}$; or (c) guessing the response. The first option yields no result since there are no collision between the transcript of two different servers since all the servers have a different server identifier $\mathsf{Id_S}$. The second option happens with a probability of $2^{-|\mathsf{sk_C}|}$. The third option occurs with a probability of $2^{-|\mathsf{Mac_S}|}$ per

session (which is to say per instance and per re-synchronization), thus a total of $q_{\mathsf{exec}} \cdot q_{\mathsf{res}} \cdot 2^{-|\mathsf{Mac_S}|}$. Thus,

$$\Pr[\mathcal{A}_{\mathbb{G}_4}\mathsf{wins}] = 2^{-|\mathsf{sk_C}|} + q_{\mathsf{exec}} \cdot q_{\mathsf{res}} \cdot 2^{-|\mathsf{Mac_S}|}.$$

Security statement: This yields the following result:

$$\mathsf{Adv}^{\mathsf{S.S.Imp}}_{\Pi}(\mathcal{A}_{\mathbb{G}_0}) \leq \mathsf{n_C} \cdot \left(\frac{q_{\mathsf{exec}} \cdot q_{\mathsf{res}}}{2^{|\mathsf{Mac_S}|}} + \frac{1}{2^{\kappa}} + 2 \cdot \mathsf{Adv}^{\mathsf{prf}}_{G}(\mathcal{A}') \right).$$

□

Each of the two bounds above depends linearly on the number of clients $\mathsf{n_C}$; while this number can be as large as, potentially, six billion, the size of the secret keys (128 or 256 bits) and of the random value (128 bits) can still make the bound negligible. The linear factor $\mathsf{n_C}$, however, highlights the importance of using authentication strings longer than 128 bits for authentication.

3.6.3 Update of the internal cryptographic functions

In our variant, we modified the inputs of the internal cryptographic algorithms to include the new value $\mathsf{Id_S}$. Thus, we need to provide an update of these algorithms. As specified previously, the AKA protocol can be based on two different sets of algorithms: TUAK and MILENAGE. To preserve backward compatibility, we propose to keep track of and update these two sets.

The seven internal cryptographic functions used in the AKA protocol take s in input the following values:

- keys: the couple of 128-bit (or 256-bit) keys: the subscriber and operator keys ($\mathsf{sk_C}$,$\mathsf{sk_{Op}}$).
- Sqn (for the functions $\mathbf{Upd}\_\mathsf{F}_1$ and $\mathbf{Upd}\_\mathsf{F}_1^*$): a 48-bit sequence number.
- AMF (except for the functions $\mathbf{Upd}\_\mathsf{F}_1$ and $\mathbf{Upd}\_\mathsf{F}_1^*$): a 16-bit authentication field management value.
- R: a 128-bit random value.
- $\mathsf{Id_S}$: a 128-bit (public) value characterizing the visited network.

We note that the functions $\mathbf{Upd}\_\mathsf{F}_1$ and $\mathbf{Upd}\_\mathsf{F}_1^*$ behave differently because they take as an additional input the value of the sequence number.

Update of MILENAGE **algorithms:** In order to ensure a stronger degree of security, we also modify the MILENAGE algorithms to output 128-bit MAC and session keys CK and IK.

Based on the Advanced Encryption Standard (AES), these functions compute firstly both values Top_C and Temp as follows:

$$\mathsf{Top}_C = \mathsf{sk_{Op}} \oplus \mathsf{AES}_{\mathsf{sk_C}}(\mathsf{sk_{Op}}), \mathsf{Temp} = \mathsf{AES}_{\mathsf{sk_C}}(\mathsf{R} \oplus \mathsf{Top}_C \oplus \mathsf{Id_S}).$$

The outputs of the MILENAGE algorithms are computed as follows:

- **Output** $\mathbf{Upd}\_\mathsf{F}_1$**:** $\mathsf{Mac_C} = \mathsf{AES}_{\mathsf{sk_C}}(\mathsf{Temp} \oplus \mathsf{Rot}_{\mathsf{r}_1}(\mathsf{Sqn}\|\mathsf{AMF}\|\mathsf{Sqn}\|\mathsf{AMF}) \oplus \mathsf{c}_1) \oplus \mathsf{Top}_C$,
- **Output** $\mathbf{Upd}\_\mathsf{F}_1^*$**:** $\mathsf{Mac}^* = \mathsf{AES}_{\mathsf{sk_C}}(\mathsf{Temp} \oplus \mathsf{Rot}_{\mathsf{r}_6}(\mathsf{Sqn}\|\mathsf{AMF}\|\mathsf{Sqn}\|\mathsf{AMF}) \oplus \mathsf{c}_6) \oplus \mathsf{Top}_C$,
- **Output** $\mathbf{Upd}\_\mathsf{F}_2$**:** $\mathsf{Mac_S} = \mathsf{AES}_{\mathsf{sk_C}}(\mathsf{Rot}_{\mathsf{r}_2}(\mathsf{Temp} \oplus \mathsf{Top}_C) \oplus \mathsf{c}_2) \oplus \mathsf{Top}_C$
- **Output** $\mathbf{Upd}\_\mathsf{F}_3$**:** $\mathsf{CK} = \mathsf{AES}_{\mathsf{sk_C}}(\mathsf{Rot}_{\mathsf{r}_3}(\mathsf{Temp} \oplus \mathsf{Top}_C) \oplus \mathsf{c}_3) \oplus \mathsf{Top}_C$,
- **Output** $\mathbf{Upd}\_\mathsf{F}_4$**:** $\mathsf{IK} = \mathsf{AES}_{\mathsf{sk_C}}(\mathsf{Rot}_{\mathsf{r}_4}(\mathsf{Temp} \oplus \mathsf{Top}_C) \oplus \mathsf{c}_4) \oplus \mathsf{Top}_C$,

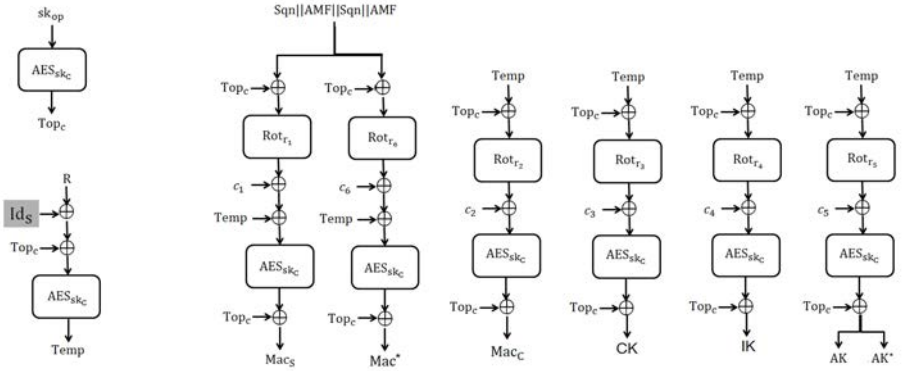


Figure 3.5: Updated MILENAGE.

- **Output $\mathbf{Upd}\_\mathsf{F}_5$:** $\mathsf{AK} = \lfloor \mathsf{AES}_{\mathsf{sk_C}}(\mathsf{Rot}_{\mathsf{r}_5}(\mathsf{Temp} \oplus \mathsf{Top}_C, \mathsf{r}_5) \oplus \mathsf{c}_5) \oplus \mathsf{Top}_C \rfloor_{0..47}$,
- **Output $\mathbf{Upd}\_\mathsf{F}_5^*$:** $\mathsf{AK}^* = \lfloor \mathsf{AES}_{\mathsf{sk_C}}(\mathsf{Rot}_{\mathsf{r}_5}(\mathsf{Temp} \oplus \mathsf{Top}_C, \mathsf{r}_5) \oplus \mathsf{c}_5) \oplus \mathsf{Top}_C \rfloor_{80..127}$,

with the five integers $\mathsf{r}_1 = 0$, $\mathsf{r}_2 = 16$, $\mathsf{r}_3 = 32$, $\mathsf{r}_4 = 64$, $\mathsf{r}_5 = 80$ and $\mathsf{r}_6 = 96$ in the range $\{0, 127\}$, which define the number of positions the intermediate variables are cyclically rotated by the right, and the five 128-bit constants c_i such as:

- $\mathsf{c}_1[i] = 0, \forall i \in \{0, 127\}$.
- $\mathsf{c}_2[i] = 0, \forall i \in \{0, 127\}$, except that $\mathsf{c}_2[127] = 1$.
- $\mathsf{c}_3[i] = 0, \forall i \in \{0, 127\}$, except that $\mathsf{c}_3[126] = 1$.
- $\mathsf{c}_4[i] = 0, \forall i \in \{0, 127\}$, except that $\mathsf{c}_4[125] = 1$.
- $\mathsf{c}_5[i] = 0, \forall i \in \{0, 127\}$, except that $\mathsf{c}_5[124] = 1$.
- $\mathsf{c}_6[i] = 0, \forall i \in \{0, 127\}$, except that $\mathsf{c}_6[123] = 1$.

This is also described in Figure 3.5.

Update of TUAK **algorithms:** We update these algorithms by only modifying the inputs of the second permutation.

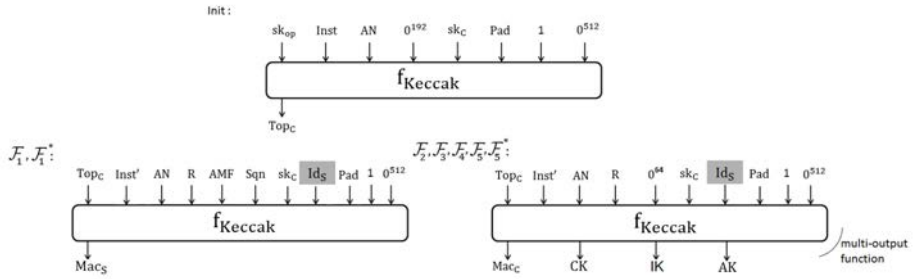


Figure 3.6: Updated TUAK.

We recall that in this instantiation, the functions $\mathbf{Upd}\_\mathsf{F}_1^*$ and $\mathbf{Upd}\_\mathsf{F}_5^*$, used for the resynchronization procedure, behave in the same way but use different values Inst', Inst. We first compute

the value Top_C as follows:

$$\mathsf{Top}_C = \lfloor \mathsf{f}_{\mathsf{Keccak}}(\mathsf{sk}_{\mathsf{Op}} \| \mathsf{Inst} \| \mathsf{AN} \| 0^{192} \| \mathsf{Key} \| \mathsf{Pad} \| 1 \| 0^{512}) \rfloor_{1..256}.$$

We note that the values AN, Inst', Inst, Pad are the same as used in the original TUAK algorithms and Key the (padded) subscriber key. At this point, the behaviour of the functions $\mathbf{Upd}\_\mathsf{F}_1$ (resp. $\mathbf{Upd}\_\mathsf{F}_1^*$) diverges from the other functions. Generating the related output, we compute the value Val_1 and for the others ones, we compute the value Val_2 which differ including the 128-bit value Id_S and the smaller paging value Pad.

$$\mathsf{Val}_1 = \mathsf{f}_{\mathsf{Keccak}}(\mathsf{Top}_C \| \mathsf{Inst}' \| \mathsf{AN} \| \mathsf{R} \| 0^{64} \| \mathsf{Key} \| \mathsf{Id}_\mathsf{S} \| \mathsf{Pad} \| 10^{512}),$$
$$\mathsf{Val}_2 = \mathsf{f}_{\mathsf{Keccak}}(\mathsf{Top}_C \| \mathsf{Inst}' \| \mathsf{AN} \| \mathsf{R} \| \mathsf{AMF} \| \mathsf{Sqn} \| \mathsf{Key} \| \mathsf{Id}_\mathsf{S} \| \mathsf{Pad} \| 10^{512}).$$

Then, we obtain the output of the seven functions truncating the related value as follows:

- **Output $\mathbf{Upd}\_\mathsf{F}_1$:** $\mathsf{Mac}_\mathsf{S} = \lfloor \mathsf{Val}_2 \rfloor_{0..127}$,
- **Output $\mathbf{Upd}\_\mathsf{F}_2$:** $\mathsf{Mac}_\mathsf{C} = \lfloor \mathsf{Val}_1 \rfloor_{0..127}$,
- **Output $\mathbf{Upd}\_\mathsf{F}_3$:** $\mathsf{CK} = \lfloor \mathsf{Val}_1 \rfloor_{256..383}$,
- **Output $\mathbf{Upd}\_\mathsf{F}_4$:** $\mathsf{IK} = \lfloor \mathsf{Val}_1 \rfloor_{512..639}$,
- **Output $\mathbf{Upd}\_\mathsf{F}_5$:** $\mathsf{AK} = \lfloor \mathsf{Val}_1 \rfloor_{768..815}$.

This is also depicted in Figure 3.6.

We note that the multi-output property is, as in the original version, not an issue for the security of the master key, since during one session we can have as many as four calls to the same function with similar inputs (and a different truncation).

The pseudorandomness of the updated TUAK and MILENAGE algorithms can be proved and the generalition as a more-or-less unitary function can be done as 3.3.3.

Chapter 4

A Privacy Analysis of the AKA Protocols

Contents

4.1 A Privacy Analysis

Proposed by the 3rd Generation Partnership Project (3GPP) as a standard for 3G and 4G mobile-network communications, the AKA protocols are meant to provide a mutually-authenticated key-exchange between clients and associated network servers. As we explained in the previous chapter, AKA protocols must basically guarantee the indistinguishability from random of the session keys (key-indistinguishability), as well as client- and server-impersonation resistance considering man-in-the-middle adversary and malicious servers. A paramount requirement is also that of client privacy, which 3GPP defines in terms of: user identity confidentiality, service untraceability, and location untraceability. Following the description of client-tracking attacks e.g. by using error

messages or IMSI catchers, Van den Broek et al. and respectively Arapinis et al. each proposed a new variant of UMTS-AKA, addressing such problems. In this chapter we use the approach of provable security to show that the exciting variants still fail to guarantee the privacy of mobile clients. We propose an improvement of AKA protocols, called PrivAKA, which retains most of its structure and respects practical necessities such as key-management, but which provably attains security with respect to servers and Man-in-the-Middle (MitM) adversaries. Moreover, it is impossible to link client sessions in the absence of client-corruptions. Then, we prove that any variant of AKA retaining its mutual authentication specificities cannot achieve client-unlinkability in the presence of corruptions. In this sense, our proposed variant is optimal. Finally, we consider additional practical and legal considerations, proposing patents for a possible standardization.

4.2 Privacy Requirements of the UMTS-AKA Protocol

The Third Generation Partnership Project (3GPP), which designed the UMTS-AKA protocol in the TS.33.102 specification [13], lists the following privacy requirements:

- **user identity confidentiality:** specifically, *"the property that the permanent user identity (*IMSI*) of a user [...] cannot be eavesdropped on the radio access link."*
- **user untraceability:** namely, *"the property that an intruder cannot deduce whether different services are delivered to the same user by eavesdropping on the radio access link."*
- **user location confidentiality:** in particular, *"the property that the presence or the arrival of a user in a certain area cannot be determined by eavesdropping on the radio access link."*

The requirements quoted above are quite informal; moreover, the nomenclature is confusing, since in the *provable-security* literature, *untraceability* refers to adversaries tracing clients in distinct protocol runs (rather than it being service-related). We discuss the three requirements below, then formalize them into cryptographic requirements. User identity confidentiality concerns only the client's *permanent* IMSI value (not, e.g. the client's sequence number) with respect to *passive* attackers (rather than active ones). However, mobile networks are notoriously prone to Man-in-the-Middle (MitM) active attacks like the IMSI catcher [62], which allows a third party (the MitM) to recover a client's IMSI. Another highly-trackable client-specific parameter is the sequence number Sqn, whose updating procedure is very simplistic and its output, predictable even without a secrey key. As a consequence we require the stronger property of *provable unlinkability*, which ensures that even an *active* MitM cannot *link* two UMTS-AKA protocols run to the same client. For user untraceability, no attacker must know whether the same service (i.e. *any* message-exchange over the secure channel) is provided to a client multiple times. From the point of view of provable security, this is equivalent to *key-indistinguishability* if the authenticated-encryption algorithms are assumed to be secure. User location confidentaility demands that eavesdroppers $\mathcal{A}$ cannot detect the presence of a client in a given area; however, the definition does not specify what information $\mathcal{A}$ links to each client (e.g. the IMSI, the sequence number, etc.). Attackers are aware of the current LAI; the difficulty lies in learning which clients *enter* the area. Unfortunately the UMTS-AKA protocol always reveals the *past* location of any arriving client, making unique (or rare) itineraries stand out. We formalize a strong degree of location privacy as a part of client-unlinkability.

Our formalizations of *client unlinkability* and *key-indistinguishability* consequently guarantee 3GPP's three privacy requirements.

4.3 Privacy Weaknesses of the UMTS-AKA Protocol

In this section, we focus on the (provable) privacy of UMTS-AKA, which completes the security analysis detailed in chapter 3. We do not need to have a very formal/theoric analysis to point out the pure UMTS-AKA protocol cannot guarantee user unlinkability as defined above. Indeed, the user identification based on temporary identifiers independent to the permanent identifier is not sufficient. Two main attacks in the literature, have been proposed namely IMSI catcher attacks [62] and IMSI paging attacks [97,98] already prove that UMTS-AKA does not offer the desired degree of client privacy. Paging request refers to the process used when server needs to locate a user equipment in a particular area. This request being not directly out of the scope of AKA protocols, we will not focused on the paging attacks. IMSI catchers allow passive and active adversaries to track clients by exploiting the fact that during the protocol run, the server will require clients to send their permanent identifier IMSI if the TMSI value cannot be traced back to an IMSI.

A known *privacy* problem of the UMTS-AKA protocol is the "IMSI catcher" attack [62], which exploits the fact that the permanent identifier IMSI is sent as a back-up for a faulty tuple (TMSI, LAI). By either observing such faulty behaviour (due to transmission errors or to server database problems), or by causing it, MitM attackers can easily track clients, in blatant violation of the user identity confidentiality requirement. This is a problem in all generations of mobile communication networks. Van den Broek et al. [104] addressed IMSI catchers by replacing IMSI values by an unlinkable pseudonym PMSI. We discuss the properties and weaknesses of this countermeasure in details in Section 4.6.2.

Arapinis et al. [31] showed that failure messages (in authentication/resynchronization) can be used to trace users. A subsequent paper by some of the same authors [29] cleverly identifies ways in which a specific implementation deviates from 3GPP recommendations and thus allows the linkability of client sessions.

An older work by Ateniese et al. [33] examines the problem of untraceable mobility, in particular noting an informal paradigm: nobody but the client should know both the client's identity and location at the same time. In this context, they provide solutions to achieving better privacy in 3-party settings; however, these solutions are extremely generic, which makes them hard to immediately applies them to UMTS-AKA. Moreover, note that server-operator communication takes place across a channel that is implemented differently in practice by different operators. The protocols proposed by [33] require this channel to be implemented in a specific way. Finally, we note that, while highly worthwhile, the goal of preventing servers from learning a client's operator is incompatible with the way authentication is currently done in the UMTS-AKA protocol (operators need to prepare session information for servers, across a secure, and mutually authenticated channel which forcibly reveals the identity – and implicitly the location of the server).

Although the UMTS-AKA protocol was designed to protect client privacy, several attacks also be used to break this property at the physical layer or by using side-channel information. Since we focus only on the privacy of UMTS-AKA at the protocol layer, this type of attacks is out of scope. Another class of attacks that is out of scope for our analysis are those that exploit faults in TMSI reallocation choices (for instance the fact that TMSIs are not updated sufficiently fast, or according to protocol), denial-of-service by jamming, or attacks enabled by version-shifting (forcing the user to employ a weaker, more vulnerable version of the protocol on 2G and 3G) [98]. Such attacks, however, indicate that the infrastructure and protocols existing *around* the UMTS-AKA handshake should be more closely examined; if possible, backward compatibility should be discouraged in order to guarantee better security and privacy.

We clearly analyze the two main privacy vulnerabilities to user unlinkability in UMTS-AKA. The first one is related to the operational difficulties related to the TMSI. The second concerns on the linkability of failure messages. These weaknesses have been partially described by Arapinis

and al. in [29,31] and Strobel [62]. These attacks are more-or-less efficient and do not necessarily point the same weaknesses. Any of the three requirements defining the user privacy cannot be guaranteed considering these attacks. One important condition to evaluate the impact of an attack is the *parallelization*, i.e the ability for an attack to do not only focus on one user, but some of them at the same time. The board 4.1 sums up the different features of each attacks and ranks the attacks considering feasibility and impacts of each of them. We note that the ranking is established comparing the attacks which target the same requirement.

Attacks	Parallelizable For Privacy	Key Vulnerabilities	Breakability of Privacy Requirement	Ranking	
				Feasibility	Impact
IMSI catcher (passive)	Yes	User Identification	User identity confidentiality	①	②
IMSI catcher (active)	Yes	User Identification	User identity confidentiality	②	①
	No	User Identification	User untraceability	①	②
TMSI management	Yes	TMSI Re-Allocation	User untraceability	②	①
Linkability of failure messages (active)	No	Management of the failure messages	User untraceability	③	③
Reveal Location	Yes	User identification	User Location Confidentiality	①	①

Figure 4.1: Comparison between attacks against user privacy property of the UMTS-AKA.

Operational difficulties with TMSI. Each mobile client owns a unique permanent identifier, denoted IMSI[1]. If such a value is directly used to identify the client to the server, the user privacy could be not guaranteed. Indeed, this value is unique and permanent, thus each time a client sends its IMSI in clear, any man-in-the-middle adversary eavesdropping the communication can recognize the client. The IMSI cannot be considered as a public and we consider that any adversary can make the link between a mobile client and a IMSI value.

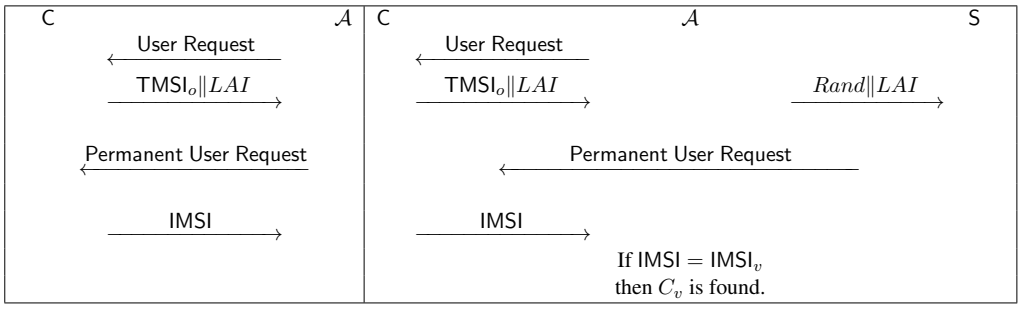
As we explained previously, 3GPP has standardized the use of temporary values TMSI, instead to use permanent identifier IMSI every time. These temporary values are generated, independently of the permanent identifier, by the local server and are exchanged after a successful execution of a UMTS-AKA session. At first glance, this option could guarantee at least the confidentiality of the user identity.

Several attacks have been proposed to force the client to use its permanent identity instead of the temporary one. In the UMTS-AKA protocol, the IMSI is only used when the user identification using temporary identifier fail. Such event can naturally arrived when a session of the UMTS-AKA protocol or a TMSI re-allocation fail. These events are pretty rare but can arrived since the messages are sent toward a radio link. Thus, some natural abort can arrive. This feature is exploited by the well-know passive *IMSI catcher* [62]. A passive IMSI catcher is a MitM adversary located in the access network sniffing the radio communication in a specific area and detects all the exchanged IMSI values. Since we consider the natural abort as not a regular event, this passive sniffing is not efficient. However, such abort can be easily forced for an active Man-in-the-Middle.

The active IMSI catcher consists in forcing a mobile client to reveal its permanent identifier In the identification of the user, when the VLR cannot recognize the received temporary identifier TMSI it requires the user's permanent identifier. Thus, by a simple attack as described on the left side of the 4.2, an adversary can obtain and force the use of the IMSI during a fresh session. Moreover, with such a behaviour, the active IMSI catcher can know the identity of the clients in a specific area dropping (or randomizing) all the answer to a "User Request". Thus, these attacks prove the lack of user identity confidentiality. Additionally, this attack can be used to check if a

[1]To the best of our knowledge, the specifications do not detail how the permanent user identifiers are generated, we consider that all the IMSI value are different.

specific client is located in this area or not. This attack is described on the right side of the figure 4.2. This attack proves the lack of user untraceability. In that sense, the active MitM is the best known attack to mobile telephony users'privacy).



Principle of IMSI catcher

Traceability of a victim client C_v

Figure 4.2: Active Attacks based on TMSI management.

The temporary identifier of each user must be used as its name suggests temporarily. The technical specifications from 3GPP, explicitly require that TMSIs are used only a single time. As specified by the UMTS-AKA protocol, a TMSI re-allocation is provided. However, a basic attack can corrupt this uniqueness. Indeed, we note that the VLR does not de-allocate the old value TMSI_o without receiving the acknowledgement message. So if an adversary drops that message, the VLR has both the allocated temporary values: the old TMSI_o and the TMSI_n. For the next identification, the VLR will accept both values to identify the user. So an adversary dropping the acknowledgement message can easily force the replay of the same TMSI in two independent sessions. Moreover, we note that the new temporary identifier TMSI_o is only sent encrypted with the session key CK and without any authentication. The mobile subscriber cannot verify whether the received value is sent by the VLR or by malicious entity. Thus, any adversary can send a false random value as the new TMSI by sending a random instead of the ciphered temporary identifier. Therefore, the user untraceability cannot be guaranteed the lack of authenticity any TMSI uniqueness.

A TMSI value is associated to a specific location area, denoted LAI, identifying the VLR having generated this temporary identifier. The LAI value is sent in cleartext with the TMSI during the user identification. As a consequence, an adversary can associate the LAI with the related IMSI (with an active IMSI catcher [62]): that permits to know where the subscriber with the permanent identifier IMSI was located during the previous session. Therefore, the user location confidentiality cannot be guaranteed.

Linkability of failure messages. Arapinis and al [31] notably provides a practical attack establishing the traceability of a user based on the study of the failure messages. Considering a victim mobile client $\mathsf{C_v}$, an adversary can detect its presence in a specific area only analyzing the response of an authentication challenge. In the UMTS-AKA protocol, a mobile client receiving such a challenge can answer either with an accepting response "authentication response" when the server is authenticated, a "Mac Failure Message" when the authentication fails, or a resynchronization procedure when the authentication check is successful but the received sequence number is unfresh. We note that one important point when the client receives the challenge: it firstly check the mac value to authenticate the client and only if this check is successful, it verifies the received sequence number. We recall that the check is based on the received sequence number. Such management implies that an *old* authentication challenge replayed to the correct mobile client will never imply a Mac failure but only a resynchronization procedure, except to a negligible probability of (2^{-128}) (due to the collision's probability). A replay of an authentication challenge always implies a fail-

ure message. An old authentication challenge replayed to the bad client the answer will be a Mac Failure Message because the Mac could not be correct except with a similar negligible probability. This difference can permit to trace a mobile client in a specific area just replaying a legitimate authentication challenge. This attack is no-parallelizable since it is focused on only one user, the one who has received the legitimate authentication challenge. It represents a breach of the user's untraceability since any active MitM adversary located in a specific area can check if a client is present or not. This active attack can be considered as cheap since no computation is required and it is just a replay attack, but it is no cheaper than the IMSI catchers. The weaknesses of the user identification are not exploited to establish this attack. Thus, only fix the user identification procedure cannot entirely provide the user privacy.

4.4 Does EPS-AKA Protocol Guarantee more Client Privacy than UMTS-AKA Protocol?

The EPS-AKA protocol tries to protect the user identifier confidentiality against Man-in-the-Middle (MitM) adversary in the same way as the UMTS-AKA scheme i.e., by using a temporary identifier. The Global User Temporary Identity (GUTI) is assigned to the user by the MME. This value is used to provide an unambiguous user identification without revealing the permanent identifier IMSI. The GUTI is slightly different from the TMSI used in 3G networks. In the EPS architecture, the temporary identifier includes an identifier of the MME which has generated the value. This identifier replaces the Location Area Identifier (LAI) used with the TMSI in the UMTS-AKA. This value is essential for the communication between the MMEs and notably the Local GUTI Unknown Procedure when a client uses a temporary identifier which has not been generated by the current server.

The Third Generation Partnership Project (3GPP), which designed the EPS-AKA protocol in the TS.33.401 specification [8], lists the following privacy requirements: *user identity confidentiality*, *user untraceability* and *user location confidentiality*. These properties are similar to the ones define for the 3G networks but with few refinements which do not concern totally the EPS-AKA protocol directly. Indeed, they notably concern the confidentiality of some others identifiers (IMEI and IMEISV) which shall be securely stored in the terminal.

All the attacks on the UMTS-AKA protocol against the user identity confidentiality and user untraceability properties can be applied directly against the EPS-AKA protocol. The third requirement demands that the presence of a client in a given local area is hidden from eavesdroppers. Since GUMMEIs, included in the temporary identifier, are not confidential, attackers are aware of the area they are in; thus the difficulty lies in learning which clients *enter* the area. A weakness of the current EPS-AKA protocol is that MitM attackers can always learn the *past* location of any arriving client studying the first part of the temporary identifier. Unique itineraries may thus stand out.

In conclusion, the respect of user privacy is as unrespected in the 4G networks as in the 3G networks.

4.5 Privacy Model

Due to their orthogonality, it is hard to formalize the notion of client-unlinkability in the same generic framework used to define security properties as key-indistinguishability. One difficulty is the fact that the unlinkability notion requires the adversary to have access to clients without knowing their identities. Following established approaches [72, 106], in the unlinkability model, we associate clients with identifiers, or handles, denoted VC (Virtual Client), and this changes the

syntax of the oracles we use.

The privacy model uses similar oracles than the ones of the security models, but with a slightly different syntax, for the two types of definitions, and thus obtain security guarantees based on traditional Bellare, Pointcheval, and Rogaway models [39].

Set up and participants. We consider a set $\mathcal{P}$ of participants, which are either a server S_i or a mobile client C_i of the type respectively VLR or ME/USIM. By contrast operators Op are not modelled as active parties. In all security games apart from state-confidentiality and soundness with respect to the server, the operators Op are black-box algorithms within the server S; in those two games, the operators are oracles, which the adversary (i.e. the server) may query. We assume the existence of $\mathsf{n_C}$ clients, n_S servers, and n_{Op} operators. If the operators are contained within the servers, we assume that all copies of the same operator Op are synchronized at all times. We associate each client with: a long-term, static secret state consisting of a tuple $(\mathsf{sk_C}, \mathsf{sk_{Op}})$, an ephemeral state $\mathsf{st_C}$ consisting of a sequence number $\mathsf{Sqn_C}$, a tuple of a static, permanent identifier IMSI and an ephemeral, temporary identifier TMSI, and finally a tuple of a current, and a past local area identifier, denoted $\mathsf{past.LAI_P}$ and $\mathsf{curr.LAI_P}$ respectively. Servers are associated with a permanent local area identifier LAI and a unique network identifier $\mathsf{ID_{S_i}}$; they also keep track of a list of tuples $(\mathsf{TMSI}, \mathsf{IMSI})$ associated with clients. Each of the at most n_S servers has black-box access to algorithms (or oracles in the case of state-confidentiality and soundness) $\mathsf{Op}_1, \ldots, \mathsf{Op}_{n_{\mathsf{Op}}}$, which are initialized with long-term keys $(\mathsf{sk}_{\mathsf{Op}_i})$ and keep track of a list of tuples $(\mathsf{IMSI}, \mathsf{sk_C}, \mathsf{Sqn}_{\mathsf{Op,C}})$. In our model, we also assume that the key space of all operators is identical (otherwise it becomes easier to distinguish between clients of different operators).

Client-Unlinkability. Informally, we call a protocol Π client-unlinkable if no adversary can know whether two executions of Π were run by the same, or by two different clients. Two sessions associated with the same client are called *linked*. Following previous works of Vaudenay [106] and Hermans et al. [72], we give the adversary access to a basic left-or-right oracle, which associates an anonymized handle virtual client VC to one of two possible clients (input by the adversary). We extend this framework to account for client mobility, giving the adversary access to a relocation oracle. Consequently, if an attacker can distinguish between clients based on their location, it will win the unlinkability game, which we detail below. At the onset of this game, the set of clients is empty and the challenger instantiates two lists $\mathcal{L}_{\mathsf{drawn}}$ and $\mathcal{L}_{\mathsf{free}}$. We initialize operators by choosing their secret keys. The adversary can then initialise servers by choosing their locations, and it can create clients to populate the system it attacks. For each newly-created client, the past location $\mathsf{past.LAI_C}$ is set to a special symbol $\perp$ and the current location is adversarially-chosen. The adversary then interacts with clients by means of several oracles. The lists $\mathcal{L}_{\mathsf{drawn}}$ and $\mathcal{L}_{\mathsf{free}}$ correspond to the two possible states of any one client. Clients can be "drawn" or "free"; at creation, all clients are "free", and they may become "drawn" if used as input to a left-or-right Client-Drawing oracle. In particular, we use a left-or-right Client-Drawing oracle (similar to that in [72, 106]), which allows the adversary to interact with one of two clients (the interacting client being chosen depending on a secret bit b). Clients input to the Drawing oracle are moved to the $\mathcal{L}_{\mathsf{drawn}}$ list; further Drawing queries can then be made concurrently as long as the input clients are in the $\mathcal{L}_{\mathsf{free}}$ list[2]. Upon drawing one of two possible clients, the adversary is given a handle on the chosen entity; following the notation of [106], we call this a *virtual* client and we associate it with the handle VC. Virtual clients can then be freed by the adversary (this would remove them from the $\mathcal{L}_{\mathsf{drawn}}$ list and re-add them to the $\mathcal{L}_{\mathsf{free}}$ list). Only free clients can be drawn. This oracle associates a handle to either the left or the right input client, depending on a secret bit b. The client unlinkability property is defined in terms of the following security experiment $\mathsf{Exp}^{\mathsf{c.unlink}}_{\mathcal{A}}(1^\lambda)$, for

[2] In particular, we want to avoid trivial attacks, in which an adversary can distinguish a client simply because it is not in its original state (having already started the protocol run beforehand).

a security parameter (in unary) 1^λ.

- The challenger randomly chooses a bit $b \in \{0,1\}$.
- The adversary may use the oracles below (with restrictions depending on its adversarial class), and the challenger answers the queries.
- The adversary finally outputs a guess d of the bit b.

We say the adversary *wins* if and only if $d = b$, and we define the adversary's advantage of winning this game against a protocol Π as:

$$\mathcal{A}_{\Pi}^{\mathsf{c.unlink}}(\mathcal{A}) := |\mathsf{Pr}[\mathcal{A} \text{ wins } \mathsf{Exp}_{\mathcal{A}}^{\mathsf{c.unlink}}(1^\lambda)] - \frac{1}{2}|.$$

Recalling the adversarial classes of [72, 106], we call an adversary *narrow* if it may not use the Result oracle, permitting it to know whether the server has authenticated the client or not. The opposite of *narrow* are *wide* adversaries. Orthogoal to the use of the Result oracle, we also classify adversaries in terms of their use of the Corrupt oracle, which gives them access to the client's data. Thus, adversaries are *weak* if they cannot use the corruption oracle; they are *forward* if any corruption query may only be followed by more corruption queries[3]. Finally, adversaries are classified as *strong* if their access to oracles is unrestricted. We note that the 3GPP requirements outlined in Section 4.1 restrict their adversaries to "*eavesdroppers on the radio link*", which seems to indicate that they target *weak* adversaries that are either *narrow* or *wide*. Moreover, they restrict their adversaries to being passive only; in this study, we also consider active weak attackers and thus obtain a better privacy guarantee.

Formalization. We quantify adversaries in terms of the following parameters: the adversarial class, which we abbreviate to α-c.unlink, with $\alpha \in \{\mathsf{nw}, \mathsf{ww}, \mathsf{nf}, \mathsf{wf}\}$ (for narrow- and wide-weak, and narrow-, respectively wide-forward adversaries); their execution time is t; the maximum number q_{exec} of sessions instantiated per client C; the maximum number q_{id} of user identification per session; and the maximum number q_G of queries to the function G. We formalize the following definitions.

Definition 13. [Weak Unlinkability] *A protocol* Π *is* $(t, q_{\mathsf{exec}}, q_{\mathsf{id}}, q_G, \epsilon)$-nw/ww*-client-unlinkable if no narrow/wide-weak-adversary running in time* t*, creating at most* q_{exec} *sessions and* q_{id} *user identification per session, and making at most* q_G *queries to the function* G*, has an advantage* $\mathsf{Adv}_{\Pi}^{\mathsf{w.c.unlink}}(\mathcal{A}) \geq \epsilon$.

Definition 14. [Forward Unlinkability] *A protocol* Π *is* $(t, q_{\mathsf{exec}}, q_{\mathsf{id}}, q_G, \epsilon)$-nf/wf*-client-unlinkable if no narrow/wide-forward-adversary running in time* t*, creating at most* q_{exec} *sessions and* q_{id} *user identification per session, and making at most* q_G *queries to the function* G*, has an advantage* $\mathsf{Adv}_{\Pi}^{\mathsf{f.c.unlink}}(\mathcal{A}) \geq \epsilon$.

Oracles. The adversary interacts with the system by means of the following oracles, in addition to a function G, which we model as a PRF and which "encompasses" all the cryptographic functions of the AKA protocol:

- CreateCl(Op, LAI) $\rightarrow$ $(\mathsf{C}_i, \mathsf{IMSI}, \mathsf{st}_{\mathsf{C}_i})$: this oracle creates a new, legitimate, free client, labelled C_i at a location LAI for which a server is already defined (else the oracle outputs $\perp$). The client's IMSI, its state $\mathsf{st}_{\mathsf{C}_i}$, and its secret key $\mathsf{sk}_{\mathsf{C}_i}$ are chosen uniformly at random from sets ID, ST, and $\mathcal{S}$ respectively; the past location and TMSI are set to a special symbol $\perp$. The client's operator key $\mathsf{sk}_{\mathsf{Op}}$ is set to the key of the operator Op. The adversary is given the parameters IMSI, $\mathsf{st}_{\mathsf{C}_i}$, and the label C_i (used later to select clients).

[3] In particular, the adversary may no longer free drawn clients, nor interact with servers or clients

- $\mathsf{CreateS}(\mathsf{LAI}) \rightarrow \mathsf{S}_i$: this oracle generates a new server S_i at location LAI, if this location is not already defined for another server (else the oracle returns $\perp$).

- $\mathsf{Launch}(\mathsf{VC}, \mathsf{S}_j) \rightarrow (s, m)$: this oracle instantiates a new session (labelled by a unique identifier s) between the client associated with VC and the server S_j, and outputs an initial protocol message m from S_j to VC. This oracle keeps track of tuples $(s, \mathsf{VC}, \mathsf{S}_j)$.

- $\mathsf{DrawCl}(\mathsf{C}_i, \mathsf{C}_j) \rightarrow \mathsf{VC}$: on input a pair of client labels, this oracle generates a handle VC, which is a monotonic counter, if the following conditions are met: (a) both clients were free when the query was made; (b) both clients have the same current location value. If either condition is not met, the oracle outputs $\perp$. Else, depending on the value of the secret bit b, the challenger associates the handle VC either with C_i (if $b = 0$) or with C_j (if $b = 1$). The challenger stores the triple $(\mathsf{VC}, \mathsf{C}_i, \mathsf{C}_j)$ in a table $\mathcal{T}$.

- $\mathsf{FreeVC}(\mathsf{VC}) \rightarrow \perp$: on input the virtual handle VC, this oracle retrieves the values $\mathsf{C}_i, \mathsf{C}_j$ associated to VC in the table $\mathcal{T}$, aborting any ongoing protocol runs.

- $\mathsf{Relocate}(\mathsf{VC}, \mathsf{LAI}^*) \rightarrow \perp$: this oracle modifies the *current* location of the two clients $\mathsf{C}_i, \mathsf{C}_j$ associated with VC in $\mathcal{T}$, to LAI^*. In particular, the challenger does the following for each of the clients: (1) it sets $\mathsf{past.LAI} := \mathsf{curr.LAI}$; (2) it sets $\mathsf{curr.LAI} := \mathsf{LAI}^*$. Any protocol sessions still running for VC are aborted.

- $\mathsf{Send}(\mathsf{P}, s, m) \rightarrow m'$: for the first input, the adversary can input either a handle VC or a server identity S. In the former case, the oracle simulates sending the message m from the adversary to the client associated with VC in session s, returning either the party's message m' or $\perp$ if either s is not associated with VC or if VC does not exist. Parties may also return $m' = \perp$ as an error message. If the first input of this oracle is set to S, the oracle simulates sending the message from the adversary to the server S.

- $\mathsf{Execute}(\mathsf{VC}, \mathsf{S}, s) \rightarrow \tau$: this oracle simulates a complete protocol run between the client associated with VC and the server S. in the presence of a passive adversary. In particular, Send queries to alternating the server S and the client associated with VC on genuinely-output messages, this oracle generates and outputs the transcript τ of the execution between the server S and the client C for which session s was created.

- $\mathsf{Result}(\mathsf{P}, s) \rightarrow \{0, 1\}$: if $\mathsf{P} = \mathsf{VC}$, this oracle returns a bit indicating whether the client associated with VC has accepted the server that VC ran session s with. If $\mathsf{P} = \mathsf{S}$, then the bit indicates whether the server accepted the client. For the AKA protocol, an acceptance bit is equivalent to the confirmation of the key-exchange. If the session is incomplete or session s is not associated with P, the oracle returns $\perp$.

- $\mathsf{Corrupt}(\mathsf{C}) \rightarrow \{\mathsf{sk}, \mathsf{st_C}, \mathsf{ID_C}, (\mathsf{past.LAI}, \mathsf{curr.LAI})\}$: for a client C, this oracle returns the full state (static and ephemeral), the identifiers and the location information of client C.

4.6 Two proposed variants of the UMTS-AKA Protocol

We proceed to describe two of the more promising improvements to the UMTS-AKA protocol, and show that these variants are vulnerable to client-unlinkability attacks.

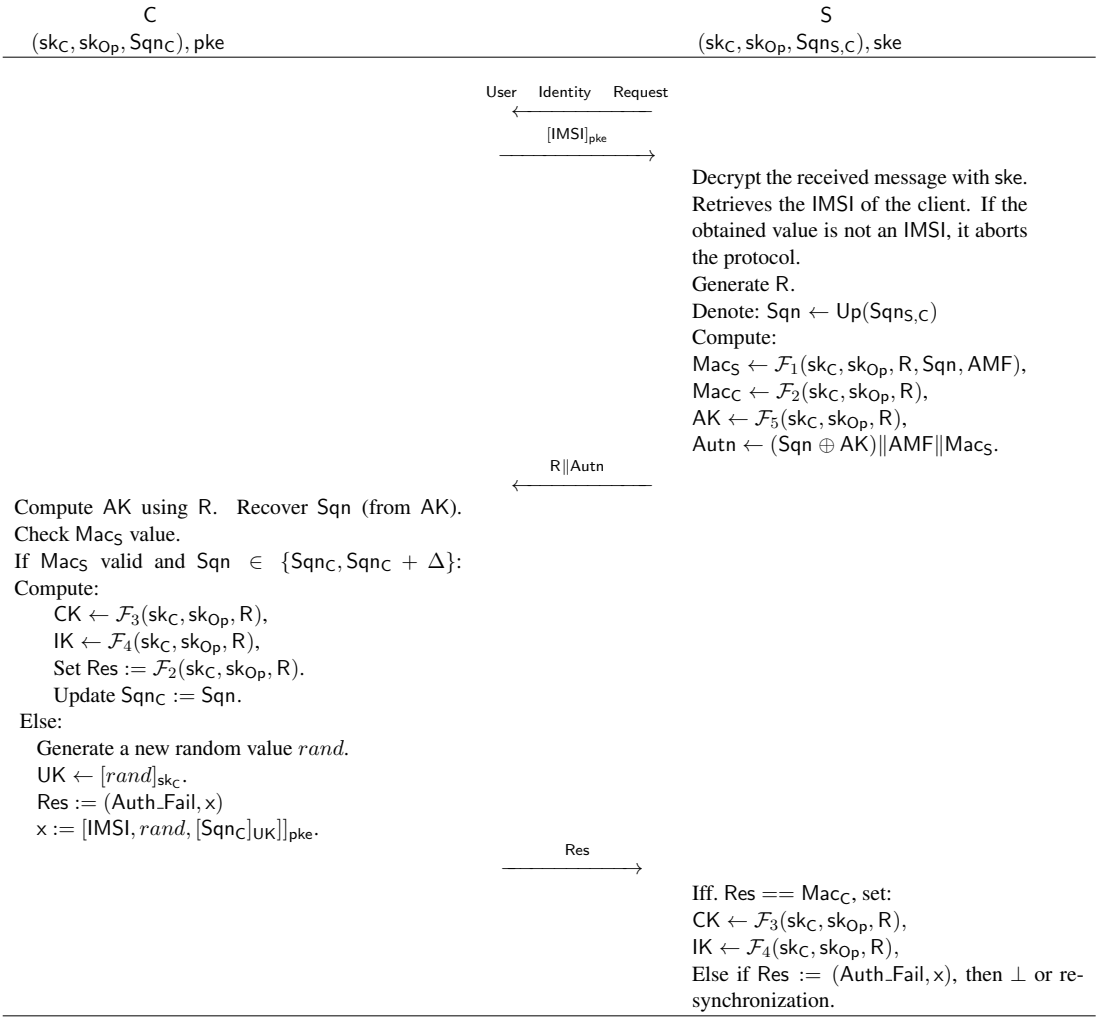


Figure 4.3: The fixed UMTS-AKA Protocol by Arapinis.

4.6.1 A first fix from Arapinis et al.

[31]

Arapinis et al. [31] propose an UMTS-AKA variant which is supposed to ensure client unlinkability by avoiding failure-message-based linking and fixing the user identification. The autors propose a simply representation of the network by concatenating the local server VLR and the client's operator HLR/AuC as an unique entity called a server S. Considering the problem of IMSI catchers, they firstly modify the identification of the user, namely addressing privacy breaches that exploit the fact that IMSIs can be sent in clear upon request. In this variant, the IMSI is encrypted with a public-key-randomized-encryption (the same as the one used for the error messages) and they remove the use of temporary identifiers. Thus, no re-allocation procedure is required. We suppose if the server cannot recover the IMSI from the encryption the protocol is aborted. Then, they propose to replace the two distinct failure messages by two indistinguishable messages (which are nonetheless intelligible to legitimate parties). The failure messages are now encrypted with a public key associated with the network and include the IMSI, a constant string Auth_Fail, a random value $rand$ and the current sequence number $\mathsf{Sqn_C}$. The latter is additionally encrypted with an unlinkability key $\mathsf{UK} = f_{\mathsf{sk_C}}(rand)$, which authenticates the error message. After decrypting this generic message, the network can deduce the cause of the failure from the IMSI and the sequence

number, and sends the appropriate answer This variant is described in details in [31].

In this variant, authors propose to fix the user identification from a public-key-infrastructure. Since the representation of the network is simply (concatenation of the local and home network), it is hard to realize the feasibility of this infrastructure. Indeed, if the local server owns the secret key ske, that implies the clients to store hundreds of certificates, notably due to roaming situations. Otherwise, the secret key is stored by the home operator. But in this case, the local server cannot decrypt the message and recover the identity of the client.

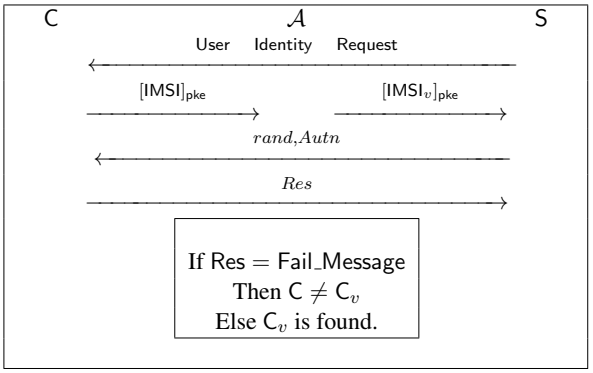


Figure 4.4: Attack on the fixed version of Arapinis.

Additionally to this practical consideration, Arapinis et al. propose a security proof of their variant in a 2-party formal model (using ProVerif), which is weaker than our 3-party computational model. Their proof idealizes the long-term state used in the protocol runs, making it unclear how far the guarantee holds for the true scheme. Moreover, we prove that the fixed protocol does not assure the untraceability of the mobile subscriber. Indeed, we propose a new attack permitting to trace a victim client based on the knowledge of a permanent user identity. We note that the IMSI values are considered as a public value. Since IMSI values follow a relatively predictable structure, this is not a strong assumption. In a specific area, our attack allows to trace a victim client C_v, whose permanent identifier IMSI_v is known by the adversary. This attack consists in replacing the answer of the user identity request, as given by the user, by a response which encrypt the permanent identifier of the victim client. In this case, if the user that was sent the identity request is truly C_v, then the adversary has perfectly simulated its behaviour and the session could succeed. Else, the user will return an error for the MAC verification. While encrypted, the message will still yield a different behaviour (notably an abort). This attack is detailed in the figure 4.4. We state the following result:

Lemma 1. *Let* Π *the protocol proposed by Arapinis et al., as presented in figure 4.3. Then there exists a* $(t, 1, 1, 0)$*-adversary* $\mathcal{A}$ *against the weak-privacy of the protocol* Π *running in time* t*, creating at most one party instance, running one user identification per each instance and making no extra query to the related internal cryptographic functions, which wins with non-negligeable advantage* $\mathsf{Adv}_{\Pi}^{\mathsf{ww\text{-}unlink}}(\mathcal{A}) = \frac{1}{2}$.

Proof. The attack we describe goes as follows:

1. At time $\mathsf{clock} = 0$, the challenger sets up the server.
2. At time $\mathsf{clock} = 1$, the adversary creates two clients C_0, C_1 from the oracles $\mathsf{CreateClient}(\mathsf{IMSI}^{\{0\}}, \mathsf{st}_0^{\{0\}})$ and $\mathsf{CreateClient}(\mathsf{IMSI}^{\{1\}}, \mathsf{st}_0^{\{1\}})$.
3. At time $\mathsf{clock} = 2 \rightarrow 3$, the adversary uses the oracle $\mathsf{DrawClient}(\mathsf{C}_0, \mathsf{C}_1)$ which returns at time $\mathsf{clock} = 3$ a virtual client $vc = \mathsf{C}_b$ following a chosen bit b.

4. At time $\mathsf{clock} = 4 \rightarrow 10$, the adversary runs an instance of the protocol Π between the virtual client and the terminal as follows: At time $\mathsf{clock} = 4$, the terminal sends a user identity request to the virtual client; at time $\mathsf{clock} = 5$, the virtual client answers with the value UID such as: $\mathsf{UID} = [\mathsf{IMSI}^{\{b\}}]_{\mathsf{pke}_b}$; at time $\mathsf{clock} = 6$, the adversary intercepts this message (before arriving to the terminal) and forges a user identification $\mathsf{UID} = [\mathsf{IMSI}^{\{0\}}]_{\mathsf{pke}_0}$ for the client C_0 which will be sent to the terminal at time $\mathsf{clock} = 7$ (instead of the user identifier computed by the virtual client); at time $\mathsf{clock} = 8$, the terminal receives the related user identifier and generates an authentication challenge which is sent to the virtual client vc at time $\mathsf{clock} = 9$; this latter sends its authentication answer Res at time $\mathsf{clock} = 10$ which is eavesdropped by the adversary.

5. At time $\mathsf{clock} = 11$, since the eavesdropped authentication answer Res, the adversary tries to guess the bit b as follows: if the message Res contains a failure message, then the adversary deduces the virtual client to be client C_0 (setting $b' = 0$). Otherwise, it assumes the virtual client to be C_1 (setting $b' = 1$).

6. At time $\mathsf{clock} = 12$, the adversary returns its guess b'.

It is clear that the success probability of a such adversary is 1. Thus, the protocol cannot guarantee the weak-privacy. We note that in practice if the adversary $\mathcal{A}$ cannot detect de-synchronizations, some "false positives" can be appear: indeed, during the tested session, if the victim client is de-synchronized, the adversary cannot. However, frequency of such an event (i.e this one of a synchronization procedure) is very weak. Moreover, the same attack procedure, if repeating, will reduce the probability of obtaining a false positive. □

4.6.2 A variant from Van der Broek et al.

[104] Van den Broek et al. [104] recently proposed an IMSI catcher countermeasure; in this improved variant, they avoid sending the IMSI in clear by replacing (IMSI, TMSI) tuples by an upgradeable pseudonym denoted PMSI. Their modified identification phase is done exclusively by means of these pseudonyms. The PMSI is chosen by the operator and sent with the authentication challenge in the preparation phase, encrypted together with the sequence number with a new secret key that is assumed to be shared by clients and their operators. The ciphertext is then used as the random value R in the authentication challenge. Indeed, a successful session of the UMTS-AKA protocol, ending in the establishment of new session keys, can only be attained if the PMSI is correctly updated. This variant is described in detail in [104].

From a practical point of view, using the operator at each key-exchange session is costly, and something that the original AKA was designed tries to avoid. Furthermore, though this variant successfully prevents IMSI catchers, it does not fully address client unlinkability. The pseudonym PMSI can be intercepted in one session; if this session is then aborted, the PMSI can be replayed in a second session, thus leading to user linkability. Furthermore, the protocol is still vulnerable to the attack based on linking failure messages, as presented by Arapinis et al. Thus, if Π denotes the protocol proposed by van den Broek et al., it holds that:

Lemma 2. *There exists a $(t, 2, 1, 0)$-adversary $\mathcal{A}$ against the narrow-weak-client-unlinkability of Π running in time t, initiating two protocol sessions, and making no query to the internal cryptographic function G, which has an advantage $\mathsf{Adv}_{\Pi}^{\mathsf{ww\text{-}unlink}}(\mathcal{A}) = \frac{1}{2}$ (and a probability of 1) to win the game.*

Proof. The attack we describe follows the strategy of Arapinis et al. for the linkability of failure messages:

1. At time $\mathsf{clock} = 0$, the challenger sets up the server.

2. At time $\mathsf{clock} = 1$, the adversary creates two clients C_0, C_1 from the oracles $\mathsf{CreateClient}(\mathsf{IMSI}^{\{0\}}, \mathsf{st}_0^{\{0\}})$ and $\mathsf{CreateClient}(\mathsf{IMSI}^{\{1\}}, \mathsf{st}_0^{\{1\}})$.

3. At time $\mathsf{clock} = 2 \rightarrow 3$, the adversary uses the oracle $\mathsf{DrawClient}(\mathsf{C_0}, \mathsf{C_1})$ which returns at time $\mathsf{clock} = 3$ a virtual client $vc = \mathsf{C}_b$ following a chosen bit b.

4. At time $\mathsf{clock} = 4 \rightarrow 5$, the adversary runs an instance of the protocol Π between the virtual client and the terminal via the oracle $\mathsf{Execute}(\mathsf{vc}, \mathsf{s})$ which returns the transcript of an instance s at time $\mathsf{clock} = 5$. This transcript includes the authentication challenge denoted $(\mathsf{R}, \mathsf{Autn})$.

5. At time $\mathsf{clock} = 6 \rightarrow 12$, the adversary runs an instance $s + 1$ of the protocol Π between the virtual client and the terminal as follows: at time $\mathsf{clock} = 6$, the terminal sends a user identity request to the virtual client; at time $\mathsf{clock} = 7$, the virtual client sends the user identifier UID to the terminal; at time $\mathsf{clock} = 8$, the terminal receives the related user identifier and generates an authentication challenge which is sent to the virtual client vc; at time $\mathsf{clock} = 9$, the adversary intercepts the fresh authentication challenge (before arriving to the virtual client) and sends the previous authentication challenge $(\mathsf{R}, \mathsf{Autn})$ to the virtual client at $\mathsf{clock} = 10$ (obtains in a previous session on the client $\mathsf{C}_0 := \mathsf{DrawClient}(\mathsf{C_0}, \mathsf{C_0})$); at time $\mathsf{clock} = 11$, the virtual client sends its authentication answer Res at $\mathsf{clock} = 12$ which is eavesdropped by the adversary.

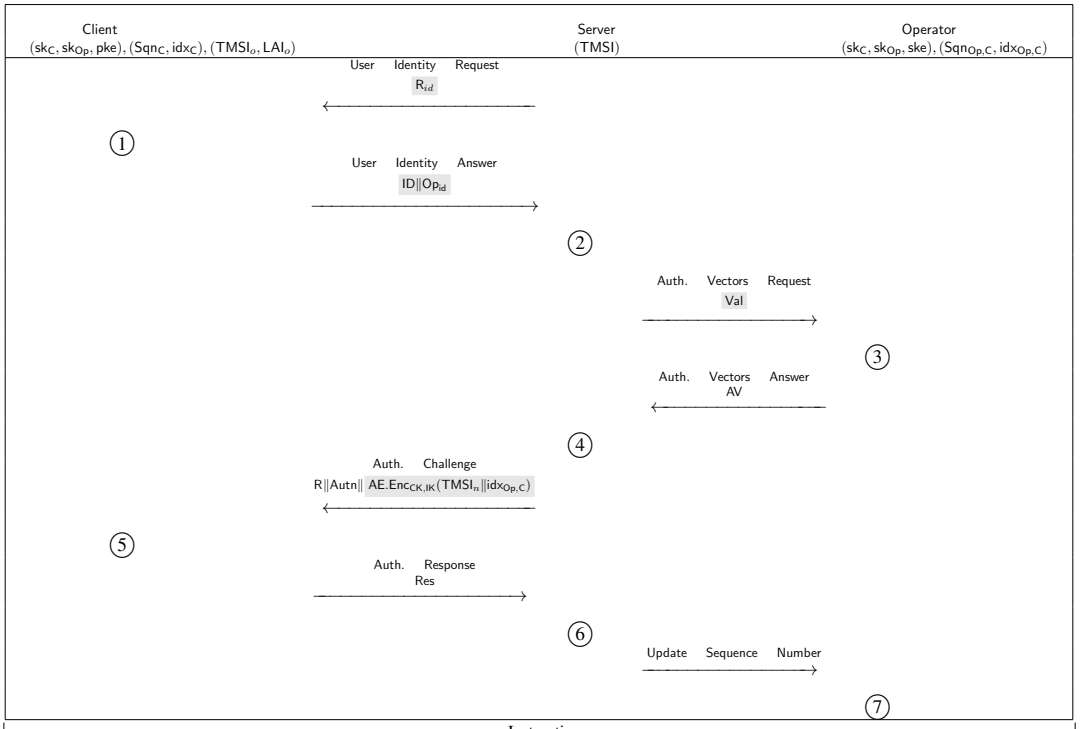
6. At time $\mathsf{clock} = 13$, since the eavesdropped authentication answer Res, the adversary tries to guess the bit b as follows: if the message Res contains a "de-synchronization message", then the adversary considers the virtual client as the client C_0 (that implies $b' = 0$). Otherwise, it considers the virtual client as the client C_1 (that implies $b' = 1$).

7. At time $\mathsf{clock} = 14$, the adversary returns its guess b'.

□

4.7 Our Complete Proposal: PrivAKA

In this section, we propose a new variant of the UMTS-AKA protocol which guarantees the weak-unlinkability, the key-indistinguishability properties and the security properties (client- and server-impersonations resistance, soundness and state-confidentiality). The weak-unlinkability, as defined in the privacy model in Section 4.5, is a property guaranteeing that two sessions run by the same client are not linkable with respect to a MitM attacker which may learn whether the server accepts the client's authentication or not, but which cannot corrupt clients to learn their keys. The other properties are defined as in our 3-party security model defined in Section 3.2. This privacy analysis notably is established with the respect to malicious servers, and considering the three-party features as an important part of the analysis. Our fixed variant takes into account practical considerations (interoperability of the servers, USIM card restrictions, ...) for a possible standardization. We note that a fix variant of the EPS-AKA protocol can be proposed to guarantee all the security and privacy requirements. This variant is really similar to the one proposed in 4.7.1 including a new temporary identifier which does not reveal the user location and the 4G session keys hierarchy. The related security and privacy properties can be proved similarly and this variant will keep all the modular properties including all the described practical considerations.



Instructions:

Client	Server	Operator
①: Compute the identifier: If $flag_{Id_{temp}} := 0$ then $ID = TMSI$. Else, $ID = PKE.Enc_{pke}(f_5(keys, R_{id}, IMSI, idx_C) \| R_{id} \| IMSI \| idx_C)$. $flag_{Id_{temp}} := 1$.		
⑤: Compute AK using R. Recover $Sqn_{Op,C}$ (from AK). Check Mac_S value. Compute: IK, CK; Retrieve the received index and the new TMSI. If abort caused or the AE does not verify, set $flag_{Id_{temp}} := 1$ and increment: $idx_C := idx_C + 1$. Else, check validity of $Sqn_{Op,C}$, i.e if one of the following conditions is correct: • $Sqn_C = Sqn_{Op,C}$. • $Sqn_C = inc(Sqn_{Op,C})$ and $idx = idx_C + 1$. If the first condition is accepted: reset the index idx_C, update the sequence number $Sqn_C = inc(Sqn_C)$. If the second condition is accepted: $idx_C = idx_C + 1$. Compute $Res := \mathcal{F}_1^*(keys, R, Sqn_{Op,C}, Id_S, AMF)$. Update the internal index. Allocate the new TMSI. $flag_{Id_{temp}} := 0$.	②: Process the identifier ID: If the identifier is a TMSI then $Val = IMSI$. Otherwise, $Val = (ID, R_{id})$. ④: Store AV. Then, it sends the authentication challenge and the new couple $(TMSI_n, idx_{Op,C})$ encrypted and authenticated by the session keys. ⑥: If the authentication of the client is verified ($Res \stackrel{?}{=} Mac_C$), then they ask to the server the update of its sequence number. Otherwise, the protocol is aborted.	③: Verify the identity of the client with Val. If this holds, retrieve idx_C, set $idx_{Op,C} := idx_C$ Generate R. Denote: $keys := (sk_C, sk_{Op})$. $Mac_S \leftarrow \mathcal{F}_1(keys, R, Sqn_{Op,C}, Id_S, AMF)$, $Mac_C \leftarrow \mathcal{F}_1^*(keys, R, Sqn_{Op,C}, Id_S, AMF)$, $CK \leftarrow \mathcal{F}_3(keys, R, Sqn_{Op,C}, Id_S, AMF)$, $IK \leftarrow \mathcal{F}_4(keys, R, Sqn_{Op,C}, Id_S, AMF)$, $AK \leftarrow \mathcal{F}_5(keys, R, Id_S)$, $Autn \leftarrow (Sqn_{Op,C} \oplus AK) \| AMF \| Mac_S$, $Sqn_{Op,C} \leftarrow inc(Sqn_{Op,C})$, $AV := (R, CK, IK, Autn, Mac_C, idx_{Op,C})$. End for. ⑦: Update the sequence number: $Sqn_{Op,C} \leftarrow inc(Sqn_{Op,C})$. Reset the index $idx_{Op,C}$.

Figure 4.5: Our fixed UMTS-AKA Procedure: PrivAKA.

4.7.1 Description

Instead of five phases, our variant only consists of three. Our protocol is designed to not require re-synchronization (which was phase 4 in AKA), and we include the TMSI reallocation (which was phase 5 in AKA) as part of the key-exchange phase (phase 3). In our construction, we use a public-key encryption scheme PKE= (PKE.KGen,PKE.Enc,PKE.Dec), such that each operator has a certified public and secret key-pair denoted as (pke_{Op}, ske_{Op}). We assume that the client stores only its own operator's public key (and its certificate) internally. In particular, we do not

give encryption keys to the servers in order to minimize key-management issues. We also use a secure authenticated encryption scheme $\mathsf{AE} = (\mathsf{AE.KGen}, \mathsf{AE.Enc}, \mathsf{AE.Dec})$. Though these can be instantiated generically, we use **AES-GCM** [86] for the AE scheme and (EC)IES [100] for the PKE scheme. We depict our variant in Figure 4.5, and indicate in the grey boxes the differences to the classical UMTS-AKA procedure. Just as the original protocol, our variant starts with an *identification* phase, run by the client and the server over the insecure channel. The server sends an identification request which includes a new random value that we denote R_{id}.

The client computes a user identification answer following a flag $\mathsf{flag}_{\mathsf{Id}_{\mathsf{temp}}}$ which it manages: the client uses either a pre-exchanged fresh temporary identifier TMSI (if the previous protocol has been accepted and if the user stayed in the same area: $\mathsf{flag}_{\mathsf{Id}_{\mathsf{temp}}} = 1$) or it sends a public-key encryption of the concatenation of the received random value R_{id} and the evaluation of the function $\mathcal{F}_5$ on input the client secret key, the operator's secret key, the random value R_{id} and the client's IMSI (only if it does not own a fresh temporary identifier or if the user is located in a new area: $\mathsf{flag}_{\mathsf{Id}_{\mathsf{temp}}} = 0$). We demand that the output size of the PKE scheme for this input size is always equal to the length of the TMSI. The value flag $\mathsf{flag}_{\mathsf{Id}_{\mathsf{temp}}}$ is only used to restrict the computation of an encrypted permanent identity and not for security reasons. In both cases, the client also appends the identity of the operator Op it subscribes to, to the message. Moreover, we assume that the client can detect when it moves in a new area. In this case, it allocates the flag at 0.

Intuitively, if the client stayed in the same area ($\mathsf{flag}_{\mathsf{Id}_{\mathsf{temp}}} = 0$), then the TMSI the client stored came from the server it is currently communicating with, hence the server can find the $(\mathsf{TMSI}, \mathsf{IMSI})$ association in its database. Otherwise, the client does not use its TMSI value, but rather encrypts a function of the IMSI with the operator's public key. The function is symmetric-key, requiring knowledge of the client and operator keys, and it is not replayable due to the fresh identification randomness R_{id}. Upon receiving a string of the form (m, Op), the server first checks whether the message m is a TMSI present in its database; if so, it retrieves the IMSI to which this value corresponds; else, it assumes that m is a ciphertext, and it sends it to the operator Op for decryption.

Phase 2, *preparation*, is run over a secure channel, between the server and the operator. If the server received a valid TMSI in the previous phase, the preparation phase begins with the server sending the corresponding IMSI to the operator; else, the server forwards the received ciphertext and the associated random value R_{id}. The operator proceeds similarly to the standard AKA preparation procedure, with the following differences:

- We add as input to each cryptographic function a server-specific value Id_{S} for a server with identity $\mathsf{S} = \mathsf{VLR}$. This is to prevent attacks in which the adversary replays authentication vectors from one network to another, as presented by Zhang [109]. We also use the constant AMF which is sent in clear, as an additional input.

- We add the sequence number $\mathsf{Sqn}_{\mathsf{Op},\mathsf{C}}$ to each of the cryptographic functions apart from $\mathcal{F}_5$. Since the sequence number is an ephemeral value, which is updated, this guarantees freshness even if the randomness R is repeated.

- We introduce an index value $\mathsf{idx}_{\mathsf{Op},\mathsf{C}}$ which essentially prevents the repetition of a challenge using the same sequence number. This value is essential in preventing a desynchronization of sequence number values. We note that the client also keeps track of a similar index $\mathsf{idx}_{\mathsf{C}}$, which will play a role in the key-exchange phase, as detailed below.

- We reduce the batch of authentication vectors to only one element. We note that such reduction is already present when the client is in a roaming situation. 3GPP specifications only allow to send more than one authentication vector if the client is located in its home network.

In the final phase, *authenticated-key-exchange*, which is run between the client and the server, the server sends a random value R, the authentication string Autn, and an authenticated encryption of the new TMSI, using the keys (CK, IK) derived for this session. The client proceeds similarly to the original AKA procedure, recovering AK by using the random value R, then checking $\mathsf{Mac_S}$. If the authentication cannot be verified, the procedure is aborted, and the new TMSI is disregarded. Else, the user computes CK, IK and decrypts the received authenticated encryption string to find the TMSI value and the operator's index $\mathsf{idx_{Op,C}}$. Then, C checks the freshness of the sequence number, i.e, it verifies if one of the following two conditions is correct:

$$\mathsf{Sqn_C} = \mathsf{Sqn}^{\{i\}}; \quad \mathsf{Sqn_C} = \mathsf{inc}(\mathsf{Sqn}^{\{i\}}) \text{ and } \mathsf{idx_{Op,C}} = \mathsf{idx_C} + 1.$$

If the protocol is run normally, the first of these conditions is the one that will hold. However, if the previous session is aborted after receiving the server's authentication challenge, then the two sequence numbers may become desynchronized by one step (the second condition). Further desynchronization is prevented by the use of an index, which indicates whether the authentication string for a particular $\mathsf{Sqn_{Op,C}}$ has already been used or not. If the first condition holds, then the client's internal index is reset; else, the index is incremented by 1. The client updates the sequence number only upon successful authentication. If none of these conditions are verified, the procedure is aborted and does not use of a resynchronization procedure.

Finally, the user computes a response $\mathsf{Res} := \mathcal{F}_2(\mathsf{keys}, \mathsf{R}^{\{i\}}, \mathsf{Sqn}^{\{i\}}, \mathsf{Id_S}, \mathsf{AMF})$, sends Res, then stores the TMSI and the new index value. The server checks Res against the prepared value $\mathsf{Mac_C}$ (else, if no response is received, the procedure is aborted).

One notable exception to the original UMTS-AKA protocol is that whenever an abort occurs on the server's side, the second phase – *preparation* – is used instead of simply querying the next vector in the prepared batch. Though this might seem more inefficient, we note that an abort only occurs in the presence of an adversary, which is considered to be a rare event. We detail the procedure upon aborts in Figure 4.6.

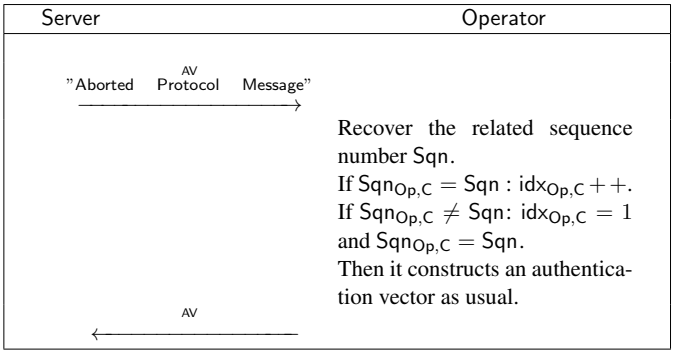


Figure 4.6: Procedure after an abort.

Internal cryptographic algorithms: In our variant, we have modified the inputs of the internal cryptographic algorithms, notably to include the sequence number and the new value $\mathsf{Id_S}$. Thus, we need to provide an update of these algorithms to consider these modifications. As specified in specifications, the UMTS-AKA protocol can be based on two different sets of algorithms: TUAK and $\mathsf{MILENAGE}$.

To preserve backwards compatibility, we propose to keep and update these two algorithm suites. Moreover, our variant requires an algorithm of authenticated encryption. We propose the use of the well-known **AES-GCM** standard [86]. It is denoted AE. Note that for our variant, we do not need the functions $\mathcal{F}_1^*$ and $\mathcal{F}_5^*$, since we have dropped the resynchronization procedure.

The five internal cryptographic functions take as input the following values:

- keys: the couple of the both 128-bit (or 256-bit) keys: the subscriber key $\mathsf{sk_C}$ and the operator key $\mathsf{sk_{Op}}$.
- Sqn (used in all but $\mathcal{F}_5$): a 48-bit sequence number.
- AMF (used in all but $\mathcal{F}_5$): a 16-bit authentication field management.
- R: a 128-bit random value.
- $\mathsf{Id_S}$: a 128-bit public value characterizing the visited network.

Update of the MILENAGE **algorithms:** MILENAGE is the original set of algorithms which is currently implemented as detailed the specification 35.206 [2].

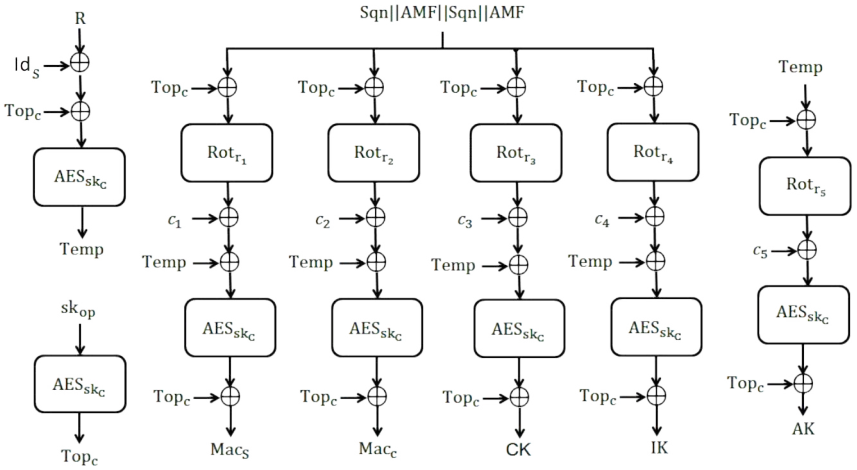


Figure 4.7: Updated Milenage.

To assure a stronger security, we modify the MILENAGE algorithms to output 128-bit MAC values and session keys CK and IK.

Based on the Advanced Encryption Standard (AES), these functions compute firstly both values Top_C and Temp as follows:

$$\mathsf{Top}_C = \mathsf{sk_{Op}} \oplus \mathsf{AES_{sk_C}}(\mathsf{sk_{Op}}), \quad \mathsf{Temp} = \mathsf{AES_{sk_C}}(\mathsf{R} \oplus \mathsf{Top}_C \oplus \mathsf{Id_S}).$$

Then, we obtain the output of the five functions as follows:

- **Output** $\mathcal{F}_1$**:** $\mathsf{Mac_S} = \mathsf{AES_{sk_C}}(\mathsf{Temp} \oplus \mathsf{Rot_{r_1}}(\mathsf{Sqn}\|\mathsf{AMF}\|\mathsf{Sqn}\|\mathsf{AMF}) \oplus \mathsf{c}_1) \oplus \mathsf{Top}_C$,
- **Output** $\mathcal{F}_2$**:** $\mathsf{Mac_C} = \mathsf{AES_{sk_C}}(\mathsf{Temp} \oplus \mathsf{Rot_{r_2}}(\mathsf{Sqn}\|\mathsf{AMF}\|\mathsf{Sqn}\|\mathsf{AMF}) \oplus \mathsf{c}_2) \oplus \mathsf{Top}_C$,
- **Output** $\mathcal{F}_3$**:** $\mathsf{CK} = \mathsf{AES_{sk_C}}(\mathsf{Temp} \oplus \mathsf{Rot_{r_3}}(\mathsf{Sqn}\|\mathsf{AMF}\|\mathsf{Sqn}\|\mathsf{AMF}) \oplus \mathsf{c}_3) \oplus \mathsf{Top}_C$,
- **Output** $\mathcal{F}_4$**:** $\mathsf{IK} = \mathsf{AES_{sk_C}}(\mathsf{Temp} \oplus \mathsf{Rot_{r_4}}(\mathsf{Sqn}\|\mathsf{AMF}\|\mathsf{Sqn}\|\mathsf{AMF}) \oplus \mathsf{c}_4) \oplus \mathsf{Top}_C$,
- **Output** $\mathcal{F}_5$**:** $\mathsf{AK} = \lfloor \mathsf{AES_{sk_C}}(\mathsf{Rot_{r_5}}(\mathsf{Temp} \oplus \mathsf{Top}_C) \oplus \mathsf{c}_5) \oplus \mathsf{Top}_C \rfloor_{0..47}$,

with the five integers $\mathsf{r}_1 = 0$, $\mathsf{r}_2 = 24$, $\mathsf{r}_3 = 48$, $\mathsf{r}_4 = 64$ and $\mathsf{r}_5 = 96$ in the range $\{0, 127\}$, which define the number of positions the intermediate variables are cyclically rotated by the right, and the five 128-bit constants c_i follows as:

- $\mathsf{c}_1[i] = 0, \forall i \in \{0, 127\}$.
- $\mathsf{c}_2[i] = 0, \forall i \in \{0, 127\}$, except that $\mathsf{c}_2[127] = 1$.

- $\mathsf{c}_3[i] = 0, \forall i \in \{0, 127\}$, except that $\mathsf{c}_3[126] = 1$.
- $\mathsf{c}_4[i] = 0, \forall i \in \{0, 127\}$, except that $\mathsf{c}_4[125] = 1$.
- $\mathsf{c}_5[i] = 0, \forall i \in \{0, 127\}$, except that $\mathsf{c}_5[124] = 1$.

This is also described in Figure 4.7.

Update of the TUAK **algorithms:** TUAK is an alternative set of MILENAGE based on the internal permutation of the Keccak [44]. The specification TS 35.231 [4] details the internal algorithms of this set. We update these algorithms by only modifying the inputs of the second permutation.

As the original TUAK algorithms, we firstly compute the value Top_C as follows:

$$\mathsf{Top}_C = \lfloor \mathsf{f}_{\mathsf{Keccak}}(\mathsf{sk}_{\mathsf{Op}} \| \mathsf{Inst} \| \mathsf{AN} \| 0^{192} \| \mathsf{Key} \| \mathsf{Pad} \| 1 \| 0^{512}) \rfloor_{1..256},$$

We note that the values AN, Inst', Inst, Pad are the same as used in the no-updated TUAK algorithms and Key the (padded) subscriber key.

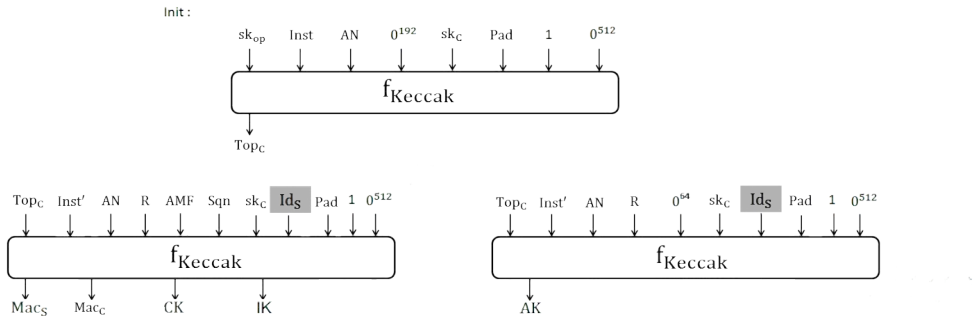


Figure 4.8: Updated TUAK

At this point, the behaviour of $\mathcal{F}_5$ diverges from that of the other functions. To generate output of $\mathcal{F}_5$, we compute the value Val_1 and while for $\mathcal{F}_1$ and $\mathcal{F}_4$, we compute the value Val_2, as follows:

$$\mathsf{Val}_1 = \mathsf{f}_{\mathsf{Keccak}}(\mathsf{Top}_C \| \mathsf{Inst}' \| \mathsf{AN} \| \mathsf{R} \| 0^{64} \| \mathsf{Key} \| \mathsf{Id}_\mathsf{S} \| \mathsf{Pad} \| 10^{512}),$$
$$\mathsf{Val}_2 = \mathsf{f}_{\mathsf{Keccak}}(\mathsf{Top}_C \| \mathsf{Inst}' \| \mathsf{AN} \| \mathsf{R} \| \mathsf{AMF} \| \mathsf{Sqn} \| \mathsf{Key} \| \mathsf{Id}_\mathsf{S} \| \mathsf{Pad} \| 10^{512}).$$

Then, we obtain the output of the five functions truncating the related value as follows:

- **Output** $\mathcal{F}_1$**:** $\mathsf{Mac}_\mathsf{S} = \lfloor \mathsf{Val}_2 \rfloor_{0..127}$,
- **Output** $\mathcal{F}_2$**:** $\mathsf{Mac}_\mathsf{C} = \lfloor \mathsf{Val}_2 \rfloor_{256..383}$,
- **Output** $\mathcal{F}_3$**:** $\mathsf{CK} = \lfloor \mathsf{Val}_2 \rfloor_{512..639}$,
- **Output** $\mathcal{F}_4$**:** $\mathsf{IK} = \lfloor \mathsf{Val}_2 \rfloor_{768..895}$,
- **Output** $\mathcal{F}_5$**:** $\mathsf{AK} = \lfloor \mathsf{Val}_1 \rfloor_{0..47}$.

This is also depicted in Figure 4.8.

We note that the multi-output property is, as the original version, not an issue for the security of the master key, since during one session we can have as many as four calls to the same function with similar inputs (and a different truncation).

4.7.2 Security and Privacy Proofs

In this section, we prove the weak-client-unlinkability, the strong-key-indistinguishability, strong-server- and client-impersonation, soundness, and state-confidentiality of PrivAKA under some classic assumptions of the security of internal cryptographic functions. Just as for our security analysis, we generalized the algorithms of TUAK and MILENAGE as two functions G and G^* as detailed in Section 4.7.4.

Weak Client Unlinkability of our fixed variant.

In this section, we prove that PrivAKA protocol achieves weak client unlinkability if we assume that the internal functions assures indistinguishability, pseudorandomness and unforgeability properties. The weak-client-unlinkability resistance of our fixed variant is proved as follows:

Theorem 19. [ww-unlink − Resistance.] *Let* $G_{:}\{0,1\}^{\kappa} * \{0,1\}^{d} * \{0,1\}^{t} * \{0,1\}^{t} \rightarrow \{0,1\}^{n}$ *be our unitary function generalizing the internal cryptographic functions* $\mathcal{F}_i$ *and* Π *our PrivAKA protocol described in Section 4.7.1. Consider a* $(t, q_{\mathsf{exec}}, q_{\mathsf{id}}, q_G, q_{\mathsf{AE}}, q_{\mathsf{PKE}})$*-adversary* $\mathcal{A}$ *against the weak privacy* ww-unlink*-security of the protocol* Π *running in time t, creating at most* q_{exec} *party instance, with at most* q_{id} *user identification per instance and making at most* q_G, q_{AE}, q_{PKE} *queries to respectively the functions G,* AE *and* PKE. *The advantage of this adversary is denoted* $\mathsf{Adv}^{\mathsf{ww\text{-}unlink}}_{\Pi}(\mathcal{A})$. *Then, there exist* $(t' \sim O(t), q' = q_{\mathsf{sess}} + q_G)$*-adversary* $\mathcal{A}_1$ *and* $(t' \sim O(t), q' = 2 \cdot q_{\mathsf{sess}} + q_G)$*-*$\mathcal{A}_2$ *against respectively the unforgeability and pseudorandomness of the function G, an* $(t'' \sim O(t), q'' = q_{\mathsf{sess}} + q_{\mathsf{AE}})$*-adversary* $\mathcal{A}_3$ *against the AE-security of the function* AE, $(t''' \sim O(t), q''' = q_{\mathsf{sess}} \cdot q_{\mathsf{id}} + q_{\mathsf{PKE}})$*-adversaries* $\mathcal{A}_4$ *against the indistinguishability of the function* PKE, $(t, q_{\mathsf{exec}}, q_{\mathsf{id}}, 0, 0, q_G, q_{\mathsf{AE}}, q_{\mathsf{PKE}})$*-adversary* $\mathcal{A}_5$ *against the key-indistinguishability of the protocol* Π *such that:*

$$\mathsf{Adv}^{\mathsf{ww\text{-}unlink}}_{\Pi}(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{K.Ind}}_{\Pi}(\mathcal{A}_5) + \frac{1 + (q_{\mathsf{exec}} \cdot q_{\mathsf{id}})^2}{2^{|\mathsf{TMSI}|}} + \frac{q^2_{\mathsf{exec}}}{2^{|\mathsf{R}|}} + \frac{(q_{\mathsf{exec}} \cdot q_{\mathsf{id}})^2}{2^{|\mathsf{R}_{id}|}} + \mathsf{n_C} \cdot (3 \cdot \mathsf{Adv}^{\mathsf{mac}}_{G}(\mathcal{A}_1) + \mathsf{Adv}^{\mathsf{prf}}_{G}(\mathcal{A}_2) + 4 \cdot \mathsf{Adv}^{\mathsf{ae}}_{\mathsf{AE}}(\mathcal{A}_3) + \mathsf{Adv}^{\mathsf{ind-cca2}}_{\mathsf{PKE}}(\mathcal{A}_4)).$$

Proof. Our proof has the following hops:
Game $\mathbb{G}_0$**:** This game works as the ww-unlink-game stipulated in section 4.5.
Game $\mathbb{G}_1$**:** We modify the original game $\mathbb{G}_0$ to ensure that the random values R_{id} and R used by honest server instances are always unique. The related security loss due to the collisions between each respective random in two different instances is given by the following expression:

$$\left| \Pr[\mathcal{A}_{\mathbb{G}_0}\mathsf{wins}] - \Pr[\mathcal{A}_{\mathbb{G}_1}\mathsf{wins}] \right| \leq \frac{q^2_{\mathsf{exec}}}{2^{|\mathsf{R}|}} + \frac{(q_{\mathsf{exec}} \cdot q_{\mathsf{id}})^2}{2^{|\mathsf{R}_{id}|}}.$$

Game $\mathbb{G}_2$**:** The game $\mathbb{G}_2$ behaves as $\mathbb{G}_1$ with the restriction that use abort the protocol if one of the three following events happen:

- **Event 1:** The adversary $\mathcal{A}$ forges successfully user identification answer against an honest server.
- **Event 2:** The adversary $\mathcal{A}$ forges successfully an authentication challenge against an honest client.
- **Event 3:** The adversary $\mathcal{A}$ forges successfully an authentication response against an honest server.

Note that if any of three events occurs, a classic attack permits to recover the user identity of the client. Such an adversary behaves as follows: the adversary submits of one of these clients (C_0 or C_1) to be chosen as the virtual client VC and forges a message x on behalf. During the session, it drops the true message from the honest server and sends the forged message. If with this modification the session is accepted, then the client for which $\mathcal{A}$ forged the message is the one chosen as the user equipment. Otherwise, the other client is the good one.

Now, focus on each event. Firstly, if the adversary wants to forge a fresh user identification response (event 1), he has two options: either guess the fresh temporary identifier TMSI or an encrypted version of the permanent identifier related to the random value R_{id} included in the user identification request. For the first option, either the adversary must obtain a new one by using the previous ones or it must decrypt the authenticated encryption which hides the TMSI in the previous session. The TMSIs are independently chosen, so they are no way to guess the fresh TMSI from the old one, except randomly, or by replaying one of these previous TMSI. The probability of a such success is at most $(1 + (q_{\mathsf{exec}} \cdot q_{\mathsf{id}})^2)/2^{|\mathsf{TMSI}|}$. The ability to recover the fresh TMSI from the encrypted version is restricted by the security of the authenticated encryption and the key-indistinguishability of the session keys. Indeed, the fresh TMSI is sent authenticated and encrypted by the algorithm AE, which is based on session keys and not long-term keys. Moreover, we note that the fresh TMSI is encrypted with the index which is predictable, but fresh for each replay of an old sequence number. So, if an adversary can forge a correct output of the authenticated encryption AE (i.e. the related input includes a correct index), it can impose its own temporary value and use it for the next user identification. The probability of such a recovery is at most $\mathsf{Adv}_{\Pi}^{\mathsf{K.Ind}}(\mathcal{A}) + \mathsf{n_C} \cdot (2 \cdot \mathsf{Adv}_{\mathsf{AE}}^{\mathsf{ae}}(\mathcal{A}))$. So, the adversary's ability to forge a correct encrypted permanent identity is restricted by his ability to forge a correct output of the function G with the IMSI and without the private key. Thus, an adversary may only succeed with probability $\mathsf{n_C} \cdot \mathsf{Adv}_{G}^{\mathsf{mac}}(\mathcal{A})$. The security loss due to the ability to forge an fresh identification response is consequently bounded by $\frac{1+(q_{\mathsf{exec}} \cdot q_{\mathsf{id}})^2}{2^{|\mathsf{TMSI}|}} + \mathsf{Adv}_{\Pi}^{\mathsf{K.Ind}}(\mathcal{A}) + \mathsf{n_C} \cdot (2 \cdot \mathsf{Adv}_{\mathsf{AE}}^{\mathsf{ae}}(\mathcal{A}) + \mathsf{Adv}_{G}^{\mathsf{mac}}(\mathcal{A}))$.

Now, focus on the second event involving a forgery on an authentication challenge. We recall that such a challenge is split into four parts: the random value R, a masked value of the current sequence number, $val = \mathsf{AK} \oplus \mathsf{Sqn}$, a message authentication code mac generated by the function G which takes in inputs the private keys keys, the random value R and the sequence number Sqn, and a tuple consisting of the next temporary identifier and the current index, both values encrypted by an authenticated encryption AE. In a fresh instance, we have two options to forge an authentication challenge: either the adversary guesses a fresh authentication challenge based on the current sequence number or it replays an old challenge based on a previous used sequence number. For the first option, the complexity to forge a such challenge is restricted by the unforgeability of the function G. Indeed even if the adversary knows the current sequence number, which is in practice masked by a one-time-pad, it cannot forge a fresh message authentication code without the related message authentication code. Moreover, we note that the index is only implied for the second condition of the freshness verification and the new TMSI will be only used for the next session. With the second condition, the best option for the adversary consists of replaying the three first parts of the previous authentication challenge from an aborted session while trying to forge an authenticated and encrypted version of a fresh index value and a fresh TMSI. Indeed, when the protocol is aborted after the server has sent the authentication challenge, the next authentication challenge is based on the same sequence number. To forge a fresh authenticated and encrypted value, the adversary chooses any non-nil value for the index value. We note that the adversaeary needs to know it because such a challenge including the index idx is considered as fresh only if there are exactly idx previous sessions aborted by a drop of the related authentication response. Moreover, we do not need to the chosen temporary value. The ability to forge a such challenge is restricted by the security of the chosen algorithm of authenticated encryption. So, considering

both conditions, the success probability is at most $\mathsf{n_C} \cdot (\mathsf{Adv}^{\mathsf{mac}}_{G}(\mathcal{A}) + \mathsf{Adv}^{\mathsf{ae}}_{\mathsf{AE}}(\mathcal{A}))$.

Finally, we focus on the third event in which $\mathcal{A}$ must forge an authentication response, which is only consists a value Res, output by the function G. Thus, the success probability is at most $n_c \cdot \mathsf{Adv}^{\mathsf{mac}}_{G}(\mathcal{A})$. We obtain a total loss of:

$$\left| \Pr[\mathcal{A}_{\mathbb{G}_1}\mathsf{wins}] - \Pr[\mathcal{A}_{\mathbb{G}_2}\mathsf{wins}] \right| \leq \frac{1 + (q_{\mathsf{exec}} \cdot q_{\mathsf{id}})^2}{2^{|\mathsf{TMSI}|}} + \mathsf{Adv}^{\mathsf{K.Ind}}_{\Pi}(\mathcal{A}) + 3 \cdot \mathsf{n_C} \cdot (\mathsf{Adv}^{\mathsf{ae}}_{\mathsf{AE}}(\mathcal{A}) + \mathsf{Adv}^{\mathsf{mac}}_{G}(\mathcal{A})).$$

Game $\mathbb{G}_3$: We modify the game $\mathbb{G}_3$ to replace outputs to call to the function G by truly randoms, but consistent values, which are independent of the input, but the same input gives same output. We argue that the security loss is precisely the advantage of the adversary against the pseudorandomness of the function G, of the public encryption scheme PKE and of the authenticated encryption scheme AE. It holds that:

$$\left| \Pr[\mathcal{A}_{\mathbb{G}_2}\mathsf{wins}] - \Pr[\mathcal{A}_{\mathbb{G}_3}\mathsf{wins}] \right| \leq \mathsf{n_C} \cdot (\mathsf{Adv}^{\mathsf{prf}}_{G}(\mathcal{A}) + \mathsf{Adv}^{\mathsf{ind-cca2}}_{\mathsf{PKE}}(\mathcal{A}) + \mathsf{Adv}^{\mathsf{ae}}_{\mathsf{AE}}(\mathcal{A})).$$

Winning the game $\mathbb{G}_3$: At this point, the adversary plays a game which consider to recover the bit b with a fully-randomized protocol. In particular, the transcript no longer depends on the clients long-term keys, hence two clients are indistinguishable. Thus, the adversary has only one option: guess the bit b without any specific information. So we obtain the following probability of winning the game $\mathbb{G}_3$: $\Pr[\mathcal{A}_{\mathbb{G}_3}\mathsf{wins}] = \frac{1}{2}$.
textbfSecurity Statement This yields the following result:

$$\mathsf{Adv}^{\mathsf{ww\text{-}unlink}}_{\Pi}(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{K.Ind}}_{\Pi}(\mathcal{A}_5) + \frac{1 + (q_{\mathsf{exec}} \cdot q_{\mathsf{id}})^2}{2^{|\mathsf{TMSI}|}} + \frac{q^2_{\mathsf{exec}}}{2^{|\mathsf{R}|}} + \frac{(q_{\mathsf{exec}} \cdot q_{\mathsf{id}})^2}{2^{|\mathsf{R}_{id}|}} + \mathsf{n_C} \cdot (3 \cdot \mathsf{Adv}^{\mathsf{mac}}_{G}(\mathcal{A}_1) + \mathsf{Adv}^{\mathsf{prf}}_{G}(\mathcal{A}_2) + 4 \cdot \mathsf{Adv}^{\mathsf{ae}}_{\mathsf{AE}}(\mathcal{A}_3) + \mathsf{Adv}^{\mathsf{ind-cca2}}_{\mathsf{PKE}}(\mathcal{A}_4)).$$

□

Key-Indistinguishability of PrivAKA.

We formalize the key-indistinguishability property, denoted K.Ind, as the guarantee that the session keys of honest sessions are indistinguishable from random. We consider the session ID sid to be a tuple: UID, $\mathsf{ID_{S_i}}$, R, R_{id}, $\mathsf{idx_C}$ and the value $\mathsf{Sqn_C}$, that are agreed during the session. As stipulated we can prove the key indistinguishability of PrivAKA, assuming the indistuiguishability, unforgeability of the outputs, and pseudorandomness properties of some of the internal cryptographic functions. This property is defined as follows:

Theorem 20. [$\mathsf{K.Ind} - \mathbf{resistance}$.] *Let $G : \{0,1\}^{\kappa} \times \{0,1\}^{d} \times \{0,1\}^{t} \times \{0,1\}^{t} \rightarrow \{0,1\}^{n}$ be our unitary function generalizing the internal cryptographic functions $\mathcal{F}_i$ and Π our PrivAKA protocol described in Section 4.7.1. Consider a $(t, q_{\mathsf{exec}}, q_{\mathsf{id}}, q_{\mathsf{s}}, q_{\mathsf{Op}}, q_G, q_{\mathsf{AE}}, q_{\mathsf{PKE}})$-adversary $\mathcal{A}$ against the* K.Ind*-security of the protocol Π, running in time t and creating at most q_{exec} party instance, with at most q_{id} user identification per instance, corrupting at most q_{s} servers, making at most q_{Op} OpAccess queries per operator per corrupted server, and making at most q_G, q_{AE}, q_{PKE} queries to respectively the functions G,* AE *and* PKE*. Denote the advantage of this adversary as $\mathsf{Adv}^{\mathsf{K.Ind}}_{\Pi}(\mathcal{A})$. Then there exists a $(t' \approx O(t), q' = q_G + q_{\mathsf{exec}})$-MAC-adversary $\mathcal{A}_1$ against G, a $(t' = O(t), q' = q_G + 2 \cdot q_{\mathsf{exec}} + 5 \cdot q_{\mathsf{s}} \cdot q_{\mathsf{Op}})$-PRF-adversary $\mathcal{A}_2$ against G, a $(t' = O(t), q' = q_G + q_{\mathsf{exec}})$-IND-CPA-adversary $\mathcal{A}_3$ against G,a $(t' = O(t), q' = q_{\mathsf{exec}} + q_{\mathsf{AE}})$-AE-adversary $\mathcal{A}_4$ against* AE*, and a $(t' = O(t), q' = q_{\mathsf{exec}} \cdot q_{\mathsf{id}} + q_{\mathsf{PKE}})$-IND-CCA2-adversary $\mathcal{A}_5$ against* PKE *such that:*

$$\mathsf{Adv}^{\mathsf{K.Ind}}_{\Pi}(\mathcal{A}) \leq \mathsf{n_C} \cdot (\frac{(q_{\mathsf{exec}} \cdot q_{\mathsf{id}})^2}{2^{|\mathsf{R}_{id}|}} + \frac{(q_{\mathsf{exec}} + q_{\mathsf{s}} \cdot q_{\mathsf{Op}})^2}{2^{|\mathsf{R}|}} + \mathsf{Adv}^{\mathsf{mac}}_{G}(\mathcal{A}_1) + \mathsf{Adv}^{\mathsf{prf}}_{G}(\mathcal{A}_2) + \mathsf{Adv}^{\mathsf{ind}}_{G}(\mathcal{A}_3) + \mathsf{Adv}^{\mathsf{ae}}_{\mathsf{AE}}(\mathcal{A}_4) + \mathsf{Adv}^{\mathsf{prf}}_{\mathsf{PKE}}(\mathcal{A}_5)).$$

Proof. Our proof has the following hops.

Game $\mathbb{G}_0$: This game works as the $\mathsf{K.Ind}$-game stipulated in our security model [3.2], but including the new oracles. The goal of the adversary $\mathcal{A}_{\mathbb{G}_0}$ is to distinguish, for a fresh instance that ends in an accepting state, the real session keys from random ones.
Game $\mathbb{G}_1$: We modify $\mathbb{G}_0$ giving $\mathcal{A}$ access to the new query $\mathsf{Corrupt}(\mathsf{P}, \mathtt{type})$. We note that this new query allows the corruption of the key operator independently of the corruption of the subscriber keys. This new query behaves as follows:

$\mathsf{Corrupt}(\mathsf{P}, \mathtt{type})$: yields to the adversary the long-term keys of party $\mathsf{P} \neq \mathsf{S}$ (else, if the oracle takes as input $\mathsf{P} = \mathsf{S}$, then it outputs $\perp$). The output of the oracle depends on the value $\mathtt{type} \in \{\mathtt{sub}, \mathtt{op}, \mathsf{all}\}$. If $\mathtt{type} = \mathtt{sub}$, then the returned value is sk_{P}. If $\mathtt{type} = \mathtt{op}$, then the oracle returns $\mathsf{sk}_{\mathsf{Op}}$. Then, for $\mathtt{type} = \mathsf{all}$, we return the both values sk_{P}, $\mathsf{sk}_{\mathsf{Op}}$. If $\mathtt{type} \in \{\mathtt{sub}, \mathsf{all}\}$, then P (and all its instances, past, present, or future), are considered to be adversarially controlled.

We argue that given any adversary $\mathcal{A}$ playing the game $\mathbb{G}_0$ and winning with probability $p_{\mathcal{A}}$ also wins game $\mathbb{G}_1$ with at least the same probability (since in game $\mathbb{G}_1$, $\mathcal{A}$ has more information).

$$\Pr[\mathcal{A}_{\mathbb{G}_0}\mathsf{wins}] \quad \leq \quad \Pr[\mathcal{A}_{\mathbb{G}_1}\mathsf{wins}].$$

Game $\mathbb{G}_2$: We modify $\mathbb{G}_1$ to only allow interactions with a single client (any future UReg calls for a client, are answered with an error symbol $\perp$). The challenger generates only a single operator key, which is associated with the operator chosen for the registered client and chooses a bit $b \in \{0, 1\}$. We proceed as follows: for any adversary $\mathcal{A}_{\mathbb{G}_1}$ winning the game $\mathbb{G}_1$ with a non-negligible success probability $\epsilon_{\mathbb{G}_1}$, we propose to construct a generic adversary $\mathcal{A}_{\mathbb{G}_2}$ winning the game $\mathbb{G}_2$ (with black-box access to the adversary $\mathcal{A}_{\mathbb{G}_1}$) with a non-negligeable probability.

Adversary $\mathcal{A}_{\mathbb{G}_2}$ begins by choosing a single client C. For every user registration request that $\mathcal{A}_{\mathbb{G}_1}$ sends to its challenger, $\mathcal{A}_{\mathbb{G}_2}$ responds as follows: if the registered client is C, then it forwards the exact UReg query that $\mathcal{A}_{\mathbb{G}_1}$ makes to its own UReg oracle. Else, if $\mathcal{A}_{\mathbb{G}_1}$ registers any client $\mathsf{C}^* \neq \mathsf{C}$, $\mathcal{A}_{\mathbb{G}_2}$ simulates the registration, generating $\mathsf{sk}_{\mathsf{C}^*}$ and $\mathsf{Sqn}_{\mathsf{C}^*}$, returning the latter value. Adversary $\mathcal{A}_{\mathbb{G}_2}$ also generates $n_{\mathsf{Op}} - 1$ operator keys $\mathsf{rsk}_{\mathsf{Op}^*}$ (for all operator Op^* such that $\mathsf{Op}^* \neq \mathsf{Op}$), and associates them with the clients as follows: the target client C is associated with the same operator given as input by $\mathcal{A}_{\mathbb{G}_1}$ to the UReg query (thus with the operator key $\mathsf{sk}_{\mathsf{Op}}$ generated by the challenger of game $\mathbb{G}_2$). Let this target operator be denoted as Op. Adversary $\mathcal{A}_{\mathbb{G}_2}$ queries $\mathsf{Corrupt}(\mathsf{C}, \mathsf{op})$ and stores $\mathsf{sk}_{\mathsf{Op}}$.

We distinguish two types of clients: the *brothers* of the target client (i.e. the clients which have the same operator as the target client) and the others ones. For the latter clients C^*, which are registered by $\mathcal{A}_{\mathbb{G}_1}$ with an operator $\mathsf{Op}^* \neq \mathsf{Op}$, adversary $\mathcal{A}_{\mathbb{G}_2}$ associates Op^* with one of its generated keys $\mathsf{rsk}_{\mathsf{Op}^*}$. Recall that, since adversary $\mathcal{A}_{\mathbb{G}_1}$ plays the game in the presence of n_{Op} operators, there are $n_{\mathsf{Op}} - 1$ keys which will be used this way. We call all clients $\mathsf{C}^* \neq \mathsf{C}$ registered by $\mathcal{A}_{\mathbb{G}_0}$ with the target operator Op the *brothers* of the target client C. Adversary $\mathcal{A}_{\mathbb{G}_2}$ associates each brother of C with the corrupted key $\mathsf{sk}_{\mathsf{Op}}$ it learns from its challenger.

In the rest of the simulation, whenever $\mathcal{A}_{\mathbb{G}_1}$ makes a query to an instance of some party C^*, not a brother of C, the adversary $\mathcal{A}_{\mathbb{G}_2}$ simulates the response using the values $\mathsf{sk}_{\mathsf{C}^*}$, $\mathsf{rsk}_{\mathsf{Op}^*}$, and the current value of Sqn. For the brothers of C, the simulation is done with $\mathsf{sk}_{\mathsf{C}^*}$, $\mathsf{sk}_{\mathsf{Op}}$, and the current Sqn. For the target client C, any queries are forwarded by $\mathcal{A}_{\mathbb{G}_2}$ to its challenger.

Any corruption or reveal queries are dealt with in a similar way. Note that $\mathcal{A}_{\mathbb{G}_2}$ cannot query $\mathsf{Corrupt}$ to its adversary (this is a condition of freshness). The simulation is thus perfect up to the Test query.

In the Test query, $\mathcal{A}_{\mathbb{G}_1}$ chooses a $\mathsf{fresh\ session}$ and sends it to $\mathcal{A}_{\mathbb{G}_2}$ (acting as a challenger). Note that $\mathcal{A}_{\mathbb{G}_2}$ will be able to test whether this instance is fresh, as freshness is defined in terms of

$\mathcal{A}_{\mathbb{G}_1}$'s queries. If $\mathcal{A}_{\mathbb{G}_1}$ queries Test with a client other than the target client C, then $\mathcal{A}_{\mathbb{G}_2}$ aborts the simulation, tests a random, fresh instance of the client C (creating one if necessary), and guesses the bit d, winning with probability $\frac{1}{2}$. Else, if $\mathcal{A}_{\mathbb{G}_1}$ queried a fresh instance of C, $\mathcal{A}_{\mathbb{G}_2}$ forwards this choice to its challenger and receives the challenger's input. The adversary $\mathcal{A}_{\mathbb{G}_2}$ forwards the input of the challenger to $\mathcal{A}_{\mathbb{G}_1}$ and then receives $\mathcal{A}$'s output d, which will be $\mathcal{A}_{\mathbb{G}_2}$'s own response to its own challenger.

Denote by E_1 the event that adversary tests C in game $\mathbb{G}_1$, while $\bar{\mathsf{E}_1}$ denotes the event that $\mathcal{A}_{\mathbb{G}_1}$ chooses to test $\mathsf{C}^* \neq \mathsf{C}$. It holds that:

$$\begin{aligned}
\Pr[\mathcal{A}_{\mathbb{G}_2}\mathsf{wins}] &= \Pr[\mathcal{A}_{\mathbb{G}_2}\mathsf{wins} \mid \mathsf{E}_1] \cdot \Pr[\mathsf{E}_1] + \Pr[\mathcal{A}_{\mathbb{G}_2}\mathsf{wins} \mid \bar{\mathsf{E}_1}] \cdot \Pr[\bar{\mathsf{E}_1}] \\
&\geq \frac{1}{\mathsf{n_C}} \Pr[\mathcal{A}_{\mathbb{G}_1}\mathsf{wins}] + \frac{1}{2} \cdot \big(1 - \frac{1}{\mathsf{n_C}}\big) \\
&\geq \frac{1}{\mathsf{n_C}} \Pr[\mathcal{A}_{\mathbb{G}_0}\mathsf{wins}] + \frac{1}{2} \cdot \big(1 - \frac{1}{\mathsf{n_C}}\big).
\end{aligned}$$

Note that adversary $\mathcal{A}_{\mathbb{G}_2}$ makes one extra query with respect to $\mathcal{A}_{\mathbb{G}_1}$, since we need to learn the key of the target operator.

Game $\mathbb{G}_3$: We modify $\mathbb{G}_2$ to ensure that the random values sampled by honest server instances are always unique. This gives us a security loss (related to the respective collisions between the R and R_{id} values in two different instances) of

$$\big|\Pr[\mathcal{A}_{\mathbb{G}_2}\mathsf{wins}] - \Pr[\mathcal{A}_{\mathbb{G}_3}\mathsf{wins}]\big| \leq \frac{(q_{\mathsf{exec}} \cdot q_{\mathsf{id}})^2}{2^{|\mathsf{R}_{id}|}} + \frac{(q_{\mathsf{exec}} + q_{\mathsf{s}} \cdot q_{\mathsf{Op}})^2}{2^{|\mathsf{R}|}}.$$

Game $\mathbb{G}_4$: This game behaves as game $\mathbb{G}_3$ with the restriction to only interact with only one server (any future UReg calls for a server would be answered with an error symbol $\perp$). Thus $\mathcal{A}$ loses the ability to obtain authentication vectors from corrupted servers. Such authentication vectors can either give information about the used sequence number and the long term keys, or forge a fresh challenge replaying some parts of these vectors. We recall that the challenge is split into four parts: a random value, a masked version of the current sequence number (a one-time-pad based on an anonymity key generated by the function G), a MAC computed with the function G, and an authenticated and encrypted version of a tuple of messages consisting of the next temporary identifier and the current index. Moreover, we note that all calls to the function G take in input a specific value of the related server, denoted $\mathsf{ID_{S_i}}$. The corrupted servers permit to obtain vectors based on the current sequence number, but different random values and different server identifier. So the related security loss is given by the collision on two outputs of the same function G with two different inputs (namely two inputs differ at least in the value of the network identifier) and by the indistinguishability of the function G. We recall that the Test Phase of the game can be only focus on a network which is or was never corrupted. This give us a security loss of:

$$\big|\Pr[\mathcal{A}_{\mathbb{G}_4}\mathsf{wins}] - \Pr[\mathcal{A}_{\mathbb{G}_3}\mathsf{wins}]\big| \leq \mathsf{Adv}^{\mathsf{ind}}_{G}(\mathcal{A}) + \mathsf{Adv}^{\mathsf{mac}}_{G}(\mathcal{A}).$$

Game $\mathbb{G}_5$: We modify $\mathbb{G}_4$ to replace outputs of the internal cryptographic functions by truly random, but consistent values (they are independent of the input, but the same input gives the same output). We argue that the security loss is precisely the advantage of the adversary $\mathcal{A}$ against the pseudorandomness of function G, and the security respectively of PKE and of AE. Note that the total number of queries to the functions are at most $2 \cdot q_G$ and one q_{AE} and one q_{PKE} per honest instance (thus totalling at most $\mathsf{q}_G + (2 \cdot q_{\mathsf{sess}})$ queries to the function G, q_{sess} queries to the function PKE, and q_{sess} queries to the function AE).

$$\big|\Pr[\mathcal{A}_{\mathbb{G}_4}\mathsf{wins}] - \Pr[\mathcal{A}_{\mathbb{G}_5}\mathsf{wins}]\big| \leq \mathsf{Adv}^{\mathsf{ind-cca2}}_{\mathsf{PKE}}(\mathcal{A}) + \mathsf{Adv}^{\mathsf{prf}}_{G}(\mathcal{A}) + \mathsf{Adv}^{\mathsf{ae}}_{\mathsf{AE}}(\mathcal{A}).$$

Winning $\mathbb{G}_5$: At this point, the adversary plays a game in the presence of a single client C. The goal of this adversary is to distinguish a random session key from a fresh session key. However, in game $\mathbb{G}_5$, queries to G return truly random, consistent values. In this case, the adversary can do no better than guessing. Thus, we have: $\Pr[\mathcal{A}_{\mathbb{G}_5}\mathsf{wins}] = \frac{1}{2}$.
Security statement: This yields the following result:

$$\mathsf{Adv}^{\mathsf{K.Ind}}_{\Pi}(\mathcal{A}_{\mathbb{G}_0}) \leq \mathsf{n_C} \cdot \Big(\frac{(q_{\mathsf{exec}} \cdot q_{\mathsf{id}})^2}{2^{|\mathsf{R}_{id}|}} + \frac{(q_{\mathsf{exec}} + q_{\mathsf{s}} \cdot q_{\mathsf{Op}})^2}{2^{|\mathsf{R}|}} + \mathsf{Adv}^{\mathsf{mac}}_{G}(\mathcal{A}_1) + \mathsf{Adv}^{\mathsf{prf}}_{G}(\mathcal{A}_2) + \mathsf{Adv}^{\mathsf{ind}}_{G}(\mathcal{A}_3) + \mathsf{Adv}^{\mathsf{ae}}_{\mathsf{AE}}(\mathcal{A}_4) + \mathsf{Adv}^{\mathsf{prf}}_{\mathsf{PKE}}(\mathcal{A}_5)\Big).$$

This concludes the proof. □

Impersonation of PrivAKA.

We present first the client-impersonation resistance proof, then the equivalent statement for server impersonations.

Theorem 21. $[\mathsf{C.Imp} - \mathbf{resistance}.]$ *Let* $G : \{0,1\}^{\kappa} \times \{0,1\}^{d} \times \{0,1\}^{t} \times \{0,1\}^{t} \rightarrow \{0,1\}^{n}$ *our unitary function generalizing the internal cryptographic functions* $\mathcal{F}_i$ *and* Π *our PrivAKA protocol described in Section 4.7.1. Consider a* $(t, q_{\mathsf{exec}}, q_{\mathsf{id}}, q_{\mathsf{s}}, q_{\mathsf{Op}}, q_G, q_{\mathsf{AE}}, q_{\mathsf{PKE}})$*-adversary* $\mathcal{A}$ *against the* $\mathsf{C.Imp}$*-security of the protocol* Π*, running in time* t *and creating at most* q_{exec} *party instance, with at most* q_{id} *user identification per instance, corrupting at most* q_{s} *servers, making at most* q_{Op} *OpAccess queries per operator per corrupted server, and making at most* q_G*,* q_{AE}*,* q_{PKE} *queries to respectively the functions* G*,* AE *and* PKE*. Denote the advantage of this adversary as* $\mathsf{Adv}^{\mathsf{C.Imp}}_{\Pi}(\mathcal{A})$*, Then, there exist a* $(t' \approx O(t), q' = q_G + 2 \cdot q_{\mathsf{exec}} + 5 \cdot q_{\mathsf{s}} \cdot q_{\mathsf{Op}})$*-PRF-adversary* $\mathcal{A}_1$ *against* G*, a* $(t' = O(t), q' = q_{\mathsf{exec}} + q_{\mathsf{AE}})$*-AE-adversary* $\mathcal{A}_2$ *against* AE*, and a* $(t' = O(t), q' = q_{\mathsf{exec}} \cdot q_{\mathsf{id}} + q_{\mathsf{PKE}})$*-IND-CCA2-adversary* $\mathcal{A}_3$ *against* PKE *such that:*

$$\mathsf{Adv}^{\mathsf{C.Imp}}_{\Pi}(\mathcal{A}) \leq \mathsf{n_C} \cdot \Big(\mathsf{Adv}^{\mathsf{prf}}_{G}(\mathcal{A}_1) + \mathsf{Adv}^{\mathsf{ae}}_{\mathsf{AE}}(\mathcal{A}_2) + \mathsf{Adv}^{\mathsf{ind-cca2}}_{\mathsf{PKE}}(\mathcal{A}_3) + \frac{(q_{\mathsf{exec}} \cdot q_{\mathsf{id}})^2}{2^{|\mathsf{R}_{id}|}} + \frac{(q_{\mathsf{exec}})^2}{2^{|\mathsf{R}|}} + \frac{q_{\mathsf{exec}}}{2^{|\mathsf{Res}|}} + \frac{1}{2^{\kappa}} + \frac{q_{\mathsf{exec}} \cdot q_{\mathsf{id}}}{2^{|\mathsf{ID}|}}\Big).$$

Proof. Our proof has the following hops:
Game $\mathbb{G}_0$: This game works as the $\mathsf{C.Imp}$-game: When $\mathcal{A}$ stops, he wins if there exists an instance S_i that ends in an accepting state with session ID sid and partner ID pid such that: (a) pid is not adversarially controlled ($\mathsf{sk}_{\mathsf{pid}}$ has not been corrupted), (b) no other instance C_i exists for $\mathsf{pid} = \mathsf{S}_i$ that ends in an accepting state, with session ID sid.
Game $\mathbb{G}_1$: We modify the game to allow the new $\mathsf{Corrupt}(\mathsf{P}, \mathtt{type})$ query from the previous proof. It holds that:

$$\Pr[\mathcal{A}_{\mathbb{G}_0}\mathsf{wins}] \quad \leq \quad \Pr[\mathcal{A}_{\mathbb{G}_1}\mathsf{wins}].$$

Game $\mathbb{G}_2$: We modify $\mathbb{G}_1$ to only interact with a single client, as in the previous proof, giving a security loss of:

$$\Pr[\mathcal{A}_{\mathbb{G}_2}\mathsf{wins}] \quad \geq \quad \frac{1}{\mathsf{n_C}} \Pr[\mathcal{A}_{\mathbb{G}_1}\mathsf{wins}] + \frac{1}{2} \cdot \big(1 - \frac{1}{\mathsf{n_C}}\big).$$

Game $\mathbb{G}_3$: We now restrict the adversary to using a single server As detailed in the key-indistinguishability proof, the related security loss is given by:

$$\left|\Pr[\mathcal{A}_{\mathbb{G}_3}\mathsf{wins}] - \Pr[\mathcal{A}_{\mathbb{G}_2}\mathsf{wins}]\right| \leq \mathsf{Adv}^{\mathsf{prf}}_{G}(\mathcal{A}).$$

Game $\mathbb{G}_4$: We modify $\mathbb{G}_4$ to replace outputs to calls to all the internal cryptographic functions by truly random, but consistent values, and as before, we lose a term:

$$\left|\Pr[\mathcal{A}_{\mathbb{G}_3}\mathsf{wins}] - \Pr[\mathcal{A}_{\mathbb{G}_4}\mathsf{wins}]\right| \leq \mathsf{Adv}^{\mathsf{prf}}_{G}(\mathcal{A}) + \mathsf{Adv}^{\mathsf{ae}}_{\mathsf{AE}}(\mathcal{A}) + \mathsf{Adv}^{\mathsf{ind-cca2}}_{\mathsf{PKE}}(\mathcal{A}).$$

Game $\mathbb{G}_5$: We modify $\mathbb{G}_4$ to ensure that the random values sampled by honest server instances are always unique. As in the unlinkability proof, this yields a loss of:

$$\left|\Pr[\mathcal{A}_{\mathbb{G}_4}\mathsf{wins}] - \Pr[\mathcal{A}_{\mathbb{G}_5}\mathsf{wins}]\right| \quad \leq \frac{(q_{\mathsf{exec}}\cdot q_{\mathsf{id}})^2}{2^{|\mathsf{R}_{id}|}} + \frac{q^2_{\mathsf{exec}}}{2^{|\mathsf{R}|}}.$$

Winning $\mathbb{G}_5$: At this point, the adversary plays a game with a single client and server. A server instance S_i only accepts $\mathcal{A}_{\mathbb{G}_5}$, if the latter can generate a correct identification response ID and a consistent authentication response Res for a fresh session sid. Assume that this happens against accepting instance S_i of the server, for some target session sid. Note that the values Res and ID computed by C_i are purely random, but consistent. Thus, the adversary has three options for each of these values: (a) forwarding a value already received from the honest client for the same input values, of which sk_C is unknown; (b) guessing the key sk_C; or (c) guessing the value. The first option yields no result, since it implies there exists a previous client instance with the same session id sid as the client. The second option happens with a probability of $2^{-|\mathsf{sk}_\mathsf{C}|}$. The third option occurs with a probability of $2^{-|\mathsf{Res}|} + 2^{-|\mathsf{ID}|}$ per session, thus a total of $q_{\mathsf{exec}} \cdot (2^{-|\mathsf{Res}|} + q_{\mathsf{id}} \cdot 2^{-|\mathsf{ID}|})$. Thus,

$$\Pr[\mathcal{A}_{\mathbb{G}_5}\mathsf{wins}] = 2^{-|\mathsf{sk}_\mathsf{C}|} + q_{\mathsf{exec}} \cdot (2^{-|\mathsf{Res}|} + q_{\mathsf{id}} \cdot 2^{-|\mathsf{ID}|}).$$

Security statement: This yields the following result:

$$\mathsf{Adv}^{\mathsf{C.Imp}}_{\Pi}(\mathcal{A}_{\mathbb{G}_0}) \leq \mathsf{n}_\mathsf{C} \cdot (\mathsf{Adv}^{\mathsf{prf}}_{G}(\mathcal{A}_1) + \mathsf{Adv}^{\mathsf{ae}}_{\mathsf{AE}}(\mathcal{A}_2) + \mathsf{Adv}^{\mathsf{ind-cca2}}_{\mathsf{PKE}}(\mathcal{A}_3) + \frac{(q_{\mathsf{exec}} \cdot q_{\mathsf{id}})^2}{2^{|\mathsf{R}_{id}|}} + \frac{(q_{\mathsf{exec}})^2}{2^{|\mathsf{R}|}} + \frac{q_{\mathsf{exec}}}{2^{|\mathsf{Res}|}} + \frac{1}{2^\kappa} + \frac{q_{\mathsf{exec}} \cdot q_{\mathsf{id}}}{2^{|\mathsf{ID}|}}).$$

□

Theorem 22. [$\mathsf{S.Imp} - \mathbf{resistance}$.] *Let* $G : \{0,1\}^\kappa \times \{0,1\}^d \times \{0,1\}^t \times \{0,1\}^t \to \{0,1\}^n$ *our unitary function generalizing the internal cryptographic functions* $\mathcal{F}_i$ *and* Π *our PrivAKA protocol described in Section 4.7.1. Consider a* $(t, q_{\mathsf{exec}}, q_{\mathsf{id}}, q_\mathsf{s}, q_{\mathsf{Op}}, q_G, q_{\mathsf{AE}}, q_{\mathsf{PKE}})$*-adversary* $\mathcal{A}$ *against the* S.Imp*-security of the protocol* Π*, running in time* t *and creating at most* q_{exec} *party instance, with at most* q_{id} *user identification per instance, corrupting at most* q_s *servers, making at most* q_{Op} *OpAccess queries per operator per corrupted server, and making at most* q_G, q_{AE}, q_{PKE} *queries to respectively the functions* G, AE *and* PKE. *Denote the advantage of this adversary as* $\mathsf{Adv}^{\mathsf{S.Imp}}_{\Pi}(\mathcal{A})$*, Then, there exist a* $(t' \approx O(t), q' = q_G + 2 \cdot q_{\mathsf{exec}} + 5 \cdot q_\mathsf{s} \cdot q_{\mathsf{Op}})$*-PRF-adversary* $\mathcal{A}_1$ *against* G*, a* $(t' = O(t), q' = q_{\mathsf{exec}} + q_{\mathsf{AE}})$*-AE-adversary* $\mathcal{A}_2$ *against* AE *and a* $(t' = O(t), q' = q_{\mathsf{exec}} \cdot q_{\mathsf{id}} + q_{\mathsf{PKE}})$*-IND-CCA2-adversary* $\mathcal{A}_3$ *against* PKE *such that:*

$$\mathsf{Adv}^{\mathsf{S.Imp}}_{\Pi}(\mathcal{A}) \leq \mathsf{n}_\mathsf{C} \cdot \big(\frac{q_{\mathsf{sess}}}{2^{|\mathsf{Mac_s}|}} + \frac{1}{2^\kappa} + \mathsf{Adv}^{\mathsf{prf}}_{G}(\mathcal{A}_1) + \mathsf{Adv}^{\mathsf{ae}}_{\mathsf{AE}}(\mathcal{A}_2) + \mathsf{Adv}^{\mathsf{ind-cca2}}_{\mathsf{PKE}}(\mathcal{A}_3)\big).$$

Proof. Our proof has the following hops:
Game $\mathbb{G}_0$: This game works as the S.Imp-game. The adversary $\mathcal{A}$ wins if there exists an instance C_i that ends in an accepting state with session ID sid and partner ID and pid s.t.: (a) pid $=$ S, (b)

no instance S_j exists such as S_j and C_i has the same session ID sid, (c) C_i and these partners are not adversarially controlled.

Game $\mathbb{G}_1$: We add the new query $\mathsf{Corrupt}(\mathsf{P}, \mathtt{type})$ as in the key-indistinguishability proof, such that:

$$\Pr[\mathcal{A}_{\mathbb{G}_0}\mathsf{wins}] \quad \leq \quad \Pr[\mathcal{A}_{\mathbb{G}_1}\mathsf{wins}].$$

Game $\mathbb{G}_2$: We modify $\mathbb{G}_1$ to only interact with a single client, as in the previous proofs, and lose:

$$\Pr[\mathcal{A}_{\mathbb{G}_2}\mathsf{wins}] \quad \geq \quad \frac{1}{\mathsf{n_C}}\Pr[\mathcal{A}_{\mathbb{G}_1}\mathsf{wins}] + \frac{1}{2}\cdot\big(1-\frac{1}{\mathsf{n_C}}\big).$$

Game $\mathbb{G}_3$: Again, we restrict the adversary to only one server. This give us a security loss

$$\big|\Pr[\mathcal{A}_{\mathbb{G}_3}\mathsf{wins}] - \Pr[\mathcal{A}_{\mathbb{G}_2}\mathsf{wins}]\big| \leq \mathsf{Adv}^{\mathsf{prf}}_{G}(\mathcal{A}).$$

Game $\mathbb{G}_4$: We modify $\mathbb{G}_3$ to replace outputs to calls to G, AE, and PKE by truly random, but consistent values and as before, it holds that:

$$\big|\Pr[\mathcal{A}_{\mathbb{G}_3}\mathsf{wins}] - \Pr[\mathcal{A}_{\mathbb{G}_4}\mathsf{wins}]\big| \leq \mathsf{Adv}^{\mathsf{prf}}_{G}(\mathcal{A}) + \mathsf{Adv}^{\mathsf{ae}}_{\mathsf{AE}}(\mathcal{A}) + \mathsf{Adv}^{\mathsf{ind-cca2}}_{\mathsf{PKE}}(\mathcal{A}).$$

Winning $\mathbb{G}_4$: At this point, the adversary plays a game with a single client C_i, which only accepts $\mathcal{A}_{\mathbb{G}_4}$, if the authentication challenge is verified for some session sid. Assume that this happens against accepting instance C_i of the target client, for some target session sid. Note that the $\mathsf{Mac_S}$ value computed (for verification) by C_i is purely random, but consistent. Thus, the adversary has three options: (a) forwarding a value already received from the honest server for the same input values R; Sqn; $\mathsf{sk_{Op}}$; $\mathsf{sk_C}$, of which $\mathsf{sk_C}$ is unknown; (b) guessing the key $\mathsf{sk_C}$; or (c) guessing the response. The first option yields no result since there are no collision between the transcript of two different servers since all the servers have a different session ID. The second option happens with a probability of $2^{-|\mathsf{sk_C}|}$. The third option occurs with a probability of $2^{-|\mathsf{Mac_S}|}$ per session, thus a total of $q_{\mathsf{exec}}\cdot 2^{-|\mathsf{Mac_S}|}$. Thus,

$$\Pr[\mathcal{A}_{\mathbb{G}_4}\mathsf{wins}] = 2^{-|\mathsf{sk_C}|} + q_{\mathsf{sess}}\cdot 2^{-|\mathsf{Mac_S}|}.$$

Security statement: This yields the following result:

$$\mathsf{Adv}^{\mathsf{S.Imp}}_{\Pi}(\mathcal{A}_{\mathbb{G}_0}) \leq \mathsf{n_C}\cdot\big(\frac{q_{\mathsf{sess}}}{2^{|\mathsf{Mac_S}|}} + \frac{1}{2^{\kappa}} + \mathsf{Adv}^{\mathsf{prf}}_{G}(\mathcal{A}_1) + \mathsf{Adv}^{\mathsf{ae}}_{\mathsf{AE}}(\mathcal{A}_2) + \mathsf{Adv}^{\mathsf{ind-cca2}}_{\mathsf{PKE}}(\mathcal{A}_3)\big).$$

□

Soundness and state-confidentiality of PrivAKA.

Theorem 23. [$\mathsf{St.Conf}-$ **resistance**.] *Let G and G^* our unitary functions generalizing the internal cryptographic functions $\mathcal{F}_i$ respectively keyed with the subscriber key and the operator key, and Π, our PrivAKA protocol described in Section 4.7.1. Consider a $(t, q_{\mathsf{exec}}, q_{\mathsf{id}}, q_{\mathsf{Op}}, q_G, q_{G^*}, q_{\mathsf{AE}}, q_{\mathsf{PKE}})$-adversary $\mathcal{A}$ against the $\mathsf{St.Conf}$-security of the protocol Π, running in time t and executing q_{sess} sessions of the protocol Π, making at most q_{Op} queries to any operator, and making q_G, q_{G^*}, $q_{\mathsf{AE}}, q_{\mathsf{PKE}}$ queries to respectively the functions G, G^*, AE and PKEresp. q_{G^*}) queries to the function G (resp. G^*). Denote the advantage of this adversary as $\mathsf{Adv}^{\mathsf{St.Conf}}_{\Pi}(\mathcal{A})$. Then there exist a $(t' \approx O(t), q' = q_G + 5\cdot q_{\mathsf{Op}} + 2\cdot q_{\mathsf{exec}})$-PRF-adversary $\mathcal{A}_1$ against G, a $(t' = O(t), q' =$*

$q_{\mathsf{exec}} + q_{\mathsf{AE}}$)*-AE-adversary* $\mathcal{A}_2$ *against* AE*, a* $(t' = O(t), q' = q_{\mathsf{exec}} \cdot q_{\mathsf{id}} + q_{\mathsf{PKE}})$*-IND-CCA2-adversary* $\mathcal{A}_3$ *against* PKE*, and* $(t' \approx O(t), q' = q_{G^*})$*-PRF-adversary* $\mathcal{A}_4$ *against* G^* *a such that:*

$$\mathsf{Adv}^{\mathsf{St.Conf}}_{\Pi}(\mathcal{A}) \leq \mathsf{n_C} \cdot (\frac{1}{2^{|\mathsf{sk_C}|}} + \frac{1}{2^{|\mathsf{sk_{Op}}|}} + \frac{1}{2^{|\mathsf{Sqn}|}} + \mathsf{Adv}^{\mathsf{prf}}_{G}(\mathcal{A}_1) + \mathsf{Adv}^{\mathsf{ae}}_{\mathsf{AE}}(\mathcal{A}_2) + \mathsf{Adv}^{\mathsf{ind-cca2}}_{\mathsf{PKE}}(\mathcal{A}_3)) + \mathsf{Adv}^{\mathsf{prf}}_{G^*}(\mathcal{A}_4).$$

Proof. Our proof has the following hops.

Game $\mathbb{G}_0$: This game works as the St.Conf-game stipulated in our security model. The goal of the adversary $\mathcal{A}_{\mathbb{G}_0}$ is to recover at least one secret value, i.e the subscriber key $\mathsf{sk_C}$, my operator key $\mathsf{sk_{Op}}$, or the subscriber sequence number $\mathsf{Sqn_C}$ for a fresh instance.

Game $\mathbb{G}_1$: We modify $\mathbb{G}_0$ to only allow interactions with one operator. The challenger related to the game $\mathbb{G}_1$ only generates a single operator key, which is associated with the operator chosen for the registered client. We proceed as follows: for any adversary $\mathcal{A}_{\mathbb{G}_0}$ winning the game $\mathbb{G}_0$ with a non-negligible success probability $\epsilon_{\mathbb{G}_0}$, we propose to construct a generic adversary $\mathcal{A}_{\mathbb{G}_1}$ winning the game $\mathbb{G}_1$ with a black-box access to the adversary $\mathcal{A}_{\mathbb{G}_0}$, also with non-negligible probability.

Adversary $\mathcal{A}_{\mathbb{G}_1}$ begins by choosing a single operator Op. It generates $n_{\mathsf{Op}} - 1$ operator keys, denoted $\mathsf{rsk_{Op^*}}$. Then, for every user registration request that $\mathcal{A}_{\mathbb{G}_0}$ sends to its challenger, $\mathcal{A}_{\mathbb{G}_1}$ responds as follows: if the request CreateCl(.) takes in input the operator Op, then it forwards the same query to its own oracle. Else, if $\mathcal{A}_{\mathbb{G}_0}$ sends a registration request based on any operator $\mathsf{Op}^* \neq \mathsf{Op}$, $\mathcal{A}_{\mathbb{G}_1}$ simulates the registration, generating a subscriber key $\mathsf{sk_{C^*}}$ and a sequence number $\mathsf{Sqn_{C^*}}$, returning the latter value. Moreover, each new client registered with the operator Op (respectively any Op^*) is associated with the related operator key $\mathsf{sk_{Op}}$(respectively $\mathsf{rsk_{Op^*}}$).

We distinguish two types of clients: the *brothers* of the target client (i.e. the clients which have the same operator as the target client) and the others ones. For the latter C^* registered by $\mathcal{A}_{\mathbb{G}_1}$ with an operator $\mathsf{Op}^* \neq \mathsf{Op}$, adversary $\mathcal{A}_{\mathbb{G}_2}$ associates Op^* with one of its generated keys $\mathsf{rsk_{Op^*}}$.

In the rest of the simulation, whenever $\mathcal{A}_{\mathbb{G}_0}$ makes a query to an instance of some party C^* (from any operator except Op), the adversary $\mathcal{A}_{\mathbb{G}_1}$ simulates the response using the values $\mathsf{sk_{C^*}}$, $\mathsf{rsk_{Op^*}}$, and the current value of $\mathsf{Sqn_{C^*}}$. For the other clients, the query is forwarded by $\mathcal{A}_{\mathbb{G}_1}$ to its own challenger.

Any corruption or reveal queries are dealt with in a similar way. Note that $\mathcal{A}_{\mathbb{G}_1}$ cannot query Corrupt to its adversary (this is a condition of freshness). The simulation is thus perfect up to the Test query.

In the Test query, $\mathcal{A}_{\mathbb{G}_0}$ chooses a fresh instance and sends it to $\mathcal{A}_{\mathbb{G}_1}$ (acting as a challenger). Note that $\mathcal{A}_{\mathbb{G}_1}$ will be able to test whether this instance is fresh, as freshness is defined in terms of $\mathcal{A}_{\mathbb{G}_0}$'s queries. If $\mathcal{A}_{\mathbb{G}_0}$ queries an instance C^*_i for the Test query ,then $\mathcal{A}_{\mathbb{G}_1}$ aborts the simulation, tests a random tuple about any fresh instance of the client C (creating one if necessary), winning with probability $\frac{1}{2^{|\mathsf{sk_C}|}} + \frac{1}{2^{|\mathsf{sk_{Op}}|}} + \frac{1}{2^{|\mathsf{Sqn_C}|}}$. Else, if $\mathcal{A}_{\mathbb{G}_0}$ sends a tuple of a fresh instance of C_i, $\mathcal{A}_{\mathbb{G}_1}$ forwards this choice to its challenger and receives the challenger's output which contains the result of this game.

Denote by E_1 the event that adversary $\mathcal{A}_{\mathbb{G}_0}$ tests an instance C_i (from the chosen operator Op), while $\bar{\mathsf{E}}_1$ denotes the event that $\mathcal{A}_{\mathbb{G}_0}$ chooses to test C^*_i.

It holds that:

$$\begin{aligned} \Pr[\mathcal{A}_{\mathbb{G}_1}\mathsf{wins}] &= \Pr[\mathcal{A}_{\mathbb{G}_1}\mathsf{wins} \mid \mathsf{E}_1] \cdot \Pr[\mathsf{E}_1] + \Pr[\mathcal{A}_{\mathbb{G}_1}\mathsf{wins} \mid \bar{\mathsf{E}}_1] \cdot \Pr[\bar{\mathsf{E}}_1] \\ &\geq \frac{1}{n_{\mathsf{Op}}} \Pr[\mathcal{A}_{\mathbb{G}_0}\mathsf{wins}] + \big(1 - \frac{1}{n_{\mathsf{Op}}}\big) \cdot (\frac{1}{2^{|\mathsf{sk_C}|}} + \frac{1}{2^{|\mathsf{sk_{Op}}|}} + \frac{2}{2^{|\mathsf{Sqn_C}|}}). \end{aligned}$$

Note that adversary $\mathcal{A}_{\mathbb{G}_1}$ makes no query with respect to $\mathcal{A}_{\mathbb{G}_0}$.

Game $\mathbb{G}_2$: We modify $\mathbb{G}_1$ to only allow interactions with a single client (any future CreateCl (Op) calls for a client would be answered with an error symbol $\perp$). We recall that the two adversaries $\mathcal{A}_{\mathbb{G}_1}$ and $\mathcal{A}_{\mathbb{G}_2}$ interact with clients from a single operator key, denoted Op, which is associated with the operator key $\mathsf{sk}_{\mathsf{Op}}$. We proceed as follows: for any adversary $\mathcal{A}_{\mathbb{G}_1}$ winning the game $\mathbb{G}_1$ with a non-negligible success probability $\epsilon_{\mathbb{G}_1}$, we propose to construct a generic adversary $\mathcal{A}_{\mathbb{G}_2}$ winning the game $\mathbb{G}_2$ with a black-box access to the adversary $\mathcal{A}_{\mathbb{G}_1}$ also with non-negligible probability. Adversary $\mathcal{A}_{\mathbb{G}_2}$ begins by choosing a single client C. For every user registration request that $\mathcal{A}_{\mathbb{G}_1}$ sends to its challenger, $\mathcal{A}_{\mathbb{G}_2}$ responds as follows: for a new client $\mathsf{C}^* \neq \mathsf{C}$, it generates $\mathsf{sk}_{\mathsf{C}^*}$ and $\mathsf{Sqn}_{\mathsf{C}^*}$, returning the latter value.

In the rest of the simulation, whenever $\mathcal{A}_{\mathbb{G}_1}$ makes a query to an instance of some party C^*, the adversary $\mathcal{A}_{\mathbb{G}_2}$ simulates the response using the oracle of the function G^* and the values $\mathsf{sk}_{\mathsf{C}^*}$ and the current value of $\mathsf{Sqn}_{\mathsf{C}^*}$. Indeed, since the adversary can corrupt any operator key, he requires the oracle of G^* permitting to simulate all the queries of the brothers of the target client.

For the target client C, all queries are forwarded by $\mathcal{A}_{\mathbb{G}_2}$ to its challenger. Any corruption or reveal queries are dealt with in a similar way. Note that $\mathcal{A}_{\mathbb{G}_2}$ cannot query Corrupt to its adversary (this is a condition of freshness). The simulation is thus perfect up to the Test query.

In the Test query, $\mathcal{A}_{\mathbb{G}_1}$ chooses a fresh instance and sends it to $\mathcal{A}_{\mathbb{G}_2}$ (acting as a challenger). Note that $\mathcal{A}_{\mathbb{G}_2}$ will be able to test whether this instance is fresh, as freshness is defined in terms of $\mathcal{A}_{\mathbb{G}_1}$'s queries. If $\mathcal{A}_{\mathbb{G}_1}$ queries Test with a client other than the target client C, then $\mathcal{A}_{\mathbb{G}_2}$ aborts the simulation, tests a random tuple as the previous reduction. Else, if $\mathcal{A}_{\mathbb{G}_1}$ queried a fresh instance of C, $\mathcal{A}_{\mathbb{G}_2}$ forwards this choice to its challenger and receives the challenger's which contains the result of this game. It holds that:

$$\Pr[\mathcal{A}_{\mathbb{G}_2}\mathsf{wins}] \;\geq\; \frac{1}{\mathsf{n}_{\mathsf{C},\mathsf{Op}}} \Pr[\mathcal{A}_{\mathbb{G}_1}\mathsf{wins}] + \frac{1}{2} \cdot \big(1 - \frac{1}{\mathsf{n}_{\mathsf{C},\mathsf{Op}}}\big)$$

, with at most $\mathsf{n}_{\mathsf{C},\mathsf{Op}}$ clients per operator.

Note that adversary $\mathcal{A}_{\mathbb{G}_2}$ makes no extra query with respect to $\mathcal{A}_{\mathbb{G}_1}$.

Game $\mathbb{G}_3$: We modify $\mathbb{G}_2$ to replace outputs of the internal cryptographic functions (G, PKE, and AE) by truly random, but consistent values (they are independent of the input, but the same input gives the same output). We argue that the security loss is precisely the advantage of the adversary $\mathcal{A}$ against the pseudorandomness of functions G and G^*, and related security of the functions PKE and AE.

$$\big|\Pr[\mathcal{A}_{\mathbb{G}_3}\mathsf{wins}] - \Pr[\mathcal{A}_{\mathbb{G}_2}\mathsf{wins}]\big| \leq \mathsf{Adv}^{\mathsf{prf}}_{G}(\mathcal{A}) + \mathsf{Adv}^{\mathsf{prf}}_{G^*}(\mathcal{A}) + \mathsf{Adv}^{\mathsf{ae}}_{\mathsf{AE}}(\mathcal{A}) + \mathsf{Adv}^{\mathsf{ind-cca2}}_{\mathsf{PKE}}(\mathcal{A}).$$

Winning Game $\mathbb{G}_3$: At this point, the adversary plays a game with an uncorruptible single client C_i in a protocol including truly but consistent values. He wins if he can output a tuple $(\mathsf{C}_i, \mathsf{sk}^*_{\mathsf{C}}, \mathsf{sk}^*_{\mathsf{Op}}, \mathsf{Sqn}_{\mathsf{C}}{}^*, \mathsf{Sqn}_{\mathsf{Op},\mathsf{C}}{}^*)$ such as at least one of these values corresponds to the real related secret value of the instance C_i. Thus, the adversary has only one choice to win this game: guessing each value. So the probability that the adversary $\mathcal{A}_{\mathbb{G}_3}$ wins is as follows:

$$\Pr[\mathcal{A}_{\mathbb{G}_3}\mathsf{wins}] = \frac{1}{2^{|\mathsf{sk}_{\mathsf{C}}|}} + \frac{1}{2^{|\mathsf{sk}_{\mathsf{Op}}|}} + \frac{2}{2^{|\mathsf{Sqn}|}}.$$

Security statement: This yields the following result:

$$\mathsf{Adv}^{\mathsf{St.Conf}}_{\Pi}(\mathcal{A}_{\mathbb{G}_0}) \leq \mathsf{n}_{\mathsf{C}} \cdot \big(\frac{1}{2^{|\mathsf{sk}_{\mathsf{C}}|}} + \frac{1}{2^{|\mathsf{sk}_{\mathsf{Op}}|}} + \frac{2}{2^{|\mathsf{Sqn}|}} + \mathsf{Adv}^{\mathsf{prf}}_{G}(\mathcal{A}_1) + \mathsf{Adv}^{\mathsf{ae}}_{\mathsf{AE}}(\mathcal{A}_2) + \mathsf{Adv}^{\mathsf{ind-cca2}}_{\mathsf{PKE}}(\mathcal{A}_3)\big) + \mathsf{Adv}^{\mathsf{prf}}_{G^*}(\mathcal{A}_4).$$

□

We provide the following theorem about the client-impersonation, denoted Sound-security, of the fixed variant of the AKA protocol.

Theorem 24. [$\mathsf{Sound}-\mathbf{resistance}$.] *Let* $G : \{0,1\}^{\kappa} \times \{0,1\}^{d} \times \{0,1\}^{t} \times \{0,1\}^{t} \rightarrow \{0,1\}^{n}$ *be our unitary function generalizing the internal cryptographic functions* $\mathcal{F}_i$ *and* Π *our PrivAKA protocol described in Section 4.7.1. Consider a* $(t, q_{\mathsf{exec}}, q_{\mathsf{Op}}, q_G, q_{\mathsf{AE}}, q_{\mathsf{PKE}})$*-adversary* $\mathcal{A}$ *against the* Sound*-security of the protocol* Π*, running in time* t *and executing* q_{exec} *sessions of the protocol, making at most* q_{Op} *queries to any operator, and making* q_G, $q_{\mathsf{AE}}, q_{\mathsf{PKE}}$ *queries to respectively the functions* G, AE *and* PKE. *Denote the advantage of this adversary as* $\mathsf{Adv}^{\mathsf{Sound}}_{\Pi}(\mathcal{A})$. *Then there exist a* $(t' \approx O(t), q' = q_G + q_{\mathsf{sess}})$*-MAC-adversary* $\mathcal{A}_1$ *against* G, $(t' \approx O(t), q' = q_G + 2 \cdot q_{\mathsf{exec}} + 5 \cdot q_{\mathsf{Op}})$*-PRF-adversary* $\mathcal{A}_2$ *against* G, *a* $(t' = O(t), q' = q_G + q_{\mathsf{exec}})$*-IND-CPA-adversary* adv_3 *against* G, *a* $(t' = O(t), q' = q_{\mathsf{exec}} + q_{\mathsf{AE}})$*-AE-adversary* adv_4 *against* AE *and a* $(t' = O(t), q' = q_{\mathsf{exec}} + q_{\mathsf{PKE}})$*-IND-CCA2-adversary* $\mathcal{A}_5$ *against* PKE *such that:*

$$\mathsf{Adv}^{\mathsf{Sound}}_{\Pi}(\mathcal{A}) \leq \mathsf{n_C} \cdot \left(\frac{q_{\mathsf{exec}}}{2^{|\mathsf{Mac_S}|}} + \frac{1}{2^{\kappa}} + \mathsf{Adv}^{\mathsf{mac}}_{G}(\mathcal{A}_1) + \mathsf{Adv}^{\mathsf{prf}}_{G}(\mathcal{A}_2) + \mathsf{Adv}^{\mathsf{ind}}_{G}(\mathcal{A}_3) + \mathsf{Adv}^{\mathsf{ae}}_{\mathsf{AE}}(\mathcal{A}_4) + \mathsf{Adv}^{\mathsf{ind-cca2}}_{\mathsf{PKE}}(\mathcal{A}_5)\right).$$

Proof. **Game** $\mathbb{G}_0$: This game works as the game Sound-game stipulated in our security model. The goal of this adversary $\mathcal{A}_{\mathbb{G}_0}$ is similar as the S.Imp-game but with a different adversary; indeed in the S.Imp-game is a MiM adversary and in the Sound-game, we have a *legitimate-but-malicious* adversary. The adversary is a malicious, but legitimate server.

Game $\mathbb{G}_1$: We consider the game $\mathbb{G}_1$ as the S.Imp-game (as previously detailed) but including the specific query $\mathsf{Corrupt}(\mathsf{P}, \mathtt{type})$, i.e with the presence of operator keys corruption. We have used such a query in some previous security proofs. We proceed as follows: for any adversary $\mathcal{A}_{\mathbb{G}_0}$ winning the game $\mathbb{G}_0$ with a non-negligible success probability $\epsilon_{\mathbb{G}_0}$, we propose to construct a generic adversary $\mathcal{A}_{\mathbb{G}_1}$ winning the game $\mathbb{G}_1$ with a black-box access to the adversary $\mathcal{A}_{\mathbb{G}_0}$, also winning with non-negligible probability.

Both adversaries play these respective games. The following oracles are similar in the two games: Send, CreateCl, Init, Execute, Reveal, and StReveal. So for each query related to these oracles from the adversary $\mathcal{A}_{\mathbb{G}_0}$, the adversary $\mathcal{A}_{\mathbb{G}_1}$ forwards these queries to its own challenger and sends to $\mathcal{A}_{\mathbb{G}_0}$ the related answers. Now focus on the two last oracles which can be used by the adversary $\mathcal{A}_{\mathbb{G}_0}$: OpAccess and Corrupt.

Recall first that the OpAccess in the game $\mathbb{G}_0$ takes in input a client identifier and outputs, for our protocol, an authentication vector consisting of the tuple $\mathsf{AV} = (\mathsf{R}, \mathsf{Autn}, \mathsf{Mac_C}, \mathsf{CK}, \mathsf{IK})$. Simulating the answer of the oracle $\mathsf{OpAccess}(\mathsf{C}_i)$, the $\mathcal{A}_{\mathbb{G}_1}$ uses the query $\mathsf{Execute}(S, \mathsf{C}_i)$ (with the server related to the *legitimate-but-malicious* adversary) and $\mathsf{Reveal}(\mathsf{C}, i)$.

Now, focus on the simulation of the Corrupt answer. We recall that we have two possible inputs: a client or an operator. If the Corrupt oracle takes in input a client, the adversary $\mathcal{A}_{\mathbb{G}_1}$ uses its own Corrupt oracle to obtain the related answer. If the input is an operator, $\mathcal{A}_{\mathbb{G}_1}$ needs to forge the following values: the operator key $\mathsf{sk_{Op}}$, and for each client of this operator the tuple $(\mathsf{UID}, \mathsf{sk_{UID}}, \mathsf{st_{Op,C}})$. To simulate such an answer, $\mathcal{A}_{\mathbb{G}_1}$ uses its specific $\mathsf{Corrupt(C)}$ and $\mathsf{StReveal}(\mathsf{C}, i, 1)$ for each client C of this operator.

At this point, the adversary $\mathcal{A}_{\mathbb{G}_1}$ can simulate any query from the adversary $\mathcal{A}_{\mathbb{G}_0}$. At the end of the simulation, the adversary $\mathcal{A}_{\mathbb{G}_1}$ replays the impersonation attempt from $\mathcal{A}_{\mathbb{G}_0}$. Thus, we have:

$$\Pr[\mathcal{A}_{\mathbb{G}_0}\mathsf{wins}] \leq \Pr[\mathcal{A}_{\mathbb{G}_1}\mathsf{wins}].$$

Winning game $\mathbb{G}_1$: This game follows the game $\mathbb{G}_1$ described in the reduction proof of the

theorem S.Imp. Thus, it holds that:

$$\mathsf{Adv}^{\mathsf{Sound}}_{\Pi}(\mathcal{A}_{\mathbb{G}_0}) \leq \mathsf{n_C} \cdot \left(\frac{q_{\mathsf{exec}}}{2^{|\mathsf{Mac_S}|}} + \frac{1}{2^{\kappa}} + \mathsf{Adv}^{\mathsf{mac}}_{G}(\mathcal{A}_1) + \mathsf{Adv}^{\mathsf{prf}}_{G}(\mathcal{A}_2) + \mathsf{Adv}^{\mathsf{ind}}_{G}(\mathcal{A}_3) + \mathsf{Adv}^{\mathsf{ae}}_{\mathsf{AE}}(\mathcal{A}_4) + \mathsf{Adv}^{\mathsf{ind-cca2}}_{\mathsf{PKE}}(\mathcal{A}_5)\right).$$

□

4.7.3 Narrow-Forward Privacy is Impossible

Our variant of UMTS-AKA preserves the structure of the original protocol, but also provably attains wide-weak client unlinkability. In this section we show that this degree of client-unlinkability is optimal with respect to the structure of UMTS-AKA. In particular, narrow-forward privacy is impossible.

Our result covers similar ground as that by Paise and Vaudenay [93], as we address protocols with mutual authentication. We extend the impossibility result to symmetric-key AKE protocols which also use public-key primitives. We also explain why the original impossibility result in [93] is imprecise, and presents some problems.

The result of [93]. Paise and Vaudenay showed an impossibility result for *authentication* protocols – rather than AKE; the extension from one environment to the other is, however, easy. In the terminology of our analysis, [93] proved that server-authentication essentially precludes narrow-forward client-unlinkability. Their attack follows these steps: (1) the adversary $\mathcal{A}$ creates two clients; (2) $\mathcal{A}$ runs an honest protocol session between one of them (chosen uniformly at random depending on a secret bit b) and the server, but stops the last message from the server to the client; (3) $\mathcal{A}$ corrupts both clients, learning their long-term state; (4) $\mathcal{A}$ distinguishes between the clients by simulating the protocol with the intercepted message.

However, this attack makes a tacit assumption on the client's behaviour, namely that if a session is aborted, the state is not updated or that it is updated in a consistent way, depending on the client's internal state. Say that upon an abort, the client reverts to a random state; assumming that the adversary cannot access the very short time-frame in which the reversion to random is done, $\mathcal{A}$ only gets the random state in response. Simulating the protocol with the received message will not match that state, thus reducing the adversary's success probability to $\frac{1}{2}$.

Another way to bypass this result is to update the client state before the "last message" is sent to the client; if such an update is done at every execution, the attack presented in [93] fails. This is, however, a rather artificial twist: indeed, mutual authentication implies that the prover *must* somehow identify the server's state as "valid" *before* it reaches a state which precludes it from verifying the server's authentication.

Two new attacks. In the UMTS-AKA protocol, it is the server which first authenticates to the client. The values used in authentication are the sequence number $\mathsf{Sqn}_{\mathsf{Op,C}}$ and the nonce R. The value $\mathsf{Sqn}_{\mathsf{Op,C}}$ is ephemeral, being updated at every session; however, it is a long-term state, and compatible with the client's own state $\mathsf{Sqn_C}$. In particular, corrupting a client yields $\mathsf{Sqn_C}$ allowing the verifier to link the client with the corresponding $\mathsf{Sqn}_{\mathsf{Op,C}}$ value.

For a better comprehension of our attacks and their impact, we define the following notations: we divide a party's state (for both clients and operators) into a *static* state $\mathsf{stat.st_P}$ and an ephemeral state $\mathsf{eph.st_P}$. Thus, an operator's static state may contain operator-specific information, such as the secret key for a PKE scheme, but it will also include state shared with clients, i.e. $\mathsf{stat.st}_{\mathsf{Op,C}}$ for a client C. The same for the ephemeral state $\mathsf{eph.st}_{\mathsf{Op,C}}$ which for the UMTS-AKA protocol consists of the sequence number $\mathsf{Sqn}_{\mathsf{Op,C}}$. We propose the following attack:

- The adversary $\mathcal{A}$ creates two clients C and C' with the same operator Op and the same location LAI.

- $\mathcal{A}$ uses DrawCl on C, C' and receives the handle VC, corresponding to either C (if the hidden bit is $b = 0$), or C' (otherwise).

- $\mathcal{A}$ runs an honest execution between the server S at LAI and the client VC until $\mathcal{A}$ receives the message $\mathsf{R}, \mathsf{Autn} = (\mathsf{Sqn}_{\mathsf{Op,C}} \oplus \mathsf{AK}) \| \mathsf{AMF} \| \mathsf{Mac}_{\mathsf{S}}$ from the server. Denote $\mathsf{Autn}[1] := \mathsf{Sqn}_{\mathsf{Op,C}} \oplus \mathsf{AK}$, $\mathsf{Autn}[2] = \mathsf{AMF}$, and $\mathsf{Autn}[3] := \mathsf{Mac}_{\mathsf{S}}$.

- $\mathcal{A}$ corrupts C and learns $\mathsf{Sqn}_{\mathsf{C}}$, sk_{C}, and $\mathsf{sk}_{\mathsf{Op}}$.

- By using the values $\mathsf{R}, \mathsf{sk}_{\mathsf{C}}$, and $\mathsf{sk}_{\mathsf{Op}}$, the adversary computes a value AK_{C} and retrieves $\mathsf{Sqn}^* := \mathsf{Autn}[1] \oplus \mathsf{AK}_{\mathsf{C}}$. Note that if $b = 0$, then $\mathsf{AK}_{\mathsf{C}} = \mathsf{AK}$ and $\mathsf{Sqn}^* = \mathsf{Sqn}_{\mathsf{Op,C}}$, while if $b = 1$, then $\mathsf{Sqn}^* \neq \mathsf{Sqn}_{\mathsf{Op,C}'}$ with overwhelming probability.

- The adversary verifies $\mathsf{Mac}_{\mathsf{S}}$, on input $(\mathsf{sk}_{\mathsf{C}}, \mathsf{sk}_{\mathsf{Op}}, \mathsf{Sqn}^*)$. If this verification succeeds, the adversary outputs a guess $d = 0$ for the bit b; else, it outputs 1.

For the analysis note that with overwhelming probability $\mathsf{Mac}_{\mathsf{S}}$ will not verify if it was computed for C', i.e. $f_1(\mathsf{sk}_{\mathsf{C}'}, \mathsf{sk}_{\mathsf{Op}}, \mathsf{R}, \mathsf{AMF}, \mathsf{Sqn}_{\mathsf{Op,C}'}) \neq f_1(\mathsf{sk}_{\mathsf{C}}, \mathsf{sk}_{\mathsf{Op}}, \mathsf{R}, \mathsf{AMF}, \mathsf{Sqn}^*)$. The key vulnerability here is that, while $\mathsf{Sqn}_{\mathsf{Op,C}}$ is never sent in clear, the masking authentication key AK only depends on the client's static state. We also use the fact that the validity of the sequence number is confirmed by the value $\mathsf{Mac}_{\mathsf{S}}$.

While the latter factor, namely the validity of $\mathsf{Mac}_{\mathsf{S}}$ certainly helps an attacker, our second attack (a variation of the first one) does not use the MAC value at all. The attack is run exactly in the same way, until we reach the final step. At that point:

- $\mathcal{A}$ compares the obtained value Sqn^* with the recovered sequence number of C, namely $\mathsf{Sqn}_{\mathsf{C}}$, verifying if $|\mathsf{Sqn}^* - \mathsf{Sqn}_{\mathsf{Op,C}}| \leq \Delta$. Note that in the actual attack presented above, the client's state $\mathsf{Sqn}_{\mathsf{C}}$ should be exactly equal to the operator's state with respect to that client; however, our attack is even stronger in the sense that we do not need to control the executions of the protocol in order to obtain exact equality.

Analysis and Impact. Since the original UMTS-AKA protocol is not even weak-client-unlinkable, it is not surprising that that protocol is not narrow-forward unlinkable either. However, the same attack works on our variant of the protocol and indeed, on any other extension or improvement of the original procedure which retains the characteristic of exchanging a message of the type $f(\mathsf{eph.st}_{\mathsf{Op,C}}, \mathsf{stat.st}_{\mathsf{Op,C}}, X)$ in the presence of a function $Match$; or exchanging that same message together with a message $g(\mathsf{eph.st}_{\mathsf{Op,C}}, \mathsf{stat.st}_{\mathsf{Op,C}}, Y)$, such that:

- f is reversible and takes as input $\mathsf{eph.st}_{\mathsf{Op,C}}, \mathsf{stat.st}_{\mathsf{Op,C}} = \mathsf{stat.st}_{\mathsf{C}}$, and a set X of publicly-known variables, giving arbitrary values in the set $\{0,1\}^*$[4];

- $Match$ takes as input two ephemeral state values $\mathsf{eph.st}_{\mathsf{C}'}$ and $\mathsf{eph.st}_{\mathsf{Op,C}}$ and it outputs a boolean value: 1 if $\mathsf{C} = \mathsf{C}'$ and 0 otherwise[5];

- g takes as input the state values $\mathsf{eph.st}_{\mathsf{Op,C}}, \mathsf{stat.st}_{\mathsf{C}}$ and a set Y of public values, and which has the property that, for randomly chosen x and $\mathsf{stat.st}_{\mathsf{C}'}$ it holds that $g(x, \mathsf{stat.st}_{\mathsf{Op,C}'}, Y) \neq g(\mathsf{eph.st}_{\mathsf{Op,C}}, \mathsf{stat.st}_{\mathsf{Op,C}}, Y)$[6].

[4] In our previous example, this is the string Autn1, which depends on $\mathsf{eph.st}_{\mathsf{Op,C}} = \mathsf{Sqn}_{\mathsf{Op,C}}$, on $\mathsf{stat.st}_{\mathsf{Op,C}} = (\mathsf{sk}_{\mathsf{C}}, \mathsf{sk}_{\mathsf{Op}})$, and on the random value R which is public.

[5] In our case, the $Match$ function returns 1 if and only if $|\mathsf{Sqn}_{\mathsf{Op,C}} - \mathsf{Sqn}_{\mathsf{C}'}| \leq \Delta$.

[6] In our example, this function is f_1, and the output value is $\mathsf{Mac}_{\mathsf{S}}$.

4.7.4 Security of the internal cryptographic functions TUAK and MILENAGE

Updated TUAK algorithms security.

In order to prove the pseudorandomess, unforgeability, and indistinguishability of our updated TUAK algorithms, we assume that the truncated keyed internal Keccak permutation is a good pseudorandom function. We propose two generic constructions to model the updated TUAK algorithms: a first one, denoted G_{tuak}, when the secret is based on the subscriber key sk and a second one, denoted G^*_{tuak} when is only based on the operator key.

It is worth noting that the construction of the TUAK functions is reminiscent of the Merkle-Damgård construction, where the output of the function f is an input of the next iteration of the function f. This is in contradiction with the Sponge construction used in the hash function Keccak given the internal permutation $\mathsf{f}_{\mathsf{Keccak}}$.

We model the truncated keyed internal permutation of Keccak by the function f and f^*:

$$f(K, x\|y, i, j) = \lfloor \mathsf{f}_{\mathsf{Keccak}}(x\|K\|y)\rfloor_{i..j}, f^*(K^*, x^*\|y^*, i, j) = \lfloor \mathsf{f}_{\mathsf{Keccak}}(K^*\|x^*\|y^*)\rfloor_{i..j},$$

with $x \in \{0,1\}^{512}$, $K, K^* \in \{0,1\}^{\kappa}$, $y \in \{0,1\}^{1088-\kappa}$, $x^* \in \{0,1\}^{512+\kappa}$, $y^* \in \{0,1\}^{1088}$ and $i, j \in \{0,1\}^t$ with $\log_2(t-1) < 1600 \leq \log_2(t)$. We note that $\forall(K, x, x^*, y, y^*, i, j)$ such that $x = K^*\|x^*$ and $y^* = K\|y$, we have $f(K, x\|y, i, j) = f^*(K^*, x^*\|y^*, i, j)$. The input x (resp. x^*) can be viewed as the chaining variable of the cascade construction of G_{tuak} given f (resp. f^*), y (resp. y^*) is an auxiliary input of the function, and i and j define the size of the truncation. The construction G_{tuak} acts as a generalization of the specific TUAK algorithms:

$$\begin{aligned}
\mathcal{F}_1(\mathsf{sk_{Op}}, \mathsf{sk_C}, \mathsf{R}, \mathsf{Sqn}, \mathsf{Id_S}, \mathsf{AMF}) &= G_{\mathsf{tuak}}(\mathsf{sk_C}, \mathsf{inp}_1, 0, 127) = G^*_{\mathsf{tuak}}(\mathsf{sk_{Op}}, \mathsf{inp}^*_1, 0, 127),\\
\mathcal{F}_2(\mathsf{sk_{Op}}, \mathsf{sk_C}, \mathsf{R}, \mathsf{Sqn}, \mathsf{Id_S}, \mathsf{AMF}) &= G_{\mathsf{tuak}}(\mathsf{sk_C}, \mathsf{inp}_1, 256, 383) = G^*_{\mathsf{tuak}}(\mathsf{sk_{Op}}, \mathsf{inp}^*_1, 256, 383),\\
\mathcal{F}_3(\mathsf{sk_{Op}}, \mathsf{sk_C}, \mathsf{R}, \mathsf{Sqn}, \mathsf{Id_S}, \mathsf{AMF}) &= G_{\mathsf{tuak}}(\mathsf{sk_C}, \mathsf{inp}_1, 512, 639) = G^*_{\mathsf{tuak}}(\mathsf{sk_{Op}}, \mathsf{inp}^*_1, 512, 639),\\
\mathcal{F}_4(\mathsf{sk_{Op}}, \mathsf{sk_C}, \mathsf{R}, \mathsf{Sqn}, \mathsf{Id_S}, \mathsf{AMF}) &= G_{\mathsf{tuak}}(\mathsf{sk_C}, \mathsf{inp}_1, 768, 895) = G^*_{\mathsf{tuak}}(\mathsf{sk_{Op}}, \mathsf{inp}^*_1, 768, 895),\\
\mathcal{F}_5(\mathsf{sk_{Op}}, \mathsf{sk_C}, \mathsf{R}, \mathsf{Id_S}) &= G_{\mathsf{tuak}}(\mathsf{sk_C}, \mathsf{inp}_2, 0, 47) = G^*_{\mathsf{tuak}}(\mathsf{sk_C}, \mathsf{inp}^*_2, 0, 47),
\end{aligned}$$

with:
$\mathsf{inp}_1 = \mathsf{sk_{Op}}\|\mathsf{cst}_1\|\mathsf{cst}_3$, $\mathsf{inp}_2 = \mathsf{sk_{Op}}\|\mathsf{cst}_1\|\mathsf{cst}_3$, $\mathsf{inp}^*_1 = \mathsf{cst}_1\|\mathsf{keys}\|\mathsf{cst}_3$, $\mathsf{inp}^*_2 = \mathsf{cst}_1\|\mathsf{keys}\|\mathsf{cst}_3$,
$\mathsf{cst}_1 = \mathsf{Inst}\|\mathsf{AN}\|0^{192}\|(\mathsf{Inst}'\|\mathsf{AN}\|\mathsf{R}\|\mathsf{AMF}\|\mathsf{Sqn})$, $\mathsf{cst}_3 = \mathsf{Id_S}\|\mathsf{Pad}\|1\|0^{192}$,
We define the cascade constructions G_{tuak} and G^*_{tuak} based on the function f and f^* as follows:

$$\begin{aligned}
G_{\mathsf{tuak}}(K, \mathsf{val}, i, j) &= f(K, f(K, \mathsf{val}_1\|\mathsf{val}_3, 0, 256)\|\mathsf{val}_2\|\mathsf{val}_3, i, j),\\
G^*_{\mathsf{tuak}}(K^*, val^*, i, j) &= f^*(f^*, \mathsf{val}^*_1\|\mathsf{val}^*_3, 0, 256), \mathsf{val}^*_2\|\mathsf{val}^*_3, i, j),
\end{aligned}$$

with G_{tuak} and G^*_{tuak} from $\{0,1\}^{\kappa}\times\{0,1\}^{d}\times\{0,1\}^{t}\times\{0,1\}^{t}$ to $\{0,1\}^{n}$, $\mathsf{val} = (\mathsf{val}_1\|\mathsf{val}_2)\|\mathsf{val}_3 \in \{0,1\}^{512}\times\{0,1\}^{256}\times\{0,1\}^{(832-\kappa)}$, $\mathsf{val}^* = (\mathsf{val}^*_1\|\mathsf{val}^*_2)\|\mathsf{val}^*_3 \in \{0,1\}^{256}\times\{0,1\}^{256}\times\{0,1\}^{(1088-\kappa)}$ two known values with $n = j - i$, $d = 1600 - \kappa$, $\kappa = |K|$ and $\log_2(t-1) < 1600 \leq \log_2(t)$, K a secret value and $0 \leq i \leq j \leq 1600$.

We express the required security properties of the generalization G_{tuak} (resp. G^*_{tuak}) assuming the prf-security of the function f (resp. f^*). Although we cannot prove the latter property, we can conjecture that the advantage of a prf-adversary would be of the form:

$$\mathsf{Adv}^{\mathsf{prf}}_{f^*}(\mathcal{A}) = \mathsf{Adv}^{\mathsf{prf}}_{f}(\mathcal{A}) \leq \mathsf{c}_1 \cdot \frac{t/T_f}{2^{|K|}} + \mathsf{c}_2 \cdot \frac{q \cdot t/T_f}{2^{1600-m}},$$

for any adversary $\mathcal{A}$ running in time t and making at most q queries at its challenger. Here, m is the output's size of our function f and T_f is the time to do one f computation on the fixed RAM model of computation and c_1 and c_2 are two constants depending only on this model. In other words, we assume that the best attacks are either a exhaustive key search or a specific attack on this construction. This attack uses the fact that the permutation is public and can be easily inverted. Even if the protocol truncates the permutation, if the output values are large, and an exhaustive search on the missing bits is performed, it is possible to invert the permutation and recover the inputs. Since the secret keys is one of the inputs as well as some known values are also inputs, it is then possible to determine which guesses of the exhaustive search are correct guess or incorrect ones. Finally, if the known inputs are shorter than the truncation, false positives can happen due to collisions and we have to filter the bad guesses. However, if the number of queries is large enough, it is possible to filter these bad guesses and uniquely recover the keys.

Pseudorandomness and Unforgeability of the TUAK algorithms. We begin by reducing the prf-security of G_{tuak} to the prf-security of the function f. This implies the mac-security of each TUAK algorithm. Recall that our main assumption is that the function f is prf-secure if the Keccak permutation is a good random permutation.

Theorem 25. $[\mathsf{prf}-\textbf{security for } \mathbf{G}^*_{\mathsf{tuak}}.]$ *Let* $G^*_{\mathsf{tuak}} : \{0,1\}^\kappa \times \{0,1\}^e \times \{0,1\}^{d-e} \times \{0,1\}^t \times \{0,1\}^t \to \{0,1\}^n$ *and* $f^* : \{0,1\}^\kappa \times \{0,1\}^e \times \{0,1\}^d \times \{0,1\}^t \times \{0,1\}^t \to \{0,1\}^m$ *be the two functions specified above. Consider a* (t,q)*-adversary* $\mathcal{A}$ *against the* prf*-security of the function* G^*_{tuak}*, running in time* t *and making at most* q *queries to its challenger. Denote the advantage of this adversary as* $\mathsf{Adv}^{\mathsf{prf}}_{G^*_{\mathsf{tuak}}}(\mathcal{A})$*. Then there exists a* $(t' \approx O(t), q' = q)$*-adversary* $\mathcal{A}$*' with an advantage* $\mathsf{Adv}^{\mathsf{prf}}_{f^*}(\mathcal{A}')$ *of winning against the pseudorandomness of* f^* *such that:*

$$\mathsf{Adv}^{\mathsf{prf}}_{G^*_{\mathsf{tuak}}}(\mathcal{A}) = \mathsf{Adv}^{\mathsf{prf}}_{f^*}(\mathcal{A}'),$$

Proof. We construct the adversary $\mathcal{A}_{f^*}$ using a prf-adversary $\mathcal{A}_{G^*}$. The latter uses $\mathcal{A}_{f^*}$ as a challenger for a prf-game $\mathbb{G}_{\mathsf{prf}(f^*)}$ and can only communicate with $\mathcal{A}_{f^*}$ whereas $\mathcal{A}_{f^*}$ has access to a challenger for f^*. To begin with, the challenger $\mathcal{C}^{\mathsf{prf}}_{f^*}$ chooses a bit b and a private $\mathsf{sk}_{\mathsf{Op}} \in \{0,1\}^\kappa$. If $b = 0$, it assigns f^* to a random function and if $b = 1$, it assigns f^* to the specific internal function.

The adversary $\mathcal{A}_{f^*}$ waits for queries from $\mathcal{A}_{G^*}$ of the form (m, a, b), with $m = m^{(1)}\|m^{(2)}\|m^{(3)}$ $\in \{0,1\}^d$, and $a, b \in \{0,1\}^t$ and responds as follows:

- It queries its challenger $\mathcal{C}^{\mathsf{prf}}_{f^*}$ for inputs $(m^{(1)}\|m^{(3)}, 0, 256)$ and receives the value Out_1.
- Then, it computes $\mathsf{Out}_2 = f^*(\mathsf{Out}_1, m^{(2)}\|m^{(3)}, a, b)$.
- It returns the value Out_2.

We note that the two first bullets permits to generate $G^*(\mathsf{sk}_{\mathsf{Op}}, m, a, b) = \mathsf{Out}_2$. This step is repeated up to a total of q queries from $\mathcal{A}_{g^*}$, with a and b fixed.

At some point, $\mathcal{A}_{g^*}$ halts and outputs a guess d of the bit b. The prf-adversary $\mathcal{A}_{f^*}$ chooses its guess b' as $b' = d$ and forwards it to $\mathcal{C}^{\mathsf{prf}}_{f^*}$, which verifies if $b = b'$.

We analyze this simulation. Recall that the challenger responded either with a random value (if its internal bit b was set to 0) or with the output of the function $f^*(\mathsf{sk}_{\mathsf{Op}}, m^{(1)}\|m^{(3)}, 0, 256)$ (if its internal bit was set as 1).

Thus, the output Out_2 matches either the output of a random function or the output of the function $G^*(\mathsf{sk}, m, a, b)$. So the prf-adversary $\mathcal{A}_{f^*}$ simulates perfectly a prf-challenger of G.

Thus, we have:

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{prf}}_{f^*}(\mathcal{A}_{f^*}) &= \big|\Pr[\mathcal{A}_{f^*} \to 1 \mid b = 1] - \Pr[\mathcal{A}_{f^*} \to 1 \mid b = 0]\big| \\
&= \big|\Pr[b = b'|b = 1] - \Pr[b = b'|b = 0]\big| \\
&= \big|\Pr[b = d|b = 1] - \Pr[b = d|b = 0]\big| \\
&= \big|\Pr[d' = d|b = 1] - \Pr[d' = d|b = 0]\big| \\
&= \big|\Pr[\mathcal{A}_{G^*} \to 1 \mid b = 1] - \Pr[\mathcal{A}_{G^*} \to 1 \mid b = 0]\big| \\
&= \big|\mathsf{Adv}^{\mathsf{prf}}_{G^*}(\mathcal{A}_{G^*})\big|.
\end{aligned}$$

Theorem 26. [$\mathsf{prf} - \mathbf{security\ for\ G_{tuak}}$.] *Let* $G_{\mathsf{tuak}} : \{0,1\}^\kappa \times \{0,1\}^e \times \{0,1\}^{d-e} \times \{0,1\}^t \times \{0,1\}^t \to \{0,1\}^n$ *and* $f : \{0,1\}^\kappa \times \{0,1\}^e \times \{0,1\}^d \times \{0,1\}^t \times \{0,1\}^t \to \{0,1\}^m$ *be the two functions specified above. Consider a* (t, q)*-adversary* $\mathcal{A}$ *against the* prf*-security of the function* G*, running in time* t *and making at most* q *queries to its challenger. Denote the advantage of this adversary as* $\mathsf{Adv}^{\mathsf{prf}}_{G_{\mathsf{tuak}}}(\mathcal{A})$*. Then there exists a* $(t' \approx 2 \cdot t, q' = 2 \cdot q)$*-adversary* $\mathcal{A}$*' with an advantage* $\mathsf{Adv}^{\mathsf{prf}}_{f}(\mathcal{A}')$ *of winning against the pseudorandomness of* f *such that:*

$$\mathsf{Adv}^{\mathsf{prf}}_{G_{\mathsf{tuak}}}(\mathcal{A}) = \mathsf{Adv}^{\mathsf{prf}}_{f}(\mathcal{A}'),$$

Proof. We construct the adversary $\mathcal{A}_f$ using a prf-adversary $\mathcal{A}_G$. The latter uses $\mathcal{A}_f$ as a challenger for a prf-game $\mathbb{G}_{\mathsf{prf}(f)}$ and can only communicate with $\mathcal{A}_f$ whereas $\mathcal{A}_f$ has access to a challenger for f. To begin with, the challenger $\mathcal{C}^{\mathsf{prf}}_f$ chooses a bit b and a private $\mathsf{sk} \in \{0,1\}^\kappa$. If $b = 0$, it assigns f to a random function and if $b = 1$, it assigns f to the specific internal function.

The adversary $\mathcal{A}_f$ waits for queries from $\mathcal{A}_G$ of the form (m, a, b), with $m = m^{(1)}\|m^{(2)}\|m^{(3)} \in \{0,1\}^d$, and $a, b \in \{0,1\}^t$ and responds as follows:

- It queries its challenger $\mathcal{C}^{\mathsf{prf}}_f$ for inputs $(m^{(1)}\|m^{(3)}, 0, 256)$ and receives the value Out_1.
- Then, it queries $\mathcal{C}^{\mathsf{prf}}_f$ for inputs $(\mathsf{Out}_1\|m^{(2)}\|m^{(3)}, a, b)$ and receives the value Out_2.
- It returns the value Out_2.

We note that the two first bullets permits to generate $G(\mathsf{sk}, m, a, b)$ computing $f(\mathsf{sk}, f(\mathsf{sk}, m^{(1)}\|m^{(3)}, 0, 256)\|m^{(2)}\|m^{(3)}, a, b)$. This step is repeated up to a total of q queries from $\mathcal{A}_g$, with a and b fixed.

At some point, $\mathcal{A}_g$ halts and outputs a guess d of the bit b. The prf-adversary $\mathcal{A}_f$ chooses its guess b' as $b' = d$ and forwards it to $\mathcal{C}^{\mathsf{prf}}_f$, which verifies if $b = b'$.

We analyze this simulation. Recall that the challenger responded either with a random value (if its internal bit b was set to 0) or with the output of the function $f(\mathsf{sk}, m^{(1)}\|m^{(3)}, 0, 256)$ (if its internal bit was set as 1).

Thus, the output Out_2 matches either the output of a random function or the output of the function $G(\mathsf{sk}, m, a, b)$. So the prf-adversary $\mathcal{A}_f$ simulates perfectly a prf-challenger of G. Thus, we have:

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{prf}}_{f}(\mathcal{A}_f) &= \big|\Pr[\mathcal{A}_f \to 1 \mid b = 1] - \Pr[\mathcal{A}_f \to 1 \mid b = 0]\big| \\
&= \big|\Pr[b = b'|b = 1] - \Pr[b = b'|b = 0]\big| \\
&= \big|\Pr[b = d|b = 1] - \Pr[b = d|b = 0]\big| \\
&= \big|\Pr[d' = d|b = 1] - \Pr[d' = d|b = 0]\big| \\
&= \big|\Pr[\mathcal{A}_G \to 1 \mid b = 1] - \Pr[\mathcal{A}_G \to 1 \mid b = 0]\big| \\
&= \big|\mathsf{Adv}^{\mathsf{prf}}_{G}(\mathcal{A}')\big|.
\end{aligned}$$

□

We use the generic result specified in 1 to reduce the mac-secure to the prf-secure of the function G.

Theorem 27. [$\mathsf{Mac}-$**security for** $\mathbf{G}_{\mathsf{tuak}}$.] *Let* $G_{\mathsf{tuak}} : \{0,1\}^{\kappa} \times \{0,1\}^{e} \times \{0,1\}^{d-e} \times \{0,1\}^{t} \times \{0,1\}^{t} \rightarrow \{0,1\}^{n}$ *and* $f : \{0,1\}^{\kappa} \times \{0,1\}^{e} \times \{0,1\}^{d} \times \{0,1\}^{t} \times \{0,1\}^{t} \rightarrow \{0,1\}^{m}$ *be the two functions specified above. Consider a* (t,q)*-adversary* $\mathcal{A}$ *against the* Mac*-security of the function* G_{tuak}*, running in time* t *and making at most* q *queries to its challenger. Denote the advantage of this adversary as* $\mathsf{Adv}^{\mathsf{mac}}_{G_{\mathsf{tuak}}}(\mathcal{A})$*. Then there exists a* $(t' \approx 2 \cdot (t + O(n+d)), q' = 2 \cdot q)$*-adversary* $\mathcal{A}$*' with an advantage* $\mathsf{Adv}^{\mathsf{prf}}_{f}(\mathcal{A}')$ *of winning against the pseudorandomness of* f *such that:*

$$\mathsf{Adv}^{\mathsf{mac}}_{G_{\mathsf{tuak}}}(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{prf}}_{f}(\mathcal{A}') + \frac{1}{2^n}.$$

Indistinguishability of the TUAK **algorithms.** We begin by reducing the ind-security of G_{tuak} to the prf-security of the function f. This implies the ind-security of each TUAK algorithm. Recall that our main assumption is that the function f is prf-secure if the Keccak permutation is a good random permutation.

Theorem 28. [$\mathsf{ind}-$**security for** $\mathbf{G}_{\mathsf{tuak}}$.] *Let* $G_{\mathsf{tuak}} : \{0,1\}^{\kappa} \times \{0,1\}^{e} \times \{0,1\}^{d-e} \times \{0,1\}^{t} \times \{0,1\}^{t} \rightarrow \{0,1\}^{n}$ *and* $f : \{0,1\}^{\kappa} \times \{0,1\}^{d} \times \{0,1\}^{t} \times \{0,1\}^{t} \rightarrow \{0,1\}^{m}$ *be the two functions specified in above. Consider a* (t,q)*-adversary* $\mathcal{A}$ *against the* ind*-security of the function* G*, running in time* t *and making at most* q *queries to its challenger. Denote the advantage of this adversary as* $\mathsf{Adv}^{\mathsf{ind}}_{G_{\mathsf{tuak}}}(\mathcal{A})$*. Then there exists a* $(t' \approx 2 \cdot t, q' = 2 \cdot q)$*-adversary* $\mathcal{A}$*' with an advantage* $\mathsf{Adv}^{\mathsf{prf}}_{f}(\mathcal{A}')$ *of winning against the pseudorandomness of* f *such that:*

$$\mathsf{Adv}^{\mathsf{ind}}_{G_{\mathsf{tuak}}}(\mathcal{A}) = \mathsf{Adv}^{\mathsf{prf}}_{f}(\mathcal{A}').$$

Proof. To prove the ind-security of G, we reduce this security to the prf-security of G which is defined in the proof of the theorem 27. We show that a prf-adversary $\mathcal{A}$ of G can simulate the ind-challenger $\mathcal{C}^{\mathsf{ind}}_{G}$. A such prf-adversary behaves as follows. At first, the challenger $\mathcal{C}^{\mathsf{prf}}_{G}$ chooses a private key K and one random bit $b \in \{0,1\}$. If $b = 0$, he assigns f to a random function and if $b = 1$, he assigns f to the specific function G. For each query $(M^{\{i\}}, a, b)$ with a fixed (a, b) from the ind-adversary $\mathcal{A}$' to the prf-adversary $\mathcal{A}$ the latter forwards each one to $\mathcal{C}^{\mathsf{prf}}_{G}$. The answer $f(K, M^{\{i\}}, a, b)$ is sent to $\mathcal{A}$ which forwards it to $\mathcal{A}$'. Then, $\mathcal{A}$' sends a specific query containing two values (M_0, M_1) to $\mathcal{A}$The latter chooses randomly a bit d and forwards (M_d, a, b) to the prf-challenger. As usual, this challenger sends $f(K, M_d, a, b)$ to $\mathcal{A}$ which forwards it to $\mathcal{A}$'.

The goal of the ind-adversary is to find the bit d chosen by the $\mathcal{A}$. To do so, it can ask again some queries $(M^{\{i\}}, a, b)$ as previously. Finally, it sends its guessing d' to $\mathcal{A}$. Upon receiving this guessing, it chooses its guessing b' of b as follows: if $d = d'$, it chooses $b' = 1$, else $b' = 0$.

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{prf}}_{G}(\mathcal{A}) &= \big|\Pr[b = b' | f \xleftarrow{\$} G(K,.,.,.), K \xleftarrow{\$} \{0,1\}^{\kappa}] - \Pr[b = b' | f \xleftarrow{\$} \mathsf{R}_{]}\big| \\
&= \big|\Pr[b = b' | b = 1] - \Pr[b = b' | b = 0]\big| \\
&= \big|\Pr[b' = 1 | b = 1] - \Pr[b' = 0 | b = 0]\big| \\
&= \big|\Pr[d' = d | b = 1] - \Pr[d' \neq d | b = 0]\big| \\
&= \big|\mathsf{Adv}^{\mathsf{ind}}_{G}(\mathcal{A}') + \tfrac{1}{2} - \tfrac{1}{2}\big| = \big|\mathsf{Adv}^{\mathsf{ind}}_{G}(\mathcal{A}')\big|
\end{aligned}$$

In the last equality, $\Pr[d' = d | b = 1]$ is the probability that the ind-adversary correctly guess the bit d which is $\mathsf{Adv}^{\mathsf{ind}}_{G}(\mathcal{A}') + 1/2$ and $\Pr[d' \neq d | b = 0]$ which is equal to $1/2$ since when $b = 0$, G is a random function, that is its output is chosen independently from its inputs. Consequently, it is not related to its inputs and the adversary cannot guess correctly the bit d. □

The security of the Updated MILENAGE algorithms.

In order to prove the unforgeability and indistinguishability properties of the MILENAGE algorithms, we assume that the AES permutation is a good pseudorandom function.

We model the AES algorithm by the function f, as follows:

$$f(K, x, y) = \mathsf{AES}_K(x \oplus y),$$

with $x, y \in \{0,1\}^{128}$, $K \in \{0,1\}^{\kappa}$.

Contrary to the TUAK algorithms, the MILENAGE algorithms do not behave as the same way but as two different ways. Let the construction G_{mil1}, the generalization of the functions $\mathcal{F}_1$, $\mathcal{F}_1^*$, $\mathcal{F}_2$, $\mathcal{F}_3$, $\mathcal{F}_4$ and G_{mil2} the generalization of the functions $\mathcal{F}_5$ and $\mathcal{F}_5^*$ as follows:

$$\begin{aligned}
G_{\mathsf{mil1}}(\mathsf{sk_C}, \mathsf{inp}_1, 0, 127) &= \mathcal{F}_1(\mathsf{sk_{Op}}, \mathsf{sk_C}, \mathsf{R}, \mathsf{Sqn}, \mathsf{Id_S}, \mathsf{AMF}),\\
G_{\mathsf{mil1}}(\mathsf{sk_C}, \mathsf{inp}_2, 0, 127) &= \mathcal{F}_1^*(\mathsf{sk_{Op}}, \mathsf{sk_C}, \mathsf{R}, \mathsf{Sqn}, \mathsf{Id_S}, \mathsf{AMF}),\\
G_{\mathsf{mil1}}(\mathsf{sk_C}, \mathsf{inp}_3, 0, 47) &= \mathcal{F}_2(\mathsf{sk_{Op}}, \mathsf{sk_C}, \mathsf{R}, \mathsf{Sqn}, \mathsf{Id_S}, \mathsf{AMF}),\\
G_{\mathsf{mil1}}(\mathsf{sk_C}, \mathsf{inp}_4, 0, 127) &= \mathcal{F}_3(\mathsf{sk_{Op}}, \mathsf{sk_C}, \mathsf{R}, \mathsf{Sqn}, \mathsf{Id_S}, \mathsf{AMF}),\\
G_{\mathsf{mil1}}(\mathsf{sk_C}, \mathsf{inp}_5, 0, 127) &= \mathcal{F}_4(\mathsf{sk_{Op}}, \mathsf{sk_C}, \mathsf{R}, \mathsf{Sqn}, \mathsf{Id_S}, \mathsf{AMF}),\\
G_{\mathsf{mil2}}(\mathsf{sk_C}, \mathsf{inp}_6, 0, 47) &= \mathcal{F}_5(\mathsf{sk_{Op}}, \mathsf{sk_C}, \mathsf{R}, \mathsf{Id_S}),\\
G_{\mathsf{mil2}}(\mathsf{sk_C}, \mathsf{inp}_6, 64, 111) &= \mathcal{F}_5^*(\mathsf{sk_{Op}}, \mathsf{sk_C}, \mathsf{R}, \mathsf{Id_S}),
\end{aligned}$$

with: $\forall i \in \{1, ..., 5\}, \mathsf{inp}_i = \mathsf{sk_{Op}} \| \mathsf{R} \| (\mathsf{Sqn} \| \mathsf{AMF}) \| \mathsf{Id_S} \| \mathsf{c}_i \| \mathsf{r}_i$,, $\mathsf{inp}_6 = \mathsf{sk_{Op}} \| \mathsf{R} \| \mathsf{Id_S} \| \mathsf{c}_6 \| \mathsf{r}_6$,

Both constructions are constructed as follows:

$$G_{\mathsf{mil1}}(K, \mathsf{val}^{(1)}, a, b) = \lfloor \mathsf{Top}_C \oplus f(K, \mathsf{val}_4, f(K, \mathsf{Top}_C, \mathsf{val}_2) \oplus \mathsf{Rot}_{\mathsf{val}_5}(\mathsf{Top}_C \oplus (\mathsf{val}_3 \| \mathsf{val}_3))) \rfloor_{a..b},$$

$$G_{\mathsf{mil2}}(K, \mathsf{val}^{(2)}, a, b) = \lfloor \mathsf{Top}_C \oplus f(K, \mathsf{val}_4, \mathsf{Rot}_{\mathsf{val}_5}(\mathsf{Top}_C \oplus f(K, \mathsf{Top}_C, \mathsf{val}_2))) \rfloor_{a..b},$$

with $G_{\mathsf{mil1}} : \{0,1\}^{\kappa} \times \{0,1\}^{d_1} \times \{0,1\}^t \times \{0,1\}^t \to \{0,1\}^n$, $G_{\mathsf{mil2}} : \{0,1\}^{\kappa} \times \{0,1\}^{d_2} \times \{0,1\}^t \times \{0,1\}^t \to \{0,1\}^n$, and $\mathsf{val}^{(1)} = \mathsf{val}_1 \| \mathsf{val}_2 \| \mathsf{val}_3 \| \mathsf{val}_4 \| \mathsf{val}_5$, $\mathsf{val}^{(2)} = \mathsf{val}_1 \| \mathsf{val}_2 \| \mathsf{val}_4 \| \mathsf{val}_5$, $\mathsf{val}_1, \mathsf{val}_2, \mathsf{val}_4 \in \{0,1\}^{128}$,$\mathsf{val}_3 \in \{0,1\}^{64}$, $\mathsf{val}_5 \in \{0,1\}^7$ and $\mathsf{Top}_C = val_1 \oplus f(K, val_1, 0)$.

We express the security properties of the generalizations G_{mil1} and G_{mil2} assuming the prf-security of the function f. While we cannot prove the latter property, we can conjecture that the advantage of a prf-adversary would be of the form:

$$\mathsf{Adv}_f^{\mathsf{prf}}(\mathcal{A}) \leq \mathsf{c}_1 \cdot \frac{t/T_f}{2^{128}} + \mathsf{c}_2 \cdot \frac{q^2}{2^{128}},$$

for any adversary $\mathcal{A}$ running in time t and making at most q queries at its challenger. Here, m is the output's size of our function f and T_f is the time to do one f computation on the fixed RAM model of computation and c_1 and c_2 are two constants depending only on this model. In other words, we assume that the best attacks are either a exhaustive key search or a linear cryptanalysis.

Pseudorandomness and Unforgeability of the MILENAGE algorithms. We begin by reducing the prf-security of G_{mil1} and G_{mil2} to the prf-security of the function f. This implies the Mac-security of each MILENAGE algorithm.

Theorem 29. $[\mathsf{prf} - \mathbf{security\ for\ G_{mil1}\ and\ G_{mil2}}.]$ *Let* $G_{\mathsf{mil1}}, G_{\mathsf{mil2}} : \{0,1\}^{\kappa} \times \{0,1\}^d \times \{0,1\}^t \times \{0,1\}^t \to \{0,1\}^n$ *and* $f : \{0,1\}^{\kappa} \times \{0,1\}^{128} \times \{0,1\}^{128} \to \{0,1\}^{128}$ *be the two functions specified above. Consider a* (t, q)*-adversary* $\mathcal{A}$ *against the* prf*-security of the function* G_{mil1} *(respectively* G_{mil2}*), running in time* t *and making at most* q *queries to its challenger. Denote the advantage of this adversary as* $\mathsf{Adv}^{\mathsf{prf}}_{G_{\mathsf{mil1}}}(\mathcal{A})$ *(respectively* $\mathsf{Adv}^{\mathsf{prf}}_{G_{\mathsf{mil2}}}(\mathcal{A})$*). Then there exists a* $(t' \approx 3 \cdot t, q' = 3 \cdot q)$*-adversary* $\mathcal{A}$*' with an advantage* $\mathsf{Adv}^{\mathsf{prf}}_f(\mathcal{A}')$ *of winning against the pseudorandomness of* f *such that:*

$$\mathsf{Adv}^{\mathsf{prf}}_{G_{\mathsf{mil1}}}(\mathcal{A}) = \mathsf{Adv}^{\mathsf{prf}}_{G_{\mathsf{mil2}}}(\mathcal{A}) = \mathsf{Adv}^{\mathsf{prf}}_f(\mathcal{A}').$$

Proof. We construct the adversary $\mathcal{A}_f$ using a prf-adversary $\mathcal{A}_{G_1}$ (respectively $\mathcal{A}_{G_2}$). The latter uses $\mathcal{A}_f$ as a challenger for a prf-game $\mathbb{G}_{\mathsf{prf}(f)}$ and can only communicate with $\mathcal{A}_f$ whereas $\mathcal{A}_f$ has access to a challenger for f. To begin with, the challenger $\mathcal{C}_f^{\mathsf{prf}}$ chooses a bit b and a private $\mathsf{sk} \in \{0,1\}^\kappa$. If $b = 0$, it assigns f to a random function and if $b = 1$, it assigns f to the specific internal function.

The adversary $\mathcal{A}_f$ waits for queries from $\mathcal{A}_G$ of the form (m, a, b), with $m = m^{(1)}\|m^{(2)}\|m^{(3)}\|m^{(4)}\|m^{(5)} \in \{0,1\}^d$, and $a, b \in \{0,1\}^t$ and responds as follows:

- It queries its challenger $\mathcal{C}_f^{\mathsf{prf}}$ for inputs $(m^{(1)}, 0^{128})$ and receives the value Out_1.
- Then, it computes $\mathsf{Top}_C = m^{(1)} \oplus Out_1$ and it queries $\mathcal{C}_f^{\mathsf{prf}}$ for inputs $(\mathsf{Out}_1, m^{(2)})$ and receives the value Out_2.
- It queries $(m^{(4)}, \mathsf{Out}_2 \oplus rot(\mathsf{Out}_1 \oplus (m^{(3)}\|m^{(3)}), m^{(5)}))$ (respectively $(m^{(4)}, rot(\mathsf{Out}_1 \oplus \mathsf{Out}_2, m^{(5)}))$) and receives the value Out_3.
- It returns the value $\lfloor \mathsf{Out}_1 \oplus \mathsf{Out}_3 \rfloor_{a,b}$.

This step is repeated up to a total of q queries from $\mathcal{A}_{G_1}$ (respectively $\mathcal{A}_{G_2}$), with a and b fixed.

At some point, $\mathcal{A}_{G_1}$ (respectively $\mathcal{A}_{G_2}$) halts and outputs a guess d of the bit b. The prf-adversary $\mathcal{A}_f$ chooses its guess b' as $b' = d$ and forwards it to $\mathcal{C}_f^{\mathsf{prf}}$, which verifies if $b = b'$.

We analyze this simulation. Recall that the challenger responded either with a random value (if its internal bit b was set to 0) or with the output of the function $f(\mathsf{sk}, ., ., .)$ (if its internal bit was set as 1).

Thus, the output Out_3 matches either the output of the output of the function $G_1(\mathsf{sk}, m, a, b)$ (respectively $G_2(\mathsf{sk}, m, a, b)$) or a random function (indeed, the combination of two random functions by a boolean addition gives a random function). So the prf-adversary $\mathcal{A}_f$ simulates perfectly a prf-challenger of G_1 (respectively G_2). Thus, we have:

$$\begin{aligned}
\mathsf{Adv}_f^{\mathsf{prf}}(\mathcal{A}_f) &= \big|\Pr[\mathcal{A}_f \to 1 \mid b = 1] - \Pr[\mathcal{A}_f \to 1 \mid b = 0]\big| \\
&= \big|\Pr[b = b'|b = 1] - \Pr[b = b'|b = 0]\big| \\
&= \big|\Pr[b = d|b = 1] - \Pr[b = d|b = 0]\big| \\
&= \big|\Pr[d' = d|b = 1] - \Pr[d' = d|b = 0]\big| \\
&= \big|\Pr[\mathcal{A}_{G_1} \to 1 \mid b = 1] - \Pr[\mathcal{A}_{G_1} \to 1 \mid b = 0]\big| \\
&= \mathsf{Adv}_{G_1}^{\mathsf{prf}}(\mathcal{A}').
\end{aligned}$$

(similar computation for G_2). □

We use the generic result specified in 1 to reduce the mac-secure to the prf-secure of the function G_1 (resp. G_2).

Theorem 30. [$\mathsf{Mac}-$**security for** $\mathbf{G}_{\mathsf{mil1}}$ **and** $\mathbf{G}_{\mathsf{mil2}}$.] *Let* $G_{\mathsf{mil1}}, G_{\mathsf{mil2}} : \{0,1\}^\kappa \times \{0,1\}^d \times \{0,1\}^t \times \{0,1\}^t \to \{0,1\}^n$ *and* $f : \{0,1\}^\kappa \times \{0,1\}^{128} \times \{0,1\}^{128} \to \{0,1\}^{128}$ *be the two functions specified above. Consider a* (t,q)*-adversary* $\mathcal{A}$ *against the* mac*-security of the function* G_{mil1} *(respectively* G_{mil2}*), running in time* t *and making at most* q *queries to its challenger. Denote the advantage of this adversary as* $\mathsf{Adv}_{G_{\mathsf{mil1}}}^{\mathsf{mac}}(\mathcal{A})$ *(respectively* $\mathsf{Adv}_{G_{\mathsf{mil2}}}^{\mathsf{mac}}(\mathcal{A})$*). Then there exists a* $(t' \approx 3\cdot(t+O(n+d)), q' = 3\cdot q)$*-adversary* $\mathcal{A}'$ *with an advantage* $\mathsf{Adv}_f^{\mathsf{prf}}(\mathcal{A}')$ *of winning against the pseudorandomness of* f *such that:*

$$\mathsf{Adv}_{G_{\mathsf{mil1}}}^{\mathsf{mac}}(\mathcal{A}) = \mathsf{Adv}_{G_{\mathsf{mil2}}}^{\mathsf{mac}}(\mathcal{A}) \leq \mathsf{Adv}_f^{\mathsf{prf}}(\mathcal{A}') + \frac{1}{2^n}.$$

Indistinguishability of the MILENAGE algorithms. We proceed to reduce the ind-security of G_{mil1} and G_{mil2} to the prf-security of the function f. This implies the prf-security of each MILENAGE algorithm.

Theorem 31. $[\mathsf{ind} - \textbf{security of } \mathbf{G_{mil1}} \textbf{ and } \mathbf{G_{mil2}}.]$ *Let* $G_{\mathsf{mil1}}, G_{\mathsf{mil2}} : \{0,1\}^{\kappa} \times \{0,1\}^{d} \times \{0,1\}^{t} \times \{0,1\}^{t} \to \{0,1\}^{n}$ *and* $f : \{0,1\}^{\kappa} \times \{0,1\}^{128} \times \{0,1\}^{128} \to \{0,1\}^{128}$ *be the two functions specified above. Consider a* (t, q)*-adversary* $\mathcal{A}$ *against the* ind*-security of the function* G_{mil1} *(resp.* G_{mil2}*), running in time* t *and making at most* q *queries to its challenger. Denote the advantage of this adversary as* $\mathsf{Adv}^{\mathsf{ind}}_{G_{\mathsf{mil1}}}(\mathcal{A})$ *(resp.* $\mathsf{Adv}^{\mathsf{ind}}_{G_{\mathsf{mil2}}}(\mathcal{A})$*). Then there exists a* $(t' \approx 3 \cdot t, q' = 3 \cdot q)$*-adversary* $\mathcal{A}$*' with an advantage* $\mathsf{Adv}^{\mathsf{prf}}_{f}(\mathcal{A}')$ *of winning against the pseudorandomness of* f *such that:*

$$\mathsf{Adv}^{\mathsf{ind}}_{G_{\mathsf{mil1}}}(\mathcal{A}) = \mathsf{Adv}^{\mathsf{ind}}_{G_{\mathsf{mil2}}}(\mathcal{A}) = \mathsf{Adv}^{\mathsf{prf}}_{f}(\mathcal{A}').$$

Proof. To prove the ind-security of G_1 (resp. G_2, we reduce this security to the prf-security of G_1 (resp. G_2) and then to the prf-security of the function f using the theorem 29 . The first reduction follows the exact lines of the proof of the prf-security of the TUAK algorithms and we obtain:

$$\mathsf{Adv}^{\mathsf{ind}}_{G_1}(\mathcal{A}) = \mathsf{Adv}^{\mathsf{ind}}_{G_2}(\mathcal{A}) = \mathsf{Adv}^{\mathsf{prf}}_{f}(\mathcal{A}').$$

□

4.7.5 Comparison Between Several UMTS-AKA Variants

The Table 4.9 compares our proposal to the UMTS-AKA protocol and to the two more promising variants we have in this chapter.

	Defeating:			Security:				
	Attack n°1	Attack n°2	Attack n°3	Prop. n°1	Prop. n°2	Prop. n°3	Prop. n°4	Prop. n°5
3G AKA	x [104]	x [31]	x	x	x	x	?	?
Arapinis	✓ [31]	✓ [31]	x	✓	x	x	?	✓* [31] [7]
Van Den Broek	✓ [104]	x	✓	✓	x	x	?	?
Our variant	✓	✓	✓	✓	✓	x	✓	✓

Attack n°1: IMSI Catcher § 4.3.	Prop. n°1: Confidentiality of the previous location §3.2.
Attack n°2: Linkability of failure messages § 4.3.	Prop. n°2: ww-unlink §4.5.
Attack n°3 : Our traceability attack § 4.3.	Prop. n°3: nf-unlink §4.5.
Prop. n°4: State-confidentiality & soundness §3.2.	Prop. n°5: Key-indistinguishability & Client- and Server-impersonation §3.2.

Figure 4.9: Comparison between several UMTS-AKA variants. For attacks, a ✓ denotes the protocol resists the attack, while x denotes a vulnerability; for properties, a x denotes the property is *not* achieved, while a ✓ indicates security with respect to that property.

4.7.6 Practical Considerations

In this section, we discuss some of our design choices for the PrivAKA protocol. As opposed to the proposal of van den Broek et al. [104], we opted to continue using (TMSI, LAI) tuples for the identification phase of the protocol. This infrastructure is maintained strictly by servers, with no operator contribution; thus it is efficient and inexpensive. Moreover, TMSI values and their correspondence to the client's IMSI is easy to find. In our proposal, we bypass IMSI catchers attacks by never sending IMSIs in clear, and we add a symmetric authentication step in the encryption, thus precluding the client-unlinkability attack we found against the UMTS-AKA variant of Arapinis et al. [31]. For the encryption, we use an IND-CCA public-key encryption scheme and we require

a minimum PKI, only for operators. A client only stores the public key (and certificate) of the operator it subscribes to, thus minimizing key-management problems. In the TMSI reallocation step, we add an implicit authentication step, preventing Denial-of-Service-based linkability. We also add a freshness index, which prevents replays of challenges based on old sequence numbers.

We do specify, however, that our variant can only guarantee client-unlinkability if the *size* of the TMSI is equal to the length of the output of the PKE scheme. This is a non-trivial requirement, since servers are expected to keep track of all the TMSIs they issue; while using a shorter TMSI does not leak anything about the IMSI value, it does allow mass-surveillance organisms to track users down by distinguishing between the length of the encrypted IMSI as opposed to the TMSI length. On the positive side, servers may store TMSI values for a shorter while, since as soon as the user leaves the area, the TMSI is no longer useful.

Moreover, we recommend using a field of 32 bits for the index values $\mathsf{idx}_{\mathsf{C}}$, $\mathsf{idx}_{\mathsf{Op,C}}$. In fact, every time a session is aborted, the index(indices) is (are) increased. The only way to replay a challenge is to previously drop 2^{32} successive authentication challenges, which is in our opinion hard to do. We require that the size of all the variables (except the network variables $\mathsf{Op}_{\mathsf{Id}}$ and Id_{S}) is: 96 bits for a 64-bit security bound, 128 bits for a 96-bit security bound and 154 bits for a 128-bit security bound.

We make the assumption that clients are aware of their current LAI, and thus avoid client-tracking by means of an itinerary. This is not a very strong assumption, since mobile devices are often equipped to detect their LAI. Finally, we bypass distinguishing attacks that exploit the re-synchronization phase by ensuring that sequence numbers cannot be desynchronized (and replays of challenges using old sequence numbers are prevented). Keeping in mind the practical requirement of minimizing the communication between servers and operators, our variant ensures that operators are contacted only in case the protocol is abnormally run or an adversary is detected. We also simplify the rather complex UMTS-AKA structure, including only three communication phases rather than five.

Finally, we require to restrict the batch of authentication vectors at only one vector if the last message (sent from the server to update the operator sequence number) can be dropped.

4.7.7 Additional Practical Considerations: Two Proposed Patents

One of the goal of this thesis is to provide practical solution for a possible future standardization and implementation. Thus, we have deposed two patents, a first one focus on the user identification raking into consideration lawful interceptions and a second one focusing on the impact of subscriber key leakage.

Lawful Interception Consideration.

In our PrivAKA variant, we consider some practical requirements, e.g. the impossibility of sharing secret values between operators, the limited and restricted communications between the USIM and the mobile equipment, and inside the core network. Despite this, there exist a lawful requirement we have to take care for standardizing and implementing a protocol. This requirement is the lawful interception.

All telecommunication operators have to deploy all resources necessary for monitoring communication on public channel, notably the radio link included in the access network. A lawful interception can be put into place to collect two main kinds of information. The first one concerns the content of the communications, i.e. conversations between some persons, mail contents, exchanged files etc. The second one concerns identifying information such as the specific call number or the received or unreceived calls. This administrative monitoring can be requested by lawful interception for reasons such as the national defence, for the national interest protection, or

the fight against terrorism. The content of the lawful interception warrants depend on the national legacy. The data is associated with a specific call number or identifier, not a physical person. That is the main reason the local operator which manages the local area must know such data for all the mobile users located in this area. The international mobile subscriber identity (IMSI) is one of the pieces of information which can be asked. Thus, the local operator has to (more or less depending the local regulation) know the identifier of all clients physically located in its managed area, i.e. clients managed by the local operator and clients in the roaming situation.

An operator can obtain this value either from the client or its operator. In order to start a communication with the local network, a client identification is required. This identification includes either the permanent identifier or a temporary one as detailed in Section 2.3 and is executed on a public channel notably in the access network. This procedure is really problematic for the user identity confidentiality as explained in section 4.3. The other option to obtain this information is from the client of the operator. But in this case, the home network could be vulnerable to an anonymization of the roaming client since their related operator can just send a fake identifier. Legally, the USIM cards and mobile equipments are implemented such that the client can anonimyze itself only using a temporary value which has been previously exchanged with the *local* server.

In Patent nr 1659583 [8], we propose a new method for the user identification procedure respecting user identity confidentiality and lawful interception. In this new procedure, the IMSI is never sent in cleartext from the client; instead only an encrypted version of this value is sent. The IMSI value in cleartext is sent from the operator. The encrypted identifier is sent from the user with a specific MAC value permitting to guarantee to the local server, the similarity of the identifier encrypted by the client and the one received, without decrypting the encrypted identifier. That allows the local server to obtain the real user permanent identity without jeopardizing user identity confidentiality.

Impact of secret subscriber key leakage.

The security of the AKA protocols is based on the confidentiality on the long-term keys sk_C and $\mathsf{sk}_{\mathsf{Op}}$ as previously detailed. It would be very difficult to achieve strong security against an adversary who knows all of the secret keys and algorithms that a subscriber is using. But we can make sure that the attacks would be much harder in practice. As denoted in the technical research TR 33.899 [23], one requirement on the AKA protocols, is that an adversary, even if he knows the subscriber secret key sk_C, would have to carry out a long-term active Man-in-the-Middle attack in order to eavesdrop on that subscriber. We note that the operator keys are not easy to recover by a client in real-world implementations, as they are never stored on the USIM card. Instead, what *is* stored in the USIM card is an intermediate value, obtained after either a first Keccak truncated permutation or a call of AES algorithm; thus the operator key is easy to use, but hard to recover.

The subscriber key is considered to be confidential. But if this security assumption fails, the loss of security is catastrophic. The subscriber key might leak to an attacker for a number of reasons, e.g.:

a. Hacking at the factory (USIM framers) where sk_C is generated and embedded.

b. Hacking of the communication channel over which sk_C is transported from USIM framers or subscription manager to mobile operator.

c. Hacking the mobile operator.

[8] This patent was submitted to the *Institut national de la propriété industrielle* (INPI) on October 4th, 2016. The request number is 201109.

d. An insider attack at a mobile operator or USIM framers.

e. A local attack (e.g. side channel) on the USIM card in the supply chain.

f. A local attack (e.g. side channel) on the USIM card when it is temporarily "borrowed" from the customer.

Operators and framers must prevent these possible leaks from happening but it is really hard to do not consider them. A powerful adversary as described before which wants to eavesdrop the communications of a specific user. Moreover, the adversary knows which authentication algorithm the related client uses in this USIM card may yet obtain them. Moreover, leakage of additional secret constraints and proprietary algorithms are also occur.

In Patent nr 1659585 [9] we propose a new method reducing the impact of secret key leakage respecting the security and privacy requirements. The main idea of this method is to avoid client impersonation on the confidentiality of the current value of the sequence number. Indeed, with although the adversary can also obtain the initial value of the client state, he cannot recover the current value since the number of executed session is unpredictable.

4.7.8 Evaluation & Modular Countermeasures

In proposing our variant of UMTS-AKA, we explicitly or implicitly addressed several attacks. We discuss these below, referring the reader to Figure 4.10 for a better overview. Note that PrivAKA is an ideal protocol variant, i.e. which includes all the required modifications to guarantee all the security and privacy properties. Most of these modifications are independent, i.e. we can propose several fixed variants of the UMTS-AKA protocol providing some (not all) properties which imply less modifications. In this section, we detail the modifications of the UMTS-AKA protocol require to guarantee each property independently.

Strong Server-Impersonation: the UMTS-AKA protocol guarantees only a weak version of the server-impersonation property, in which the adversaries cannot corrupt any server. We add of a (public) server-specific value in all the internal cryptographic algorithms. Such a modification prevents attacks in which an adversary replays authentication vectors from one network to another. The added value implies an update of all the internal cryptographic algorithms; however the update is nearly for free since we respect the same construction inside both instantiations (MILENAGE and TUAK). Moreover, we propose to include the sequence number in the inputs of the algorithms (except the function $\mathcal{F}_5$ since the anonymity keys cannot depend to the sequence number). Intuitively that allows the protocol to offer a better general security, since collision of the random value R in two sessions does not imply a collision in the value Res in both sessions.

Server Corruptions: the original UMTS-AKA protocol only offers a weak degree of key-indistinguishability and impersonation security, in the absence of server corruptions. Since servers are trusted to run the authenticated key exchange step, corrupting a server compromises any security of a channel this server establishes; however, in the UMTS-AKA routine, this flaw is exacerbated, since the corruption results can be re-used later in non-vulnerable areas. This is an active, and rather complex attack, but it is highly parallelizable and has a great security impact. To mitigate this risk, we added a server-specific, unique, publicly-known identifier $\mathsf{Id_S}$, which is now given as input to all the cryptographic functions.

[9] this patent was submitted to the *Institut national de la propriété industrielle* (INPI) on October 4th, 2016. The request number is 201169.

Added countermeasures	Cost	Attacks it Prevents	Attack Impact
Client sends encrypted IMSI	- Needs IND-CCA PKE encryption - Simple PKI (only operators)	Client Confidentiality: (IMSI Catchers)	Trace many users - Parallelizable - Passive/Active
	Large TMSI size	Client unlinkability: Distinguish TMSI/IMSI msg.	Trace 1 user: - Non-parallelizable - Active only
Authenticate TMSI reallocation (see also: index)	New reallocation alg.	Client unlinkability: (Denial-of-Service)	Trace many users - Parallelizable - Active only
		Client unlinkability: Distinguish TMSI/IMSI	Trace 1 user - Non-parallelizable - Active only
Index idx_C, idx_S	New 1-bit state variable	Client unlinkability: Prompt resynch, distinguish	Trace 1 user - Non-parallelizable - Active only
		S.Imp-resistance: Challege is un-replayable	Impersonate servers - Parallelizable - Active only
Introducing Id_S	- New server identifier - Changed crypto algs.	S.Imp-resistance k.ind-security Sound-security (Server Corruptions)	Break sec. channel - Parallelizable - Needs corruptions - Active only
Use only current LAI	- Clients must know LAI - Clients store Id_S	Location privacy: (Track past LAI)	Trace 1 user per LAI - Non-Parallelizable - Passive

Figure 4.10: Assessment of our UMTS-AKA variant: cost and effect of countermeasures.

Client Confidentiality: the confidentiality of the user identity is the most problematic property that we need to assure in the communication between a mobile and its home network. In the UMTS-AKA protocol, some weaknesses (as described in the section 4.3) restrain the guarantee of a such confidentiality. Thus, a new user identification procedure is required. In the PrivAKA protocol, we never send the permanent identity IMSI in cleartext. Instead we send a public-key encryption of the permanent identity as detailed in section 4.7.1. Such a modification requires a new user identification request including a random value R_{id} and a "cheap" PKI between the operator and the client. Indeed, the latter only needs one long-term couple of asymmetric keys pre-exchanged between the operator (more precisely one for each HLR) and all of its subscribers. Thus this minimizes key-management problems and we recommend to use only one secret/public key pair for all the clients owning the same USIM card generation instead one couple for all the clients managed by the same HLR. Our new user identification procedure notably implies a new problem: since the user answers with the encryption of its permanent identifier, the server cannot recover the IMSI of this client. The Mobile Country Code MCC and Mobile Network Code MNC values are

included in the permanent identity. These values notably enable the server to find the HLR of the client. Without these values, the VLR cannot request authentication vectors since it cannot guess which HLR is the one of the client. In the PrivAKA protocol, the IMSI is sent encrypted, and the VLR has to require the HLR of the client to obtain the IMSI value. Thus, the client has to sent the MCC and MNC values with the encrypted IMSI value in the user identification answer. That implies the PrivAKA protocol cannot guarantee the confidentiality of the user's operator. But this confidentiality is not required by the 3GPP standards and does not question. We still remain if such an additional is enough to solve this deficiency.

Client Unlinkability: since the UMTS-AKA protocol is vulnerable to attacks such as IMSI catchers or client-tracking by means of error messages, the protocol cannot guarantee the user untraceability. Firstly, we fix the user identification phase. The user location and identity confidentiality (and their related modifications) are required to guarantee the untraceability. Moreover, a flag $\mathsf{flag}_{\mathsf{Id}_{\mathsf{temp}}}$ is required to obtain an unique user identification answer. Indeed, this flag manages the choice to use either an encryption of the permanent identity or the temporary identifier. We remove the permanent identifier request implying the abort of a session if the server cannot identify the user with the first answer. Then, we require that the size of temporary identifiers and the encrypted permanent identifier are identical to make indistinguishable the user identification answer. We also include the TMSI-reallocation in the challenge-response. Indeed, we require to use the new identification procedure described in Section 4.7.1. The new challenge-response described in Section 4.7.1, is also required. This new procedure needs a new management of the sequence number and a new abort procedure and implies the following modifications and costs:

- An index value which removes the replay attacks.
- A new check of the sequence number: indeed, the client accepts only one sequence number instead Δ ones.
- A new management of the sequence number update: the operator sequence number is updated after that the protocol is accepted instead of during the authentication vectors generation. That is really important since we remove the resynchronization procedure. Moreover we require an improvement of the size of the sequence number. This requirement permits to also improve the different others security properties.

State-confidentiality & Soundness: we recall that the state-confidentiality and soundness properties are guaranteed (as in the UMTS-AKA protocol) without modifications. As previously detailed, in the classic UMTS-AKA protocol the soundness property is guaranteed since the servers do not exchange unused authentication vectors. In our variant, such an assumption is not required since we add to key-indistinguishability and impersonation security by adding a server-specific value in all the cryptographic functions. We also remove the local TMSI unknown procedure. This latter modification is required to guarantee user location confidentiality.

Denial of Service: apart from being a means of breaking client-unlinkability, DoS attacks can also facilitate IMSI catchers, and add to the complexity of the AKA procedure. One way of causing a DoS in the original protocol is to send a random string as a replacement for the TMSI reallocation message. The client will parse this as a different TMSI than the intended one, and thus the server will need to request the user's IMSI in clear. We mitigate DoS attacks by using authenticated encryption for the TMSI reallocation and ensuring that no desynchronizations can occur.

Itinerary tracking: one disadvantage of UMTS-AKA is that the client's past location is revealed during the protocol, allowing to track up to one user per LAI at any one time. We bypass this difficulty by only using current LAI values.

4.7.9 A Required Desynchronization Analysis

Deny-of-Service attack, denoted DoS, is an attack aiming to make the delivery of services unavailable for an indefinite time. The goal of this attack is to prevent the protocol's execution apart from any security or privacy properties. Thus, this attack is not taken into account in our security and privacy models. Our goal in this paper is not to consider such attacks. However, we do wish to achieve DoS resistance against PrivAKA. Since the resynchronization procedure has been removed, a desynchronization will directly imply a Deny-of-Services. Two possible entities can desynchronize operator/client's states: the server or a MitM adversary between the client and server.

An easy attack consists in desynchronizing the operator through a malicious server. Indeed, a server can play the last message "Update Sequence Number" without running the challenge-response phase with the client. Thus, a server can update the operator's state out-of-synchronization with the respective client. However, we note that servers have little to gain in desynchronizing the operator's state from client is. In particular, servers have more no gain by communicating with clients and attempting to learn sensitive information. More globally, we assume that only a MitM adversary is relevant to a Deny-of-Service attack. If a DoS of this type is relevant to the application scenario, an easy fix would be to change the structure of the authentication vectors sent by the operator to the server, such that the server no longer knows the correct response value. Whenever it receives the response, the server forwards this value to the operator, instead of the "Authentication OK" message, allowing the operator to verify the validity of the response. However, this countermeasure comes at a communication cost of one added message (namely the accept/reject message from the operator to the server), and it also implies that the operator needs to remember the generated authentication vectors at least until they become invalid.

Finally, we consider a Man-in-the-Middle adversary can deny the protocol's execution from a desynchronization. We are studying with formal proof if such an assumption is reasonable or not. This study is based on the Tamarin verification tool.

Chapter 5

Proxying over the TLS Handshake: Keyless SSL

Contents

In this chapter, we will study the well-known Transport Layer Security (TLS) protocol, in particular when the latter is proxied through an intermediary entity between web-client and web-server, alternatively called middlebox and edge server. The presence of such an entity implies some issues for the all-or-nothing disclosure of end-user traffic to middleboxes and, more globally, for the global security of the handshake protocols proved through. In this chapter, we are focused on one of the main handshakes-delegation proposal when the intermediate entity does not have its own certificate to establish the secure channel with the web-client: Keyless SSL. We firstly give a global overview of the web delivery services, and the context of the use of the proxied TLS protocol. Then, we consider the different security and practical requirements for secure handshakes between endpoints (client and server), with the presence of intermediary entity (the middlebox/edge-server) and propose a first formalization of 3-(S)ACCE security. Our model is used to provide a formal security analysis of the Keyless SSL architecture, but it can also capture generic content delivery network (CDN) characteristics. The analysis on Keyless SSL points out some security and practical weaknesses, both with respect to traditional 2-party secure-channel establishment, and respect to the additional requirements of our model. Lastly, we also propose an enhanced design based on TLS 1.2 that does achieve all the security requirements that we envisaged in our 3-(S)ACCE model. We finally discuss the strengths and shortcomings of existing three-party ACCE architectures based on CDNs, from two standpoints (efficiency and security), and suggest new architecture for secure 3-(S)ACCE.

5.1 Web-delivery Services and TLS

One of the most fundamental requirements of modern-day cryptography is that of securing the communication channel between two parties, even when these exchange messages over an insecure network, such as the Internet. Arguably one of the best known such protocols is TLS/SSL, which is used to secure most Internet connections today. The Transport Layer Security (TLS) protocol is the most widely deployed secure-channel protocol on the Internet. For example, connections between modern web browsers and popular websites are secured using HTTP over TLS (HTTPS). With increased concerns over both mass surveillance and criminal activity on the Internet, the use of TLS is slowly becoming mandatory for many use cases, including the Web.[1]. TLS is designed to be used between a client and a server, which are typically authenticated using public-key certificates. In TLS, the initial authenticated key exchange phase is called the *handshake* and the subsequent authenticated encryption phase is called the *record*. The handshake protocol supports a number of modes, such as RSA key transport, Diffie-Hellman key exchange, or pre-shared keys. The record protocol also supports a variety of constructions, including stream ciphers, block ciphers, as well as modern schemes that provide AE with additional data (AEAD).

TLS and AKA protocols. Although they both aim to construct secure channels, TLS differs from the AKA protocols presented in the previous chapters. In a few crucial way, the TLS protocol relies on public-key and certificates in the handshake (contrary to the symmetric AKA protocols). In addition, the AKA protocols require mutual authentication whereas this feature is only optimal for TLS (the default mode features server-only-authentication). TLS is specified [60] as a two-party protocol only, by contrast the AKA protocols are designed to be run in the presence of a middlebox. directly specified considering the intermediary server. Finally, one very typical feature of TLS is its key-confirmation step, which once more sets it apart from AKA (and most other protocols in the literature). In TLS, the main functionality of key-confirmation is allowing the two parties to confirm that they computed the same keys, and that their views of the transcript they share are identical. Consequently, these two protocols (AKA and TLS) cannot easily be analysed in the same way. This justifies the very different approaches we take in this manuscript.

Recall that an authenticated key-exchange (AKE) protocol, is traditionally said to be secure if a MitM (man-in-the-middle) adversary is unable to distinguish the real, established session-keys from random keys [41]. These protocols offer at least the guarantee of *authenticated and confidential channel establishment* (ACCE) security as we detailed in Section 5.2.1. The AKE protocols were designed as 2-party protocols, normally run in a client-server infrastructure. However, TLS 1.2 handshakes do not meet the standard AKE security definition because the channel keys are used to encrypt the finished messages, even before the key agreement is complete. This provides a real-from-random distinguishing oracle for the MitM attacker. After years of struggling to find the right secure channel definition that is suitable for TLS, researchers developed the notion of ACCE-security [74], which requires that the keys generated by the TLS handshake can safely be used for authentication encryption in the record layer.

Two-pary TLS 1.2 Handshakes. In the TLS 1.2 protocol version, a client and a server rely on a public key infrastructure (PKI) and on some exchanged information, to establish a *master secret*, msk, using a *nonce* and a *key share* each. In particular, each party provides a nonce and a key share msk or the *TLS handshake*, as follows. Firstly, a pre-master secret pmk is computed using only the two key shares. Secondly, the msk is obtained as the result of applying a pseudorandom function (PRF), keyed with the pmk value, to the two session-nonces. Then, this pmk value is used as key to a pseudorandom function (PRF) to obtain a master secret msk, on input the two session nonces.

[1] See the latest HTTP/2 draft: http://http2.github.io/http2-spec/

The TLS protocol, executed between client and server, has two main constituents: the Handshake protocol, which is responsible for session keys agreement and authentication; and the Record protocol which provides secure channel with the established session keys. It exists three modes of TLS 1.2 handshake, namely TLS-RSA, based on RSA construction, TLS-DH and TLS-DHE based on Diffie Hellman key exchange (the first uses a static server's Diffie Hellman key and an ephemeral client's key while in the second both parties contribute ephemeral DH keys). All of these modes provide server authentication and optionally client authentication.

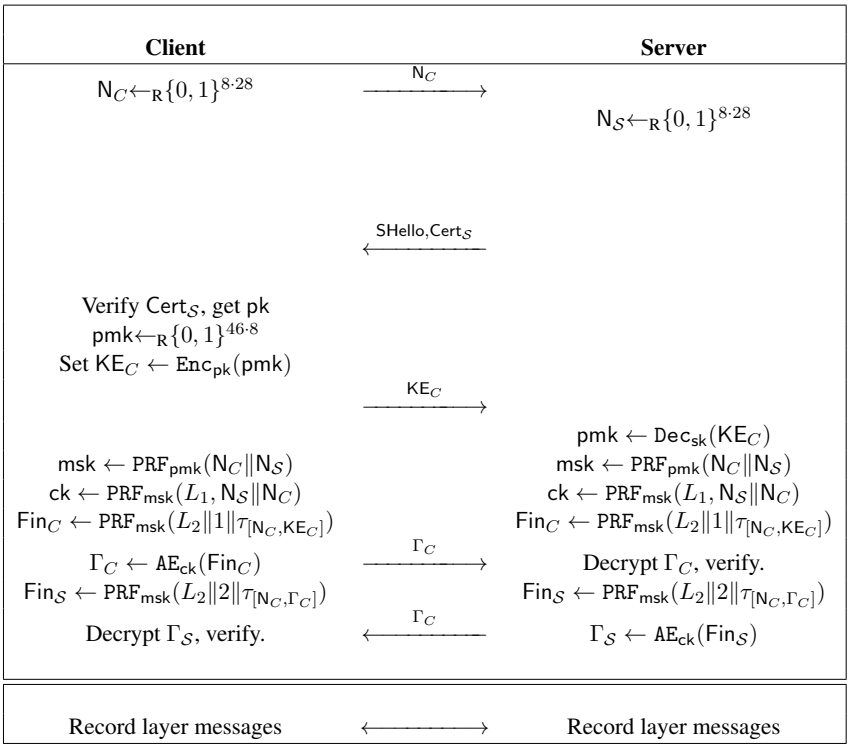


Figure 5.1: The two-party TLS-RSA 1.2 handshake.

The RSA Handshake In TLS-RSA, depicted in Figure 5.1, the client C sends a nonce N_C to the server S, which responds with its own nonce $\mathsf{N}_{\mathcal{S}}$ and an RSA public-key certificate $\mathsf{Cert}_{\mathcal{S}}$. The client then generates and sends a *pre-master secret* pmk encrypted under the server's public key. The server decrypts pmk and the client and server both compute a *master secret* msk using pmk and the two nonces. To complete the handshake, both client and server use msk to mac the full handshake transcript and send (in an encrypted form) these MACs to each other in *finished messages* ($\mathsf{Fin}_C, \mathsf{Fin}_{\mathcal{S}}$). These messages provide key confirmation, and also help detect whether a network attacker has tampered with the handshake messages. At the end of the handshake, both client and server derive connection keys ck from msk and the two nonces, and these keys are subsequently used for authenticated encryption of application data in the record phase. (In fact, these keys have already been used to encrypt the finished messages.)

The TLS-RSA protocol is the oldest and the most popular handshake mode in TLS, but it has recently fallen out of favour because it does not provide *forward secrecy*, which means that if an adversary records a TLS-RSA connection and much later it compromises the server's private key, it can decrypt the pmk, derive the connection keys, and hence read the application data. This may seem like an unrealistic threat, but with new concerns about mass surveillance by powerful adversaries, TLS-based applications increasingly require forward secrecy by default. Furthermore, the RSA encryption mode used in TLS, called RSA-PKCS#1v1.5, has been shown to be vulnerable

to a series of increasingly effective padding oracle attacks, first described by Bleichenbacher, that have proved hard to fix. For both these reasons, the TLS working group is getting rid of RSA in TLS 1.3.

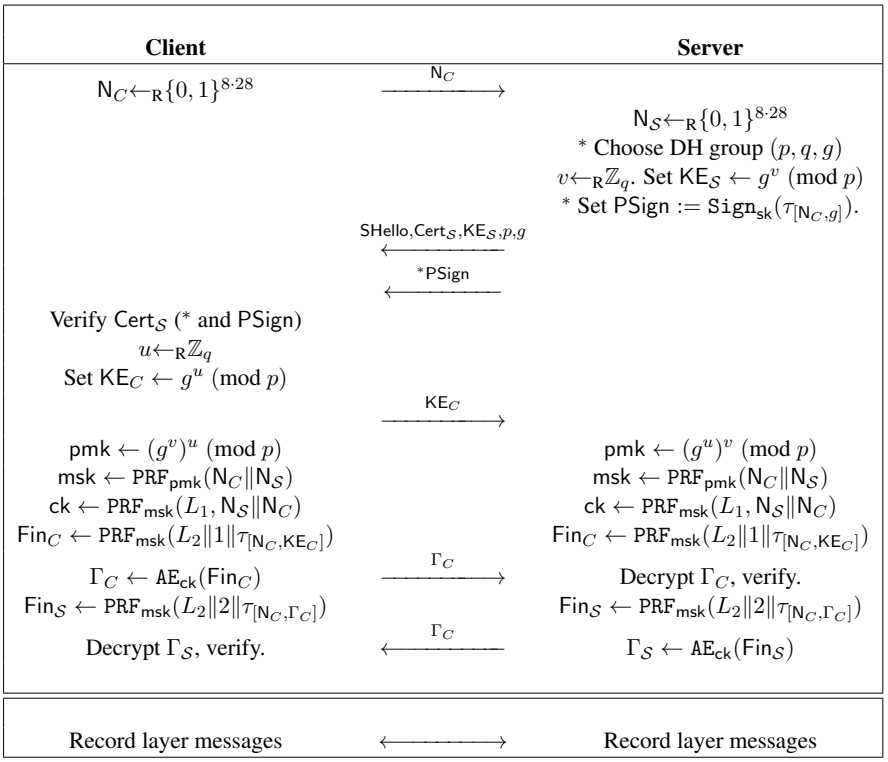


Figure 5.2: The two-party TLS-DH 1.2 handshake.

The DH and DHE Handshakes In the DH modes, the client and server first exchange nonces and the server certificate just like in TLS-RSA. The server additionally selects some DH parameters p, q, g [2], calculates one element of a DH tuple, KE_S (only for DHE mode) and it sends p, g, KE_S and along with the SHello message (i.e., KE_S is $g^v \mod p$ where v is the local, (ephemeral) exponent of the server). Additionally, if the mode is TLS-DHE, then the server also sends to the client its signature PSign on the DH parameters (i.e., see the figure 5.2, where PSign is transmitted). At the next step, the client verifies the certificate. At this stage, the client calculates the key-share, KE_C ; The rest of the protocol and computations ($\mathsf{msk}, \mathsf{ck}, \mathsf{Fin}_C, \mathsf{Fin}_S$) proceed as in TLS-RSA. The DH mode is depicted in Figure 5.2,

The key advantage of TLS-DHE is that it does provide forward secrecy as long as both parties generate fresh keypairs for each handshake and use a strong Diffie-Hellman group. Furthermore, the elliptic curve variant of TLS-DHE is considered to be as fast (if not faster) than TLS-RSA. (TLS distinguishes between Diffie-Hellman modes based on explicit primes from those based on elliptic curves, but for the results in this paper, the difference between the two is immaterial, and we refer to both as TLS-DHE.)

Session Resumption: After an established session, it is possible to store the corresponding master secret *msk*, after the connection is closed, and then resume that session. The client signals it wants to resume a past session in a particularity designed CHello message. The server can thus accept the session resumption offer. If the server accepts, it sends an accepting SHello message. In this case, the master key will be reuse of derive new session keys, on input two newly exchanged

[2] In the DH mode, the DH parameters are the same for all the sessions.

nonces. This mechanism obviously requires the server and client to remember a previous master key. Alternative session-resumption mechanisms based on session tickets have also been proposed. These mechanisms offer to the server to avoid the store of the msk.

ChangeCipherSpec Message: The *ChangeCipherSpec* message signals the activation of encryption, and only after this message the encryption in the record layer start. There is only one message whose goal is to trigger the use of the negotiated parameters.

Proving 2-ACCE for TLS 1.2. A series of papers have developed a precise cryptographic specification for TLS, called authenticated and confidential channel establishment (ACCE) [74], and proved that various combinations of handshake and record layer modes achieve this specification. A variant of ACCE, called SACCE, applies to the most common implementation of TLS where only the server is authenticated, but the client remains anonymous [81]. The first ACCE-definition for TLS appeared alongside a 2-ACCE-security proof for TLS-DHE in [74]. A subsequent paper introduced server-only authentication (SACCE) and proved the security of both TLS-RSA and TLS-DHE [81], but strictly in a two-party setting. In fact, [47] describes an attack which involves using the TLS handshake between three parties, one of which is malicious. In this thesis, we also use the recent result of Brzuska et al. [55], who proved that for mutually-authenticated *TLS-like* protocols (including TLS), deriving an export key from the master secret via a pseudorandom function yields a session key that is indistinguishable from random.

In Section A.2, we recall the standard ACCE (and (S)ACCE) security model (which we will call 2-(S)ACCE). For instance, the pivotal TLS1.2 –which we will investigate herein– in its different modes can achieve ACCE and/or SACCE security, under specific assumptions [81]. On the other hand, a number of modes and constructions used in TLS have also been shown to be insecure, resulting in high-profile attacks on both the handshake and record layers [27, 43].

Proxied Handshakes a 3-party TLS. Classical network architectures typically feature multiple users, which can connect to each other in communication sessions. For instance, by using the http protocol on top of TCP/IP protocols, a client can access a server and the data that the latter might host.

If traffic is routed over http, then adding such intermediate entities across a network presents no great difficulty, since all messages are sent in clear. However, the paradigm underlying present-day browsing and messaging is that of end-to-end security in other words, the ability to ensure that only authorized parties (in this case, the end nodes) are able to read and write information once they have established a communication session.

In theory, the TLS protocol is executed between two entities, the client and the server, to provide the secure channel. In practice, this protocol requires significant infrastructures and many intermediary entities for exchanging messages, notably the large volumes. Web sites requiring to exchange services over secure channel need to be fast and reactive, and must also secure channel establishment. Thus, one direct consequence is the latency. TLS connections may potentially be used to transfer large amounts of data (e.g., movies in streaming applications). If the content owner (the origin server) and the receiver (the client) are situated geographically far away, data transfer will be slow, involving extensive routing.

To reduce the latency of the execution of the TLS protocol and to speed up such connections, servers can be geographically distributed and located (physically) located close to the clients.

We will call such distributed servers as *middleboxes*. They are delivery services, running inside a local network and which are located between the endpoints (client and server) of the communications. These middleboxes are proxies, managing services which have been delegated by the server to the clients. They are considered as partially trusted and thus undesirable in the security and privacy sense. Indeed, in a perfect architecture all functionality will be managed in the endpoints. Naylor et al. have provided in [91] an accurate description of middleboxes and why they are beneficial to keep them in the Internet architecture.

The global approach to reduce the latency notably when large volumes are exchanged, is the use of the content delivery networks (CDNs).

They were introduced as large networks of reverse-caching proxies to accelerate HTTP delivery. Nowadays CDNs also operate onto the TLS layer within (HTTPS) communications. To achieve this, a variable number of the ACCE-related tasks of the end-server in the traditional ACCE architecture have to be delegated to the CDN's proxy-servers. To speed up such connections, content owners can hire Content Delivery Networks (CDNs) that cache popular content at edge servers located around the world and deliver them to clients based on geographic proximity.

Deploying CDNs for public HTTP traffic is relatively straight-forward. The origin server (e.g., example.com) decides what content will be cached by the CDN and puts it on some subdomain (e.g., cdn.example.com). The origin server and the CDN use the DNS protocol to direct requests for this subdomain to the nearest edge server. The client sees a significant performance improvement, but is otherwise unaware that it is not directly connected to the origin server. For HTTPS connections, however, this is more problematic. The client expects a direct secure channel to the origin server, and redirection to an unknown edge server will be seen as an attack. Consequently, the origin server and the CDNs need to agree on a way by which edge.

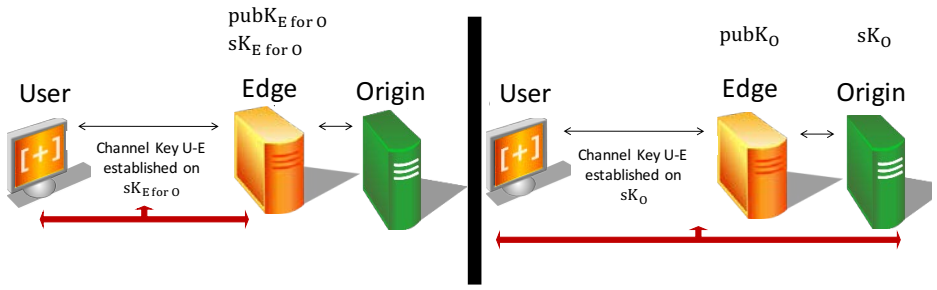


Figure 5.3: Standard TLS in CDNs (left-hand side); "Keyless SSL" in CDNs (right-hand side)

The original CDN architecture. In classic CDNs, the origin server provides a certificate and private key (e.g., for `cdn.example.com`) to the edge server, effectively allowing the edge server to impersonate some sub-domain of the origin server. This is depicted on the left-hand side of Figure 5.3. This architecture works well for simple origin servers who wish to delegate all TLS-related operations to the CDN. The main risk is that any attack on the edge server will expose the private key to an adversary who can then impersonate the origin server, at least until the corresponding certificate expires or is revoked.

Primarily, a Content Delivery Network is a sub-part of the internet infrastructure which facilitates the delivery of web-content on behalf of origin server. For instance, Akamai [68] is a well-known commercial CDN, which delivers content originating from servers owned by, e.g., AirBnB. The founding principle of a CDN is that a large set of interconnected servers, which act as reverse caching proxies between the origin-servers and the requesting end-users, can cache data from origin servers to so-called *edge-servers* $\mathcal{E}$. Then, as cached content is requested from clients geographically close to $\mathcal{E}$, the latter can deliver it faster than the distant origin-servers, and $\mathcal{E}$ can even optimise it, filter it, compress it, etc. However, all of these become sensitive when the delivering origin-server and the requesting end-users are communicating not over HTTP, but over HTTPS, as customary nowadays.

As depicted in left-hand side of Figure 5.3, in classical CDNs, the edge servers impersonate the origin servers to end-users (clients) by using public keys that were certified for the *origin-server*, which are associated with secret keys known to the *edge servers*. In this setting, the edge-servers execute TLS handshakes with the end-user by using "origin-impersonating" certificates

provisioned by the CDN itself. This setup allows the edge-server to retain, decrypt, and inspect the client's and origin server's data and inspect it for purposes such as malicious web-application attacks. When the traffic needs to be forwarded to the origin server (i.e., in the case of request for dynamic HTTP content), the edge-server establishes a separate TLS/SSL connection with the origin-server, this time with the origin-server using its own certificate when acting as a responder during the handshake. Thereafter, content relayed from the end-user (e.g., HTTP-request body) is re-encrypted by the edge-server on its way to the origin-server.

To mitigate risks as especially for high-value origin servers who want to use the CDN only for performance, but do not want to expose their long-term private keys, CloudFlare implemented a version of proxied TLS called *Keyless SSL* [101], relying on a patent by Akamai [68].

The Keyless SSL approach. In 3-party TLS design for CDNs as above, we can ponder on the fact that the edge-servers hold a secret key on behalf of the origin-server. One could alleviate this by letting the origin-servers hold their secret keys; then, the TLS handshake between the client and the edge-servers would need extending such that information would be relayed by the CDN to the origin-server for the secret-key-related computations. In other words, it would be the origin-server's secret key that is used to establish the channel-key for a TLS-session between an end-user and an edge-server. A CDN called Cloudflare implemented a version of this idea in a product called "keyless SSL" [101], whose setup is based on a patent by Akamai [68]. The right-hand side of Figure 5.3 depicts this setting. In particular, at the cost of a reduced speed-up for the content-delivery, this infrastructure presents fewer risks of having a potentially-large number of origin-impersonating secret keys leak to an adversary. However, the question of how to delegate securely part of the TLS handshake from the CDN's edge-server to the origin-servers remains.

Multi-Context TLS (mcTLS): A more principled approach of extending the 2-party ACCE infrastructure to a 3-party setting is that of multi-context TLS (mcTLS) [91] where a middle-entity (called *middlebox*) can mitigate and/or relay TLS handshakes and record communication between end-users and servers, for a variety of purposes like content filtering, compression, etc. .This approach is currently studied in a similar 3-(S)ACCE security model.

5.2 3-(S)ACCE Security Model

In the following, we will firstly recall how the classic ACCE (authenticated and confidential channel establishment) security has been defined in [73]. We define this model including the SACCE model which a variant of the ACCE model when only the server is authenticated [81].

The classic ACCE (authenticated and confidential channel establishment) security has been defined in [73] and detailed in Section A.2. We firstly introduced a new notion called *Mixed Entity Authentication* has been introduced to consider the ability to have a mutual authenticated session and unilateral authenticated session. Then, we define the 3-(S)ACCE security model, a non-trivial adaptation of traditional 2-party ACCE security (which we will note 2-ACCE to be clearly), as detailed in Section A.2, when the handshake is run in the presence of middleboxes (MB) such as CDN edge servers.

Our 3-(S)ACCE model captures the characteristics of several types of proxied handshakes including classic CDNs and Keyless SSL.

Though the terminology we use is that of Jager et al. from [74], we often use the notation introduced by Brzuska et al. [55] to which we add elements and particularities to help with our transition from a 2-party notion towards a 3-party notion for ACCE.

5.2.1 Related work

The traditional notion of AKE security, entailing that the output session keys should be indistinguishable from random output of the same length (to a Man-in-the-Middle attacker), was first proposed by Bellare and Rogaway [40]. Further extensions and concrete instantiations of these definitions followed in [39,42]. We present the AKE security notion in detail in the Appendix A.2.

However protocols like TLS/SSL, which include a *key-confirmation* step, cannot attain AKE security, since an adversary can distinguish true keys from random keys by testing them during key-confirmation. In other words, only shortened or modified versions of the TLS protocol (e.g., ones in which the key-confirmation is established without encryption [90]) can be proved secure with respect to the AKE notion. As a relaxation of traditional AKE security, Jager et al. [74] introduced the notion of ACCE security, which guarantees the security of only one application for those keys, namely, that of authenticated encryption. In particular ACCE security implies both mutual authentication guarantees and the security of the channel the parties communicate across. If an AKE protocol is used with only one-sided authentication, as is the case for most applications of e-Commerce and secure Web browsing, then the entity-authentication and secure-channel guarantees in ACCE are both one-sided only; this notion is called Server ACCE (SACCE), since it is mostly relevant in the client-server setting. This model was later refined to take into account the notion of length-hiding encryption [73], leading to the most complete analysis of the security of TLS 1.2 to date, which is due to Krawczyk, Paterson, and Wee [81]. These authors showed that, under a number of hardness assumptions and assumptions on the underlying primitives, TLS 1.2 guarantees the security notion of SACCE (ACCE restricted to the case of server authentication only), in all three operating modes: TLS-DH, TLS-DHE, and TLS-RSA. A less complete, but modular and composable proof of the security of TLS 1.2 was given later, in the constructive cryptography framework [78]. Finally, the work of Bhargavan et al., e.g. [49], used notions of cryptographic agility and secure implementations in order to understand the security of TLS with respect to renegotiations and multiple versions.

Although, by its nature, the guarantee of AKE security for the derived keys allows for some form of composability of this primitive to other symmetric-key cryptographic protocols, a first *universally-composable* definition of this model was presented by Canetti and Krawczyk [57]. Finally, an alternative, composable definition of secure channels is given by Maurer et al. [85]. These two latter generic definitions capture, as the AKE security, the idea that the handshake produces channel keys which are indistinguishable from random with respect to a MitM attacker.

5.2.2 2-(S)ACCE New Requirement: Mixed Entity Authentication

In this section, we present an additional assumption for an intermediary security property, called *Mixed Entity Authentication*. Ideally, since we construct the proxied handshake from a single, unilaterally authenticated TLS negotiation (between the client and the middlebox), and a mutually-authenticated one (between the middlebox and the server), we would want to reduce the security of our schemes only to the basic 2-SACCE and 2-ACCE games, respectively. The two differ only in their Entity Authentication property, with 2-SACCE restricting the adversary's *winning condition* to client instances π_i^m only. However, for technical reasons, we need to rely on a slightly different security notion. Namely, we consider ACCE protocols in which clients are also issued certificates. We associate party instances with a bit denoted $\mathsf{b}_{\mathsf{auth}}$ and called *authentication flag*, which is set to 0 by default. The flag value is initialized at the instantiation of π_i^m depending if the client authentication is required ($\pi_i^m.\mathsf{auth} = 1$) or not ($\pi_i^m.\mathsf{auth} = 0$). In the mEA game, we consider a NewSession oracle described as follows:

1*. NewSession($\mathsf{P}_i, \rho, pid, b_{auth}$): This query creates a new session π_i^m executed by party P_i with the role ρ, having the partner identifier pid and initializes the authentication flag with

the value $\pi_i^m.\mathsf{auth} = b_{auth}$. Moreover, the session is set to be in the running state and the associated party is its initiator, i.e., $\pi_i^m.\alpha = \perp$ and $\pi_i^m.\rho = init$.

The 2-(S)ACCE matching-conversation definitions can be adapted trivially to include that partnering instances must have matching mode-flags' values.

In the mEA game, like in the EA games, the adversary queries the first four oracles above and its goal is to make one instance, π_i^m of an uncorrupted P_i *accept maliciously*. That is, π_i^m must end in an accepting state, with partner ID P_j also uncorrupted, such that no other unique instance of P_j partnering π_i^m exists, and its authentication flag has been instantiated to 1.

We note that the importance of the condition $\pi_i^m.\mathsf{auth} = \pi_j^n.\mathsf{auth}$ for the correctness between π_i^m and π_j^n. Indeed, since the bit b_{auth} is not directly initialized for all the parties considering the studied protocol, it is important that the two parties are agreed on the authentication-feature (unilateral or mutual) of the session.

Although it is not known whether TLS 1.2 is mEA-secure as adapted above, this assumption seems quite reasonable, given the separate results given for unilaterally, and respectively mutually authenticated handshakes.

5.2.3 A Security Model: 3-(S)ACCE

The attacks presented in Section 5.1 show that proxied TLS is difficult to get right, even if the individual channels are 2-ACCE secure. Furthermore, we saw that generalizing 2-party security to 3 parties requires care since the intermediary entity has to be considered. We now present 3-(S)ACCE, our security definition for proxied handshakes.

An overview. Our framework captures several types of architectures in a generic interaction model. We consider a PKI in which middlebox and servers have registered credentials, but clients do not; in particular, any secure channel the client establishes is only *unilaterally* authenticated. Middlebox receive credentials for content-delivery upon *registering* with the content owner. This models both architectures in which the middlebox impersonates the server for content-delivery (e.g., CDNs) and those in which it uses its own credentials. Our notions of *partnering* and *freshness* capture both the designs in which the middlebox can independently compute the session keys (as in CDNs) and those in which the middlebox needs the server's help for some computations (like in Keyless SSL).

Apart from extending classical 2-ACCE entity authentication and channel security, we require two additional properties: *accountability* and *content soundness*.

Accountability. In CDN-ing over TLS, the middlebox impersonates servers to clients that are oblivious of this fact. To protect against malicious or compromised middleboxes, the server must have the power to detect and de-authorize such middleboxes. Accountability requires that the server should be able to compute the channel keys used by the client and the middlebox. With this key, the server can audit the behaviour of the middlebox and take action against it, thus becoming *accountable* for any damage to the client.

Content soundness. Servers may authorize CDN middleboxes to deliver some of their contents. Content soundness requires that no malicious middlebox may serve content that it is not authorized to deliver. Typically, servers can achieve this by distributing content within separate subdomains with their own certificates, and authorizing middleboxes to serve content for specific subdomains. This demands a more extensive certificate infrastructure at the server. We explicitly note that we do not take into consideration websites with "public" content, i.e., in our model, a middlebox can only obtain content from an issuing server. In that sense, we also rule out (certain kinds of) phishing attacks.

Attributes and Partnering.

We extend the 2-ACCE setting to three parties, by adding a set $\mathbb{MB}$ of *middlebox* entities to the existing sets of clients and servers' entities. We instantiate parties as before, but extend the model to include new attributes as follows.

Names and certificates. In 2-ACCE infrastructures, if one party is identified by the other by means of, e.g, a certificate, then that certificate will point to the right partner (unless an impersonation has occurred). By contrast, especially in CDNs and Keyless SSL, the infrastructure *allows* entity impersonation to some extent. To capture the fact that one party may effectively compute keys based on another party's certificate, we no longer associate pid values with parties, but with *names* (for instance the common name –CN– entry in X.509 certificates). We assume that each certificate has a unique name associated to it. Finally, since we consider only unilaterally-authenticated handshakes for the clients, the latter have no certificates, and they are identified (for partnering purposes) by a generic name denoted "Clients". We thus define, for each 3-(S)ACCE party P_i, two new attributes: a *name* $\mathsf{P}_i.\mathsf{name}$ and a set of certificates the party may use, denoted as $\mathsf{P}_i.\mathsf{CertSet}$.

Pre-channel keys. In some proxying architectures, the middlebox is unable to compute channel keys for middlebox-client sessions independently, without a server's assistance. In Keyless SSL, the server must compute any values requiring the secret key associated with the server's certified public key. We model this as follows. We assume that server instances and middlebox instances partnered with server instances may compute a local value, which allows the middlebox to ultimately compute the channel keys ck for a session it runs in parallel with the client. We call this value a *pre-channel key* and denote it as a new attribute denoted $\pi_j^n.\mathsf{pck}$. For example, for Keyless SSL running in RSA mode, $\pi.\mathsf{pck} = \mathsf{pmk}$, whereas for the DHE mode, $\pi.\mathsf{pck} = (p, g, \mathsf{KE}_{\mathcal{S}}, \mathsf{Cert}_{\mathcal{S}}, \mathsf{PSign})$ (see Figure 5.7).

More formally, let PCK be the set of all pre-channel keys, Π the set of all possible transcripts τ, and CK the set of all channel keys. We are interested in protocols for which a function f such as $f: \ PCK \times \Pi \rightarrow CK$, taking as input a pre-channel key pck and a (handshake) transcript τ and outputting a channel key ck, can be defined. We require that $\forall pck, pck' \in PCK, \forall \pi, \pi' \in \Pi$ such that $\pi \neq \pi'$, we have that if $f(pck, \pi)$=$f(pck', \pi')$, then π=π' (i.e., that the projection of f on Π be injective).

The notion of pre-channel keys is crucial in identifying *all* partnering instances associated with a given instance. Whenever the middlebox can compute all session keys independently of the server, we have the classical two-instance partnering from 2-ACCE. However, when the server needs to aid the middlebox, and the pre-channel key is required, four instances are partnered instead of two.

Owned and proxied content. As indicated earlier, another important aspect of proxying TLS connections is related to *content*. We associate to each server $\mathsf{P}_j \in \mathbb{S}$ the content it owns, denoted Ω^j (which the adversary could choose). We assume that a content is composed of unitary elements $\omega_i \in \Omega^j$ (e.g, subdomains or single web-pages). We assume that each handshake is meant to deliver a single content unit. Recall also that we do not consider public contents, i.e. the attacker may only retrieve a content by demanding it from a server which owns it.

Middlebox can receive (and then deliver) server-content in two steps. First the middlebox *registers* with a server for a given content, receiving credentials that allow it to authenticate to the client as a legitimate distributor of that content. As a second step, the middlebox must execute a handshake with the correct server, then – upon successful mutual authentication and verification of registration for the content – the latter is delivered. We restrict the delivery to one content per handshake, sent as a response to a "$\mathsf{Contentrequest}, \omega$" message from the middlebox.

We assume that clients can verify whether a content they receive in the record layer matches

the certificate they received during the handshake[3].

Party Attributes. We formalize the requirements presented above in terms of additional attributes, for each *party* P_i:

- The **party's name** $\mathsf{P}_i.\mathsf{name}$, linked to (possibly multiple) certificates. This value is unique per middlebox or server, and is set to the label "Clients" for all clients.
- An **indexed set** $\mathsf{P}_i.\mathsf{CertSet}$ **of certificates** held locally by P_i, for which P_i stores the associated public keys that are certified, and possibly also the corresponding secret keys. This is only true for middlebox and servers; for clients, the certificate set is empty.
- An **indexed set** $\mathsf{P}_i.\mathsf{PKSet}$ **of public keys**, containing certified public keys held locally by P_i (either a middlebox or a server), such that the k-th public key of that party, $\mathsf{P}_i.\mathsf{Cert}_k$, is certified in the k-th certificate. This set is empty for all clients.
- An **indexed set** $\mathsf{P}_i.\mathsf{SKSet}$ **of secret keys** held locally by P_i, defined analogously as $\mathsf{P}_i.\mathsf{PKSet}$. For any secret keys that P_i does not have, P_i stores $\perp$ instead.
- A **set** $\mathsf{P}_i.\mathsf{Contents}$ **of contents** to which P_i has access. For a server P_j, we denote its content by Ω^j. For middleboxs P_k, we denote its entire registered content with server P_j by Ω^k_j. For clients, this set is empty.
- A **hashtable** $\mathsf{P}_j.\mathsf{Contracts}$ held locally by servers P_j, for which the entries are of the form $(\mathsf{P}_k, \Omega^k_j)$ with $\mathsf{P}_k \in \mathbb{MB}$ and Ω^k_j is the subcontent P_k registered with P_j. This table is used by the server so that it forwards *only* registered contents to appropriate middlebox.
- A **list of instances** $\mathsf{P}_i.\mathsf{Instances}$, to keep track of all existing instances of this party.

Party-Instance Attributes. We proceed to extend the attribute-set of each *party-instance* π^m_i as follows.

- The **instance parameters** $\pi^m_i.\mathsf{par}$ consisting of: a certificate, the corresponding public key, and the matching secret key from the sets $\mathsf{P}_i.\mathsf{CertSet}, \mathsf{P}_i.\mathsf{PKSet}, \mathsf{P}_i.\mathsf{SKSet}$. Clients have no such values; servers always have non-$\perp$ values; and middlebox may have the secret key set to $\perp$.
- The **partner identifier** $\pi^m_i.\mathsf{pid}$, which will be set to a party *name* (rather than a party).
- A **pre-channel key** $\pi^m_i.\mathsf{pck}$ such that $\pi^m_i.\mathsf{pck} \in PCK$, set to $\perp$ for clients and for middlebox instances whose partner identifier is set to "Client". Only *accepting* instances of servers or middlebox with partner identifier a server have a pre-channel key not equal to $\perp$.
- A **record-layer transcript** $\pi^m_i.\mathsf{rec}\tau$, which is non-$\perp$ only for accepting instances that have already computed a channel key $\pi^m_i.\mathsf{ck}$. The record-layer transcript stores plaintext messages that a party encrypts and sends to its partner, or that the party decrypted from its partner.

[3]This is modelled by means of an additional function with a Boolean output.

Partnering and Freshness.

For 2-ACCE, partners are parties sharing the same session identifier sid. For 3-(S)ACCE this is not sufficient. For instance, the design of Keyless SSL induces the dependency between a client-middlebox session and an associated middlebox-server session through which the server aids the middlebox to compute its channel keys with the client. These two sessions (and four instances) are entwined since they contain information about each other.

Indeed, we characterise 3-(S)ACCE partnering for a given instance π_i^m of a party P_i in terms of two sets. The first of these, denoted $\pi_i^m.\mathsf{PSet}$, stores the *parties* that are partnered with π_i^m (for instance, in Keyless SSL, a client, a middlebox, and a server). The second set, denoted $\pi_i^m.\mathsf{InstSet}$, stores the *instances* with which π_i^m partners. These sets include the party P_i or respectively the instance π_i^m itself, for easiness with security definitions.

In 3-(S)ACCE the *partner-instances* $\pi_i^m.\mathsf{InstSet}$ and *partner-parties* $\pi_i^m.\mathsf{PSet}$ *of an instance* π_i^m are defined constructively by Algorithm as detailed in Figure 6.4.

Partnering in 3-party ACCE.

Require: The input is a party instance π_i^m (ending in accepting state).
Ensure: The output are two sets $\pi_i^m.\mathsf{PSet}$ and $\pi_i^m.\mathsf{InstSet}$.

1. State: **Set** $\pi_i^m.\mathsf{PSet} := \{\mathsf{P}_i\}$ and $\pi_i^m.\mathsf{InstSet} := \{\pi_i^m\}$.
2. If: $(\pi_i^m.\mathsf{pid} = \mathsf{P}_j.\mathsf{name} \mid \text{for some } \mathsf{P}_j \in \{\mathbb{MB}, \mathbb{S}\})$
3. If: $(\exists$ unique π_j^n s. that $\pi_i^m.\mathsf{sid} = \pi_j^n.\mathsf{sid}$ and $\pi_j^n.\mathsf{pid} = \mathsf{P}_i.\mathsf{name})$
4. State: **Do** $\pi_i^m.\mathsf{PSet} \leftarrow \pi_i^m.\mathsf{PSet} \cup \{\mathsf{P}_j\}$ and $\pi_i^m.\mathsf{InstSet} \leftarrow \pi_i^m.\mathsf{InstSet} \cup \{\pi_j^n\}$
5. If: $(\mathsf{P}_i \in \mathbb{MB}$ and $\pi_i^m.\mathsf{pck} \neq \perp)$
6. State: **Find** $\mathsf{P}_k \in \mathcal{C}$ s.that $\exists \pi_i^p, \pi_k^\ell$ with $\pi_k^\ell.\mathsf{sid} = \pi_i^p.\mathsf{sid}$ and $\pi_k^\ell.\mathsf{ck} = \pi_i^p.\mathsf{ck} = f(\pi_i^m.\mathsf{pck}, \pi_k^\ell.\tau)$
7. State: **Do** $\pi_i^m.\mathsf{PSet} \leftarrow \pi_i^m.\mathsf{PSet} \cup \{\mathsf{P}_k\}$ and $\pi_i^m.\mathsf{InstSet} \leftarrow \pi_i^m.\mathsf{InstSet} \cup \{\pi_i^p, \pi_k^\ell\}$
8. Endif
9. If: $\mathsf{P}_j \in \mathbb{MB}$ and $\pi_j^n.\mathsf{pck} \neq \perp$:
10. State: **Find** $\mathsf{P}_k \in \mathcal{C}$ s.t. $\exists \pi_j^p, \pi_k^\ell$ with $\pi_k^\ell.\mathsf{sid} = \pi_j^p.\mathsf{sid}$ and $\pi_k^\ell.\mathsf{ck} = \pi_j^p.\mathsf{ck} = f(\pi_j^n.\mathsf{pck}, \pi_k^\ell.\tau)$
11. State: **Do** $\pi_i^m.\mathsf{PSet} \leftarrow \pi_i^m.\mathsf{PSet} \cup \{\mathsf{P}_k\}$ and $\pi_i^m.\mathsf{InstSet} \leftarrow \pi_i^m.\mathsf{InstSet} \cup \{\pi_j^p, \pi_k^\ell\}$
12. Endif
13. State: **Return** $\pi_i^m.\mathsf{PSet}$ and $\pi_i^m.\mathsf{InstSet}$.
14. Else
15. State: **Find** unique P_k, π_k^ℓ s.that $\pi_k^\ell.\mathsf{sid} = \pi_i^m.\mathsf{sid}$
16. State: **Do** $\pi_i^m.\mathsf{PSet} \leftarrow \pi_i^m.\mathsf{PSet} \cup \{\mathsf{P}_j, \mathsf{P}_k\}$ and $\pi_i^m.\mathsf{InstSet} \leftarrow \pi_i^m.\mathsf{InstSet} \cup \{\pi_k^\ell\}$
17. If: $\exists \pi_k^p, \pi_j^n$ s.t. $\pi_k^p.\mathsf{sid} = \pi_j^n.\mathsf{sid}$ and $\pi_i^m.\mathsf{ck} = \pi_k^\ell.\mathsf{ck} = f(\pi_j^n.\mathsf{pck}, \pi_i^m.\tau)$
18. State: **Do** $\pi_i^m.\mathsf{InstSet} \leftarrow \pi_i^m.\mathsf{InstSet} \cup \{\pi_k^p, \pi_j^n\}$
19. Endif
20. Endif
21. Else
22. State: **Find** unique P_j, π_j^n s.that $\pi_i^m.\mathsf{sid} = \pi_j^n.\mathsf{sid}$
23. State: **Do** $\pi_i^m.\mathsf{PSet} \leftarrow \pi_i^m.\mathsf{PSet} \cup \{\mathsf{P}_j\}$ and $\pi_i^m.\mathsf{InstSet} \leftarrow \pi_i^m.\mathsf{InstSet} \cup \{\pi_j^n\}$
24. If: $\mathsf{P}_i \in \mathbb{S}$
25. State: **Return** $\pi_i^m.\mathsf{PSet}, \pi_i^m.\mathsf{InstSet}$
26. Else
27. If: $\pi_j^n.\mathsf{pid} = \mathsf{P}_k.\mathsf{name}$ for $\mathsf{P}_k \in \mathbb{S}$
28. State: **Do** $\pi_i^m.\mathsf{PSet} \leftarrow \pi_i^m.\mathsf{PSet} \cup \{\mathsf{P}_k\}$
29. **If:** $\exists\, \pi_k^\ell$ of party P_k s.t. $\pi_j^n.\mathsf{ck} = f(\pi_k^\ell.\mathsf{pck}, \pi_j^n.\tau)$
30. State: **Find** unique π_i^p such that $\pi_i^p.\mathsf{sid} = \pi_k^\ell.\mathsf{sid}$
31. State: **Do** $\pi_i^m.\mathsf{InstSet} \leftarrow \pi_i^m.\mathsf{InstSet} \cup \{\pi_i^p, \pi_k^\ell\}$
32. State: **Return** $\pi_i^m.\mathsf{PSet}, \pi_i^m.\mathsf{InstSet}$
33. Endif
34. Endif
35. Endif
36. Endif
37. State: **Return** $\pi_i^m.\mathsf{PSet}, \pi_i^m.\mathsf{InstSet}$

Figure 5.4: Algorithm of Partnering in 3-party ACCE.

This algorithm covers all the possible communication-links in proxied handshakes: client–server, client–middlebox, client–middlebox–server, middlebox–server. If server-impersonation is used, like in CDNs, we denote this by $MB(S)$. If the middlebox uses its own credentials, we use $MB(MB)$. We assume that in all middlebox-server sessions, the middlebox always authenticates as itself, and do not specify this explicitly. To give the reader an intuition, we summarize all the possibilities in Figure 5.5. The rows are designed such that, for each entry, the partnering is defined from the viewpoint of one fixed party-instance involved in the link (this party is underlined).

For instance let us assume we wish to determine the partner-instances and the partnering parties of one instance π_i^m of the client-type. This instance can find itself in the following communication scenarios: (1) speaking to a server in a direct "C–S" link; (2) speaking to a middlebox that authenticates as itself, i.e., a "C – MB(MB)" link; (3) speaking to a middlebox impersonating a server, but running the handshake independently: "C – MB(S)" ; (4). speaking to a middlebox impersonating a server, needing the latter's assistance, as in Keyless SSL:"C – MB – S". These are the first four rows in the table in Figure 5.5.

	Actual Link	Partnering by Alg.1	Partner-sets forming Relevant Alg.1 lines
1.	$\underline{C} - \mathcal{S}$	$\pi_i^m.\mathsf{PSet} = \{\mathsf{P}_i \in C, \mathsf{P}_j \in \mathcal{S}\}$ $\pi_i^m.\mathsf{InstSet} = \{\pi_i^m, \pi_j^n\}$	4; 13
2.	$\underline{C} - MB(MB)$	$\pi_i^m.\mathsf{PSet} = \{\mathsf{P}_i \in C, \mathsf{P}_j \in MB\}$ $\pi_i^m.\mathsf{InstSet} = \{\pi_i^m, \pi_j^n\}$	4; 13
3.	$\underline{C} - MB(\mathcal{S})$	$\pi_i^m.\mathsf{PSet} = \{\mathsf{P}_i \in C, \mathsf{P}_k \in MB, \mathsf{P}_j \in \mathcal{S}\}$ $\pi_i^m.\mathsf{InstSet} = \{\pi_i^m, \pi_k^l\}$	16; 37
4.	$\underline{C} - MB - \mathcal{S}$	$\pi_i^m.\mathsf{PSet} = \{\mathsf{P}_i \in C, \mathsf{P}_k \in MB, \mathsf{P}_j \in \mathcal{S}\}$ $\pi_i^m.\mathsf{InstSet} = \{\pi_i^m, \pi_k^l, \pi_k^p, \pi_j^n\}$	16, 18; 37
5.	$\underline{MB} - \mathcal{S}$	$\pi_i^m.\mathsf{PSet} = \{\mathsf{P}_i \in MB, \mathsf{P}_j \in \mathcal{S}\}$ $\pi_i^m.\mathsf{InstSet} = \{\pi_i^m, \pi_j^n\}$	4; 13
6.	$\underline{MB(MB)} - C$	$\pi_i^m.\mathsf{PSet} = \{\mathsf{P}_j \in C, \mathsf{P}_i \in MB\}$ $\pi_i^m.\mathsf{InstSet} = \{\pi_i^m, \pi_j^n\}$	23; 37
7.	$\underline{MB(\mathcal{S})} - C$	$\pi_i^m.\mathsf{PSet} = \{\mathsf{P}_j \in C, \mathsf{P}_i \in MB, \mathsf{P}_k \in \mathcal{S}\}$ $\pi_i^m.\mathsf{InstSet} = \{\pi_i^m, \pi_j^n\}$	23, 28; 37
8.	$\mathcal{S} - \underline{MB} - C$	$\pi_i^m.\mathsf{PSet} = \{\mathsf{P}_k \in C, \mathsf{P}_i \in MB, \mathsf{P}_j \in \mathcal{S}\}$ $\pi_i^m.\mathsf{InstSet} = \{\pi_i^m, \pi_i^p, \pi_k^l\}$	7; 13
9.	$C - \underline{MB} - \mathcal{S}$	$\pi_i^m.\mathsf{PSet} = \{\mathsf{P}_j \in C, \mathsf{P}_i \in MB, \mathsf{P}_k \in \mathcal{S}\}$ $\pi_i^m.\mathsf{InstSet} = \{\pi_i^m, \pi_j^n, \pi_i^p, \pi_k^l\}$	23, 28, 31; 32
10.	$\underline{\mathcal{S}} - C$	$\pi_i^m.\mathsf{PSet} = \{\mathsf{P}_j \in C, \mathsf{P}_i \in \mathcal{S}\}$ $\pi_i^m.\mathsf{InstSet} = \{\pi_i^m, \pi_j^n\}$	23; 25
11.	$\underline{\mathcal{S}} - MB$	$\pi_i^m.\mathsf{PSet} = \{\mathsf{P}_j \in MB, \mathsf{P}_i \in \mathcal{S}\}$ $\pi_i^m.\mathsf{InstSet} = \{\pi_i^m, \pi_j^n\}$	4; 13
12.	$\underline{\mathcal{S}} - MB - C$	$\pi_i^m.\mathsf{PSet} = \{\mathsf{P}_k \in C, \mathsf{P}_j \in MB, \mathsf{P}_i \in \mathcal{S}\}$ $\pi_i^m.\mathsf{InstSet} = \{\pi_i^m, \pi_j^n, \pi_j^p, \pi_k^l\}$	4, 11; 13

Figure 5.5: Algorithm Coverage of 3-Party Communication Settings

To show how algorithm describing partnering in 3-party ACCE operates, take as an example row 3. In this case, the party for which the algorithm is run, P_i, is a client, and its partner identifier will be the name of some party $\mathsf{P}_j \in \mathbb{S}$. Thus, the algorithm enters the `IF` on line 2, but not the `IF` on line 3 (which essentially demands that there be "matching conversation" between that instance of the client and an instance of P_j). We enter the `ELSE` on line 14. Via 15, we find the middlebox that is *actually* partnering P_i, which is a middlebox P_k, for an instance π_k^ℓ. In this case, the middlebox will not need the server's aid to authenticate to the client, so the pre-channel keys will not be used; this makes us skip line 17, and we output the partnering sets in line 37.

Now suppose we are in the infrastructure depicted in row 4. The algorithm will behave as above, apart from the fact that the `IF` on line 17 will also be entered. At that point, we find

the matching middlebox-server instances which will compute the pre-channel key for the session between π_i^m, π_k^ℓ. These are: an instance of that same middlebox P_k and an instance of the server who is the purported partner of π_i^m. In line 18 we add that party to the partnering instance-set of π_i^m, and these values are returned in line 37.

The other cases follow similarly.

Session freshness in 3-(S)ACCE: Like in 2-party ACCE, for the 3-(S)ACCE-channel-security game, a notion of freshness is needed. In our case, this is as follows.

Definition 15. [**3-(S)ACCE Freshness.**] *An instance π_i^m of P_i is* fresh *with intended partner P_j if the following conditions hold:*

- $\pi_i^m.\alpha = 1$ *with* $\pi_i^m.\mathsf{pid} = \mathsf{P}_j.\mathsf{name}$.
- *All parties in $\pi_i^m.\mathsf{PSet}$ are uncorrupted. Note that this includes P_i itself.*
- *No* Reveal *query was made to any of the instances in $\pi_i^m.\mathsf{InstSet}$, which includes π_i^m itself.*

Correctness in 3-(S)ACCE. We extend the definition of correctness (i.e., 2(S)ACCE matching-conversation) in terms of the partnering algorithm presented above. In particular, we demand that, for any instance π_i^m ending in an accepting state with partnering instance-set $\pi_i^m.\mathsf{InstSet}$ and partnering party-set $\pi_i^m.\mathsf{PSet}$, the following conditions hold.

1. If $|\pi_i^m.\mathsf{InstSet}| = 2$, then both instances in $\pi_i^m.\mathsf{InstSet}$ compute the same channel key ck.
2. Consider the case of $|\pi_i^m.\mathsf{InstSet}| = 4$. Let $\pi_j^n, \pi_k^\ell \in \pi_i^m.\mathsf{InstSet}$ for $\mathsf{P}_i, \mathsf{P}_j \in \pi_i^m.\mathsf{PSet}$, such that π_i^m, π_j^n share a session identifier. Then, the following holds:

   a). Any 2-partnered instances in $\pi_i^m.\mathsf{InstSet}$ compute the same channel key. So,
   - π_i^m and π_j^n compute the same channel key x, i.e., $\pi_j^n.\mathsf{ck} = \pi_i^m.\mathsf{ck}$ and $x := \pi_i^m.\mathsf{ck}$;
   - the other two instances in $\pi_i^m.\mathsf{InstSet}$ (i.e. π_k^ℓ and $\pi_i^m.\mathsf{InstSet} \setminus \{\pi_i^m, \pi_j^n, \pi_k^\ell\}$) compute the same channel key x';

   b). Furthermore, the pre-channel key computed in the middlebox-server session must be consistent with that of the client-middlebox channel key. Thus,
   - If P_i or P_j is a client, then $x = f(\pi_k^\ell.\mathsf{pck}, \pi_i^m.\tau)$.
   - If neither P_i nor P_j is a client, it holds that $x' = f(\pi_i^m.\mathsf{pck}, \pi_k^\ell.\tau)$.

3-(S)ACCE Properties & Adversarial Interactions.

In this section, we define four security notions, which together constitute the definition of 3-(S)ACCE-security. Each notion is defined in terms of a game, which the adversary plays against a challenger by interacting with a number of oracles. Many of these are standard 2-ACCE oracles, for which we need to adapt the syntax. We also need an additional party-registration oracle.

New Oracles. In addition to the 2-ACCE oracles, we require the new party-registration oracle we present below.

- $\mathsf{RegParty}(\mathsf{P}_i, \Omega, \mathsf{dest})$: taking as input a party P_i, a content Ω (possibly set to a special symbol $\perp$), and a so-called *destination* $\mathsf{dest} \in \{\text{“}\texttt{MB}''\text{, “}\texttt{Server}''\} \cup \mathbb{S}$. This query works in three modes.

- **The server mode**, with input $(\mathsf{P}_i, \Omega,$ "$\texttt{Server}''$). This query registers P_i as a server (unless this has been done before), outputting, for each subcontent $\omega \in \Omega$, a tuple consisting of a secret key, a public key, and a certificate for that public key on the server's name. The content Ω is added to the server's already-registered content. The adversary receives the certificates and public keys.
- **The middlebox mode**, with input $(\mathsf{P}_i, \perp,$ "$\texttt{MB}''$). The query registers P_i as middlebox, outputting a single secret key, public key, and certificate for that public key associated with P_i's name (the latter two are also output to the adversary).
- **The contract mode**, with input $(\mathsf{P}_i, \Omega, \mathsf{P}_j)$ and $\mathsf{P}_i \in \mathbb{MB}$, $\mathsf{P}_j \in \mathbb{S}$, both registered parties such that $\Omega \subseteq \Omega^j$ (the requested content is a subset of the server's content). This oracle registers locally in the server's hashtable $\mathsf{P}_j.\mathsf{Contracts}$ the entry P_j, Ω, and it adds Ω to the middlebox's content Ω^i_j. Then a set of secret keys, public keys, and certificates are given to P_i (one tuple of keys with the certificate per sub-content). In some cases, the secret key may be set to $\perp$. All the public information is also given to the adversary.

We define 3-(S)ACCE-security as a union of four properties: *entity authentication*, *channel security*, *accountability*, and *content soundness*, presented in more detail below.

For each game, we consider a PPT adversary $\mathcal{A}$ playing against a challenger and interacting with various party instances sequentially or concurrently by means of the above-mentioned oracles. We quantify the attack in terms of the number N_{P} of parties in the system (and in particular, also the number of clients N_C, the number of servers $N_{\mathcal{S}}$, and the number of middlebox parties N_{MB}), the number q_n of $\mathsf{NewSession}$ queries that the adversary makes, and the number t of the total of queries made by the $\mathcal{A}$. The time-complexity of the adversary is determined by t and q_n.

Entity Authentication (EA). In the *entity authentication game*, the adversary $\mathcal{A}$ is given access to the new oracle $\mathsf{RegParty}$, as well as the traditional 2-ACCE oracles. Finally, $\mathcal{A}$ ends the game by outputting a special string "Finished" to its challenger. The adversary is said to *win* the EA game if there exists a party instance π^m_i *maliciously accepting* a partner $\mathsf{P}_j \in \{\mathbb{S}, \mathbb{MB}\}$, according to the following definition.

Definition 16. **[Winning conditions – EA game.]** *An instance π^m_i of some party P_i is said to* maliciously accept *with partner $\mathsf{P}_j \in \{\mathbb{S}, \mathbb{MB}\}$ if the following holds:*

- $\pi^m_i.\alpha = 1$ *with* $\pi^m_i.\mathsf{pid} = \mathsf{P}_j.\mathsf{name} \neq$ "Client''*;*
- *No party in* $\pi^m_i.\mathsf{PSet}$ *is corrupted, no party in* $\pi^m_i.\mathsf{InstSet}$ *was queried in* Reveal *queries;*
- *There exists no unique* $\pi^n_j \in \mathsf{P}_j.\mathsf{Instances}$ *such that* $\pi^n_j.\mathsf{sid} = \pi^m_i.\mathsf{sid}$*;*
- *If* $\mathsf{P}_i \in \mathcal{C}$*, there exists no party* $\mathsf{P}_k \in \mathbb{MB}$ *such that:* $\mathsf{RegParty}(\mathsf{P}_k, \cdot, \mathsf{P}_j)$ *has been queried, and there exists an instance* $\pi^\ell_k \in \pi^m_i.\mathsf{InstSet}$.

The adversary's advantage, denoted $\mathsf{Adv}^{\mathsf{EA}}_{\Pi}(\mathcal{A})$, is defined as its winning probability i.e.:

$$\mathsf{Adv}^{\mathsf{EA}}_{\Pi}(\mathcal{A}) := \mathsf{Pr}[\mathcal{A} \text{ wins the EA game}],$$

where the probability is taken over the random coins of all the N_{P} parties in the system.

Channel Security (CS). In the *channel security game*, the adversary $\mathcal{A}$ can use all the oracles (including $\mathsf{RegParty}$) adaptively. Finally, the adversary outputs a tuple consisting of a fresh party instance π^j_i (in the sense of Definition 15) and a bit b'. The winning condition is defined below:

Definition 17. [**Winning Conditions – CS Game.**] *We say an adversary* $\mathcal{A}$ breaks the channel security *of a 3-(S)ACCE protocol, if it terminates the channel security game by outputting a tuple consisting of a party instance* π_i^j *and a bit* b' *such that:*

- π_i^m *is fresh with partner* P_j*;*
- $\pi_i^m.\mathsf{b} = b'$.

The advantage of the adversary $\mathcal{A}$ is defined as follows:

$$\mathsf{Adv}_{\Pi}^{\mathsf{CS}}(\mathcal{A}) := \left|\mathsf{Pr}[\mathcal{A} \text{ wins the CS game}] - \frac{1}{2}\right|,$$

where the probability is taken over the random coins of all the N_{P} parties in the system.

Accountability (Acc). In the *accountability* game the adversaries interacts arbitrarily with the challenger by means of all the oracles presented in the previous section. Finally, $\mathcal{A}$ halts by outputting a "Finished" string to its challenge. The adversary $\mathcal{A}$ wins if there exists an instance π_i^m of a client P_i such that the following winning condition applies.

Definition 18. [**Winning Conditions – Acc.**] *We say that an adversary* $\mathcal{A}$ breaks the accountability *for instance* π_i^m *of* $\mathsf{P}_i \in \mathcal{C}$*, if:*

- $\pi_i^m.\alpha = 1$ *such that* $\pi_i^m.\mathsf{pid} = \mathsf{P}_j.\mathsf{name}$ *for an* uncorrupted $\mathsf{P}_j \in \mathbb{S}$*;*
- *There exists no instance* $\pi_j^n \in \mathsf{P}_j.\mathsf{Instances}$ *such that* $\pi_j^n.\mathsf{ck} = \pi_i^m.\mathsf{ck}$*;*
- *There exists no probabilistic algorithm* S *polynomial in a given security parameter of the model, such that when* S *is given the view of* P_j *(namely all instances* $\pi_j^n \in \mathsf{P}_j.\mathsf{Instances}$ *with all their attributes),* S *outputs* $\pi_i^m.\mathsf{ck}$*.*

The adversary's advantage is defined as its winning probability, i.e.:

$$\mathsf{Adv}_{\Pi}^{\mathsf{Acc}}(\mathcal{A}) := \mathsf{Pr}[\mathcal{A} \text{ wins the Acc game}],$$

where the probability is taken over the random coins of all the N_{P} parties in the system.

Content Soundness (CSound). In the *content soundness* game, the adversary $\mathcal{A}$ can once again query oracles arbitrarily, and ends the game with a "Finished" message to the challenger. The adversary wins if there exist: a client instance π_i^m and a content ω such that $\mathcal{A}$ will break the soundness of that content in the definition below.

Definition 19. [**Winning Conditions – CSound.**] *We say an adversary* $\mathcal{A}$ *breaks soundness of an arbitrarily fixed context* ω *an 3-(S)ACCE protocol if there exists a client-instance* $\pi_i^m \in \mathsf{P}_i.\mathsf{Instances}$ *for* $\mathsf{P}_i \in \mathcal{C}$ *and* $\pi_i^m.\alpha = 1$*, and there exists no server-instance* π_u^ℓ *of a party* $\mathsf{P}_u \in \mathbb{S}$ *such that:* $\pi_i^m.\mathsf{pid} = \mathsf{P}_u.\mathsf{name}$*,* $\omega \in \Omega^u$*, and* $\pi_i^m.\mathsf{sid} = \pi_u^\ell.\mathsf{sid}$*, and the following conditions simultaneously hold:*

- $\omega \in \pi_i^m.\mathsf{rec}\tau$*;*
- *there exists no instance* π_k^p *of a party* P_k *such that* $\pi_i^m.\mathsf{sid} = \pi_k^p.\mathsf{sid}$ *and, by defining* $\Omega_k := \{\hat{\omega} \ : \ \mathsf{RegParty}(\mathsf{P}_k, \Omega, \cdot)$ *queried and* $\hat{\omega} \in \Omega\}$*, it holds that* $\omega \in \Omega_k$*;*
- *it holds that any party* P_x *such that* $\mathsf{RegParty}(\mathsf{P}_x, \Omega, \cdot)$ *was queried, with* $\omega \in \Omega$*,* P_x *is uncorrupted.*

- *Furthermore, it holds that no party* P_y *such that* $\mathsf{RegParty}(\mathsf{P}_y, \Omega, \text{“}\texttt{Server}''$) *was queried, with* $\omega \in \Omega$*, is corrupted.*

The adversary's advantage is defined as its winning probability, i.e:

$$\mathsf{Adv}^{\mathsf{CSound}}_{\Pi}(\mathcal{A}) := \mathsf{Pr}[\mathcal{A} \text{ wins the CSound game}],$$

where the probability is taken over the random coins of all the N_{P} parties in the system.

5.3 Proxied TLS Connection: Keyless SSL

5.3.1 The Keyless SSL Protocol

The TLS protocol supports many versions, extensions, and ciphersuites. Each TLS connection begins with a *handshake* that negotiates the version and other protocol parameters and then executes an authenticated key exchange. In this section, we give a description of the proxied TLS in Keyless SSL.

In proxied TLS, the client C wants to connect to a server $\mathcal{S}$ and is instead redirected to a geographically close CDN edge server (or middlebox) MB that serves cached content from $\mathcal{S}$. The server $\mathcal{S}$ trusts MB enough to allow it to decrypt requests and encrypt responses for this content. However, unlike in classical CDNs, $\mathcal{S}$ does not want MB to impersonate it indefinitely; instead, $\mathcal{S}$ wants to keep some control over its long-term private keys.

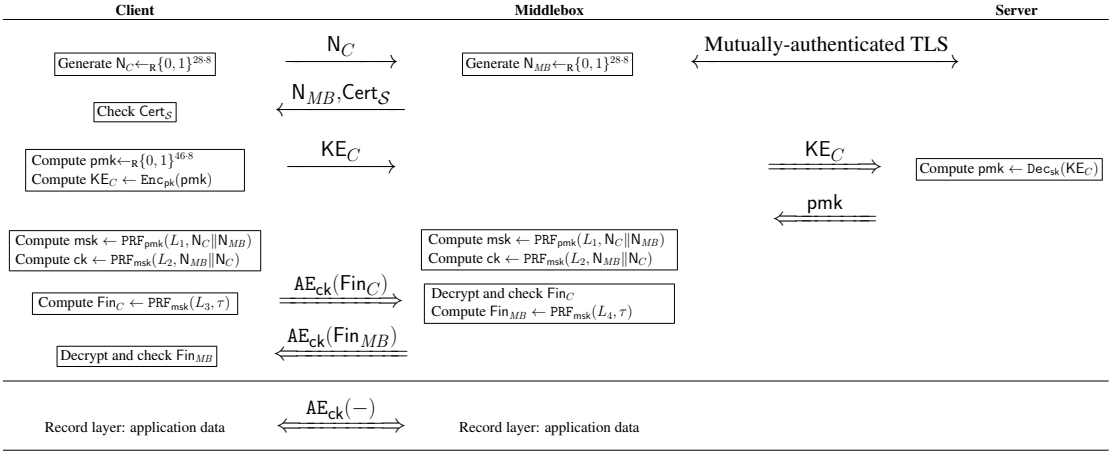


Figure 5.6: The Keyless SSL architecture as implemented by Cloudflare using TLS-RSA.

In both TLS-RSA and TLS-DHE, the server authenticates by proving possession of its certificate private key, using it to either decrypt some secret (TLS-RSA) or to sign some value (TLS-DHE). The key observation of proxied TLS, as originally developed by CloudFlare as Keyless SSL is that the middlebox MB does not need to be given the server's private key as long as it can query the latter when it needs to decrypt or sign some value with the private key.

This leads to the design in Figure 5.7. The client C and middlebox MB execute a standard TLS-RSA or TLS-DHE handshake, where MB uses $\mathcal{S}$'s certificate. In TLS-RSA, C sends a client key exchange message KE_C containing the encrypted pmk and MB forwards it to $\mathcal{S}$'s key server, which decrypts and returns pmk to MB. In TLS-DHE, MB generates the DHE keypair, composes the hashed value sv that needs to be signed, and sends it to $\mathcal{S}$'s key server, which signs and returns the signature. All other computations are performed by MB with no assistance from $\mathcal{S}$.

The queries from MB to the key server $\mathcal{S}$ are performed over a mutually-authenticated TLS channel. CloudFlare issues a client certificate (with a distinguished issuer) to each edge server MB, and a key server certificate to each $\mathcal{S}$.

Keyless SSL is engineered for high performance. Most of the computation can be performed by edge servers; the key server can remain oblivious of the details of TLS. The additional cost of proxying is reduced to a single round-trip between the edge server and the key server. Furthermore, by using session resumption, the edge server can use the same master secret msk over many TLS connections with the same client, without needing to recontact S.

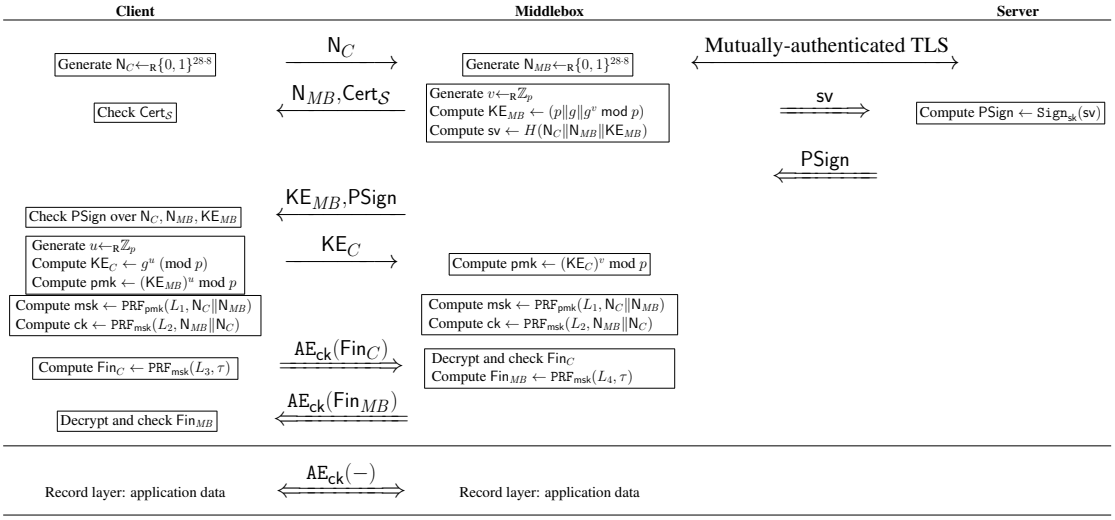


Figure 5.7: The Keyless SSL architecture as implemented by Cloudflare: TLS-DHE.

5.3.2 Security Goals of Keyless SSL

The security goals for Keyless SSL were informally described in [101], where the authors observed that the addition of the third party MB necessitated a few new security goals.

In classic 2-party TLS with server-only authentication, as long as a server's private key is kept secret, we expect (i) *server-to-client authentication*, which says that the attacker cannot impersonate the server or otherwise interfere in handshakes between honest clients and the server, and (ii) *channel security*, which says that the attacker cannot read, alter, or insert application data on connections between an honest client and an honest server. These goals must hold in a threat model in which the attacker controls the network and any number of dishonest clients and malicious or compromised servers. Many variants of these goals have been previously formalized and proved for TLS. In this paper, we assume that classical TLS satisfies them.

In proxied TLS, the main new threat is that we need to consider malicious or compromised middleboxes MB. In "classic" CDNs, MB holds a long-term private key identifying $\mathcal{S}$, and MB uses it on $\mathcal{S}$'s behalf on the C–MB side of the TLS-connections. So, any attack that leaks secrets stored at MB (e.g., HeartBleed) could compromise this private key and allow the adversary to then impersonate $\mathcal{S}$.

In Keyless SSL, the server keeps the private key and may even store it securely in a hardware security module. So, the real threat is from attackers who learn the private key of some middlebox MB and use it to query the key server and thereby impersonate the server. The key goal is that once the key server learns of this compromise and de-authorizes the middlebox's certificate, the attacker should not be able to interfere with connections any longer.

To reflect this intuition, [101] presents three goals for proxied TLS. The first two, *key-server-to-client* and *edge-server-to-client* authentication, generalize the authentication goals to three parties. The third generalizes *channel security*:

> The adversary cannot read or insert messages on the authenticated encryption channel between the client and edge server, provided that the client and edge server's session keys were not compromised, and the long-term private key of the origin server that the client thinks it is talking to was not compromised, and no edge server's private key was compromised between the time when the client sent its first message and accepts.

That is, the confidentiality and integrity of a proxied TLS connection is guaranteed only if the server's private key is kept secret, and also only if *all middleboxes' private keys are secret until the end of the handshake*.

Are these new security definitions adequate? Does Keyless SSL satisfy them? [101] presents informal arguments about why these properties hold in Keyless SSL. In the rest of this section, we seek to answer these questions through a detailed cryptographic analysis. But first, we show that even with the informal definitions above, Keyless SSL is vulnerable to important attacks.

A Middlebox Attack on Keyless TLS-RSA. As discussed earlier, TLS-RSA does not provide forward secrecy, so if the server's private key is compromised, the adversary can decrypt previously recorded connections. In Keyless SSL, if we assume that the server's private key is kept secret, we should normally not have to worry about forward secrecy. However, we still need to consider compromised middlebox.

Suppose an attacker records all the messages in a proxied TLS-RSA connection between C, MB, and $\mathcal{S}$. Much later, suppose that the attacker compromises the private key for some middlebox MB' that is authorized to query the key server S. The attacker can use this private key to establish a mutually-authenticated connection with $\mathcal{S}$, it can ask $\mathcal{S}$ to decrypt the encrypted pmk for any previous connection and thereby decrypt its contents. This attack directly contradicts the channel security property of [101] presented above, since we only compromised the edge server after the connection was complete. (The authors of [101] acknowledged this attack and are seeking to fix their definition.) The attack can either be read as a forward secrecy attack (MB compromised after the connection is complete) or a cross-middlebox attack (MB' is compromised to break C's connection with MB).

The attack is particularly worrisome for CDNs because it means that an attacker who compromises some edge server in country A will then be able to decrypt all prior TLS-RSA conversations recorded in any country B. This emphasizes the new risks of proxied TLS: it strictly reduces the security of client-server connections by increasing the attack surface to include middleboxes distributed around the world. We are also in discussions with CloudFlare to fix this attack; all fixes require the key server to do more than just act as a decryption oracle. For example, a minimal fix for the attack would be for the key server to generate the server nonce N_S and then, when the edge server queries it with the two nonces and the encrypted pmk, it directly derives the msk and returns it. This ensures that a compromised MB cannot make decryption queries on old connections (since N_S will be different).

A Cross-Protocol Attack on Keyless TLS-DHE. Proxied TLS-DHE in Keyless SSL is vulnerable to a different attack that also relies on using the key server as an oracle. The key server is willing to sign any hashed value that the middlebox MB provides. Even if the key server wanted to, it could not check that the value it is signing is the hash of a valid TLS-DHE server key exchange message (KE_{MB}). This leads to a cross-protocol attack between Keyless SSL and QUIC.

The QUIC protocol[4] was proposed as a faster alternative to TLS 1.2 by Google and it is trans-

[4] See https://www.chromium.org/quic

parently used (instead of TLS) on most HTTPS connections from the Chrome web browser to servers that support it. QUIC servers reuse the same X.509 server certificates as TLS to execute a handshake similar to TLS-DHE. However, instead of per-connection signatures, QUIC requires each server to sign a long-term server configuration message SCFG that contains a semi-static Diffie-Hellman key. Once a client has obtained and cached a signed SCFG, it can use it for many connections to the server, without needing any new server signatures, until the configuration expires.

The format of the signed value in QUIC is quite different from that of TLS. For example, the signed value begins with the ASCII string `"QUIC server config signature"`. This value is first hashed using SHA-256 and then signed, using ECDSA, for example. The key observation that leads to our attack is that once the signed values in QUIC or TLS are hashed, the key server cannot tell the difference between them.

Suppose the attacker has compromised the private key of some middlebox MB. It then composes a QUIC SCFG message containing its own Diffie-Hellman public value and a large expiry time ($2^{64} - 1$ seconds). It sends this message to the key server, which will sign it thinking it is signing for a TLS-DHE handshake. The attacker can now pretend to be a QUIC server for S until the configuration expires, *even though S never intended to support QUIC*. De-authorizing the middlebox does not stop this attack.

The flaw in Keyless SSL that enables this attack is that the key server blindly signs messages without checking them. For example, the attack is prevented if MB provides the client nonce and key exchange value to the key server S, and S generates the server nonce and compiles the value to-be-signed before hashing and signing it.

The Problem with Session Resumption. Once a client has established a TLS session with a server, it does not have to redo the full handshake on new connections. Instead, it can rely on an abbreviated handshake, comm only called session resumption, that relies on the client and server storing the master secret msk and other parameters of previously established sessions. A variant called session tickets allows servers to offload session storage to clients.

The extensive use of session resumption in modern web browsers is crucial to the efficiency of Keyless SSL, since it means that for a majority of connections, the edge server need not contact the origin server. However, resumption also allows an adversary who has compromised an edge server to create a session ticket with a long expiry time, and thereafter impersonate the server on resumed connections until the session expires, even if the origin server de-authorizes the edge server immediately. This attack is difficult to prevent without changing the way web browsers work, and so for strong security against malicious middleboxes, we forbid session resumption in proxied TLS, except in special cases.

Towards a Stronger Security Definition. We have described two concrete attacks that break the intended security goals of Keyless SSL. Before fixing Keyless SSL, however, it is worth asking if the original goals were the right ones in the first place, or whether they are too weak and need to be strengthened.

The channel security definition from [101] quoted above only applies if none of the middlebox private keys is compromised. Suppose an honest client managed to connect to an uncompromised middlebox MB in country A. The definition says that this connection is not guaranteed to be secure if the attacker can compromise some edge server MB' in any country B. However, it seems valuable to strengthen the goals to require security for connections to honest middleboxes even if other middleboxes were compromised.

The authentication goals in [101] are also quite weak. The client authenticates the key server, the edge server authenticates the key server, but there is no guarantee that the client and edge server agree on the identity of the key server. That is, the definitions allow the case where the

client thinks it is connected to $\mathcal{S}$ via MB but MB thinks it is connected to $\mathcal{S}'$. In the CDN context, MB would then be serving content from $\mathcal{S}'$ (instead of $\mathcal{S}$) to C, and this becomes a serious attack which is not forbidden by the authentication goals.

More generally, extending two-party secure channel definitions to three-party scenarios like proxied TLS requires close attention or it may leave gaps that miss new attacks. Over the next two sections, we explore and present a formal definition for proxied TLS that attempts to close these gaps.

5.4 Our variant of Keyless SSL Protocol: KeylessTLS1.2

Keyless SSL is not 3-(S)ACCE-secure. The attack on TLS-RSA in Section 5.3.2 breaks our channel security definition, since a malicious middlebox MB' can decrypt connections to an honest middlebox MB. Similarly, a malicious middlebox can confuse a honest middlebox about the identity of the server, breaking entity authentication. Session resumption breaks accountability.

5.4.1 Achieving 3-(S)ACCE-security for Keyless SSL

In this section, we introduce a 3-(S)ACCE-secure variant of Keyless SSL, which will preserve the general infrastructure of the original protocol, in the keyless architecture. In this description, we focus mainly on the differences between this variant and the original Keyless SSL protocol.

A more extensive PKI. In the original Keyless SSL architecture, whenever a middlebox registers with the server, for any content, the server forwards the same certificate, which we denote as $\mathsf{Cert}_{MB,\mathcal{S}}$. However, in order to ensure content soundness and entity authentication, we need servers to issue one public key (with a corresponding certificate) for each subcontent ω *and* for each middlebox authorized to deliver that subcontent. We denote a certificate associated with a server P_j which is used by a middlebox P_k for some content ω as $\mathsf{Cert}^{\omega}_{\mathsf{P}_k,\mathsf{P}_j}$.

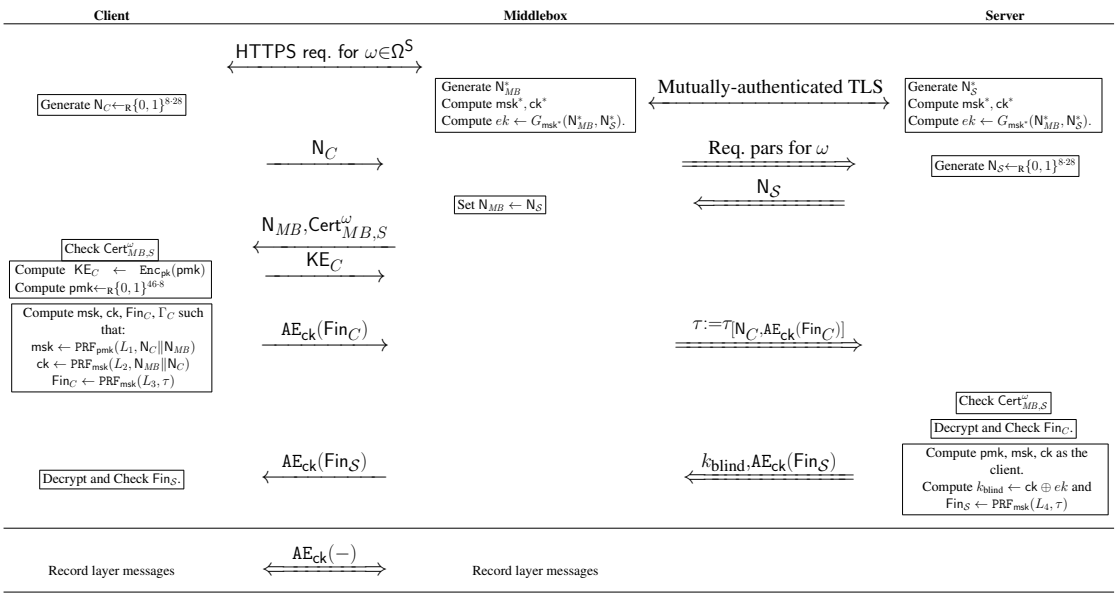


Figure 5.8: An 3-(S)ACCE-secure variant of Keyless SSL: TLS-RSA

The protocol description. We depict our 3-(S)ACCE-secure variant of Keyless SSL in Figure 5.9. We list some key elements of this protocol below.

An export key ek. We begin by describing the handshake between the middlebox and the server. In Keyless SSL this is a generic 2-ACCE-secure protocol; in our design we use a TLS handshake, and then compute an export key ek, which is computed as the result of a pseudorandom function G which is independent of the one used in the TLS handshake itself. We key this function with the master secret used in the TLS handshake between the middlebox and the server, and use it on input the nonces used in that session. For this type of protocol, Brzuska et al. [55] proved that ek is indistinguishable from random from any party other than the two protocol participants.
The key share KE_{MB}. In the traditional Keyless TLS-DHE architecture, the middlebox always generates its own key share KE_{MB} and sends a hash of that and other values to the appropriate server to certify. In our new design, it is the server that generates the key share. This is essential in order to achieve accountability: if the middlebox were able to generate msk, it could use session resumption in a future session, thus winning the accountability game. Note that the server has already run a handshake with the middlebox at this point, and the server has verified the identity of its partner. The middlebox forwards the content ω requested by the client and the two session nonces used in the client-middlebox session; then the server generates the signature of the public key certified in $\mathsf{Cert}^{\omega}_{MB,\mathcal{S}}$.
Computing the channel key. In DHE mode, the server now holds the private DH exponent to the key share KE_{MB}. After the middlebox receives (in its session with the client) the client's key share KE_C and the Finished message Γ_C, it transmits the entire session transcript to the server. The latter verifies: that the nonces in the transcript are those received earlier; that the certificate used by the middlebox in that transcript is the correct one, and finally, that the client's Finished message verified (note that the server can compute pmk, msk, and the channel key ck locally). Finally, the server also computes the Finished message on behalf of the middlebox, and encrypts and authenticates it.

In TLS-RSA mode, the client chooses pmk and sends it to the middlebox, encrypted with the received certified public key. The middlebox forwards the entire session transcript up to, and including, the client's encrypted Finished message. The server checks that the certificate is compatible with the querying middlebox' authentication information (certificate). Then it obtains pmk by decrypting under the secret key corresponding to the certificate the middlebox forwarded. After computing msk and ck, the server verifies the validity of the client's Finished message. Upon successful verification, the server computes and sends the encrypted Finished message the client expects from the middlebox.
Blinding the key. In both modes, the server will first blind the computed channel key ck with the export key ek, thus sending the encrypted Finished message and $ek \oplus \mathsf{ck}$ to the middlebox, who will recover the value ck and use the Finished message. In particular, the blinding is necessary in the security proof in order to reduce the security of our variant to the 2-(S)ACCE security of TLS (we will need to simulate the encrypted messages so that they are consistent with what the adversary expects).

5.4.2 The Security Analysis of the KeylessTLS1.2

In this section, we proved the entity authentication, secure channel, accountability and content soundness properties under some classic properties of the internal cryptographic functions and the proved mEA- and 2-(S)ACCE-security of the unilaterally-authenticated and mutually-authenticated TLS 1.2 handshakes.

The security statement. We state the following informal security statement, proving the security of the KeylessTLS1.2 protocol. We begin by noting that, following the results of Brzuska et al. [55], if G is a PRF that is independent from the key-derivating PRF used in TLS, then the keys ek computed by the middlebox and the server are indistinguishable from random. Furthermore,

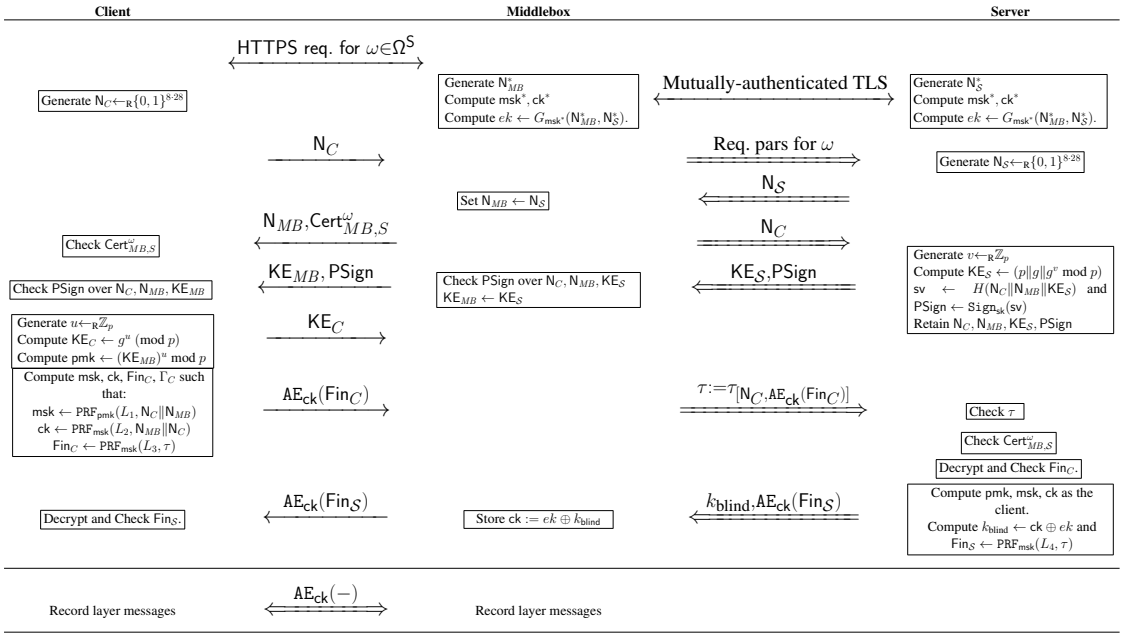


Figure 5.9: An 3-(S)ACCE-secure variant of Keyless SSL: TLS-DHE.

the TLS 1.2 protocol with unilateral authentication was proved to attain 2-SACCE security by Krawczyk et al. [81]; they also proved the same protocol was 2-ACCE-secure.

Theorem 32. [**EA-security of** KeylessTLS1.2**.**] *Let Π be our 3-(S)ACCE variant of Keyless SSL,* KeylessTLS1.2*, let P be the unilaterally-authenticated TLS 1.2 handshake, P', the mutually-authenticated TLS handhsake between middlebox and servers, P'', a mix between unilaterally-authenticated and mutually-authenticated 2-party TLS handshake, and Ψ, the transformation of P' to an AKE protocol by the computation of the export key ek. Consider a (t,q)-adversary $\mathcal{A}$ against the* EA*-security of the protocol Π running at most t queries and creating at most q party instances per party. We denote by $\mathsf{n_P}$ the number of parties in the system, and denote $\mathcal{A}$'s advantage by $\mathsf{Adv}^{\mathsf{EA}}_{\Pi}(\mathcal{A})$. If such an adversary exists, then there exist adversaries $\mathcal{A}_1$ against the* SACCE *security of P, $\mathcal{A}_2$ against the* ACCE *security of P', $\mathcal{A}_3$ against the AKE security of the EAP-TLS protocol composed as $\Psi(P')$ with resulting export key ek, $\mathcal{A}_4$ against the M-EA security of P'' and either: (for TLS-DHE) $\mathcal{A}_5$ against the existential unforgeability (EUF-CMA) of the signature algorithm used to generate* PSign *and $\mathcal{A}_6$ against the hash function H, or (for TLS-RSA) $\mathcal{A}_5$ against the channel security of P, each adversary running in time $t' \sim O(t)$ and instantiating at most $q' = q$ instances per party, such that*

- *For TLS-DHE:*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{EA}}_{\Pi}(\mathcal{A}) \;\leq\; & 2\mathsf{n_P}^2 \cdot \mathsf{Adv}^{\mathsf{ACCE}}_{P'}(\mathcal{A}_2) + \mathsf{n_P}^3 \cdot \mathsf{Adv}^{\mathsf{SACCE}}_{P}(\mathcal{A}_1) + \mathsf{Adv}^{\mathsf{C.Res}}_{H}(\mathcal{A}_6) \\ & + \mathsf{n_P} \cdot \mathsf{Adv}^{\mathsf{Unf}}_{\mathsf{Sign}}(\mathcal{A}_5) + \mathsf{n_P}^3 \cdot \mathsf{Adv}^{\mathsf{AKE}}_{\Psi}(\mathcal{A}_3) + \mathsf{n_P}^3 \cdot \mathsf{Adv}^{\mathsf{M-EA}}_{\Pi''}(\mathcal{A}_4).\end{aligned}$$

- *For TLS-RSA:*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{EA}}_{\Pi}(\mathcal{A}) \;\leq\; & 2\mathsf{n_P}^2 \cdot \mathsf{Adv}^{\mathsf{ACCE}}_{P'}(\mathcal{A}_2) + \mathsf{n_P}^2 \cdot \mathsf{Adv}^{\mathsf{SACCE}}_{P}(\mathcal{A}_1) + \mathsf{n_P}^3 \mathsf{Adv}^{\mathsf{AKE}}_{\Psi}(\mathcal{A}_3) \\ & + \mathsf{n_P}^2 \cdot \mathsf{Adv}^{\mathsf{CS-SACCE}}_{P}(\mathcal{A}_5) + \mathsf{n_P}^3 \cdot \mathsf{Adv}^{\mathsf{M-EA}}_{\Pi''}(\mathcal{A}_4).\end{aligned}$$

Proof. Our proof has the following hops:

Game $\mathbb{G}_0$: This game works as the EA- game stipulated in Section 5.2.3.
Game $\mathbb{G}_1$: This is the same game as the EA-game, except that if the adversary can no longer win if its winning instance π_i^m belongs to a server, i.e., $\mathsf{P}_i \in \mathbb{S}$. We argue that the security loss in this game hop corresponds to $\mathsf{nP}^2 \cdot \mathsf{Adv}^{\mathsf{ACCE}}_{P'}(\mathcal{A}_2)$, accounting for the fact that the reduction must guess the identity of the server that will maliciously accept, and of the middlebox that is being impersonated. Indeed, we note that in our EA definition, the only way the adversary can win if P_i is a server is if the accepting instance π_i^m for which $\mathcal{A}$ wins has to accept for $\pi_i^m.\mathsf{pid} = \mathsf{P}_j.\mathsf{name}$ with $\mathsf{P}_j \in \mathbb{MB}$. Thus,

$$|\mathbf{Pr}[\mathcal{A}_{\mathbb{G}_0} \text{ wins}] - \mathbf{Pr}[\mathcal{A}_{\mathbb{G}_1} \text{ wins}]| \leq \mathsf{nP}^2 \cdot \mathsf{Adv}^{\mathsf{ACCE}}_{P'}(\mathcal{A}_2)$$

Game $\mathbb{G}_2$: This game behaves as $\mathbb{G}_1$, except we now rule out the possibility that P_i is a middlebox. Notably, if that is the case, then its partner only be a server ($\mathsf{P}_j \in \mathbb{S}$). In a similar way as in the previous reduction, we can reduce this to the 2-ACCE-EA security of P', namely,

$$|\mathbf{Pr}[\mathcal{A}_{\mathbb{G}_1} \text{ wins}] - \mathbf{Pr}[\mathcal{A}_{\mathbb{G}_2} \text{ wins}]| \leq \mathsf{nP}^2 \cdot \mathsf{Adv}^{\mathsf{ACCE}}_{P'}(\mathcal{A}_2).$$

Game $\mathbb{G}_3$: We recall that in this game, the adversary may only win against an instance π_i^m of a client $\mathsf{P} \in \mathcal{C}$. In this game, we rule out the possibility that $\pi_i^m.\mathsf{pid} = \mathsf{P}_j.\mathsf{name}$ such that P_j is a server, such that there exists an instance π_j^n such that $\pi_i^m.\mathsf{sid} = \pi_j^n.\mathsf{sid}$. In other words, we are ruling out the possibility of the adversary winning in a direct client-server handshake. In this game-hop we lose $\mathsf{nP}^2 \cdot \mathsf{Adv}^{\mathsf{SACCE}}_{P}(\mathcal{A}_1)$, accounting for the 2-SACCE security in the client-server handshake (namely P):

$$\mathbf{Pr}[\mathcal{A}_{\mathbb{G}_2} \text{ wins}] \leq \mathbf{Pr}[\mathcal{A}_{\mathbb{G}_3} \text{ wins}] + \mathsf{nP}^2 \cdot \mathsf{Adv}^{\mathsf{SACCE}}_{P}(\mathcal{A}_1).$$

We note that at this point, $\mathbb{G}_3$ corresponds to $\mathbb{G}_0$ with the restriction that P_i is a client and the targeted instance π_i^m has the related partnering: $\pi_i^m.\mathsf{pid} = \mathsf{P}_j.\mathsf{name}$ with $\mathsf{P}_j \in \mathbb{S}$ and such that there exists some middlebox P_k and an instance π_k^p such that π_k^p and π_i^m are 2-partnered (they have the same session ID).
Winning game $\mathbb{G}_3$: In this game, we rely quite strongly on the fact that the server P_j, which is uncorrupted, provides distinct certificates per middlebox, per content. For one fixed content ω, the implication is that if P_k and another middlebox P_x both registered for ω with the same server P_j, then the public keys and the certificates used by $\mathsf{P}_k, \mathsf{P}_x$ on behalf of P_j are different.

We first prove that, more than just having a middlebox for a real partner and a server for the intended partner, it also holds that: there exists a matching instance π_k^ℓ such that π_k^ℓ and π_j^n are also 2-partnered, and furthermore, $\pi_i^m.\mathsf{ck} = \varphi(\pi_j^n.\mathsf{pck}, \pi_i^m.\tau)$. In this case, we recall that the partnering in 3-(S)ACCE implies that $\pi_i^m.\mathsf{PSet} = \{\mathsf{P}_i, \mathsf{P}_j, \mathsf{P}_k\}$ and $\pi_i^m.\mathsf{InstSet} = \{\pi_i^m, \pi_j^n, \pi_k^p, \pi_k^\ell\}$.
• **TLS-DHE** To begin with, we focus on the transcript of π_i^m. Assume that this transcript yields the certificate(and public key) $\mathsf{Cert}^{\omega}_{\mathsf{P}_k,\mathsf{P}_j}$. We first rule out the possibility that another party P_x as above, is able to maliciously authenticate to P_i using that certificate. We already ruled out the possibility that P_x impersonates P_k to P_j during the middlebox-server handshake. We now rule out the possibility that the client accepts P_x as though it were P_k, which is bounded, first by the collision-resistance of the hash function H ($\mathsf{Adv}^{\mathsf{C.Res}}_{H}(\mathcal{A}_6)$), and secondly, by the unforgeability in the signature PSign ($\mathsf{nP} \cdot \mathsf{Adv}^{\mathsf{Unf}}_{\mathsf{Sign}}(\mathcal{A}_5)$), accounting for getting which party the signature is generated for.
• **TLS-RSA** In this setting, the equivalent security is guaranteed by the fact that the encryption is under the public key of the purported partner P_k of π_i^m. We have already ruled out middlebox impersonations to the server, which leaves, as an only option, having a party that is not P_k decrypt the encrypted pre-master key. We can build a reduction to the SACCE security of the underlying protocol P, such that the adversary can learn the secret bit of instance π_i^m (by learning the pre-master secret and then computing the channel key). The security loss is given by $\mathsf{nP}^2 \cdot \mathsf{Adv}^{\mathsf{SC-SACCE}}_{P}(\mathcal{A}_5)$

We now resume our proof. We first fix the three parties $\mathsf{P}_i, \mathsf{P}_j, \mathsf{P}_k$, and lose a factor $\mathsf{n_P}^3$.

We also note that the EA definition stipulates that no Reveal query can be made on the instances in π_i^m.InstSet. An intermediate reduction is that, for instances π_k^ℓ and π_j^n as above, we claim that the keys ek these parties compute are indistinguishable from random (as indicated by the AKE property of $\Psi(P')$). Thus, we lose here a term $\mathsf{Adv}_\Psi^{\mathsf{AKE}}(\mathcal{A}_3)$.

We then reduce the remaining winning probability to the M-EA property of P". The M-EA security game simulates simultaneously unilateral server- and mutual authentication TLS handshake sessions. To prove the security reduction, we propose to simulate the security game from the M-EA security game. The most important point in the simulation is the instantiation of the 3-(S)ACCE parties and instances. The servers are defined as a server party in the M-EA game and clients are defined as a client party in the M-EA game. Let π_i^m the instance of a client, π_j^n an instance of a server and π_k^s an instance of a middlebox. Three kind of communications have to be simulated by the M-EA security game:

- A two-party communication between a client instance π_i^m and a server instance π_j^n: This communication is simulated as a 2-SACCE communication. These instances are initialized in $\mathbb{G}_3$ with requests $\mathsf{NewSession}(\pi_i^m, init, \pi_j^n.pid)$ and $\mathsf{NewSession}(\pi_j^n, resp, \pi_i^m.pid)$. With the game $\mathbb{G}_{M-EA}$, there are simulated with respectively the queries $\mathsf{NewSession}^*(\pi_i^m, init, \pi_j^n.pid, 0)$ and $\mathsf{NewSession}^*(\pi_j^n, resp, \pi_i^m.pid, 0)$.

- A two-party communication between a middlebox instance π_k^s and a server instance π_j^n: This communication is simulated as a 2-ACCE communication. These instances are initialized in $\mathbb{G}_3$ with requests $\mathsf{NewSession}(\pi_k^s, init, \pi_j^n.pid)$ and $\mathsf{NewSession}(\pi_i^m, resp, \pi_k^s.pid)$. With the game $\mathbb{G}_{M-EA}$, there are simulated with respectively the queries $\mathsf{NewSession}^*(\pi_k^s, init, \pi_j^n.pid, 1)$ and $\mathsf{NewSession}^*(\pi_j^n, resp, \pi_k^s.pid, 1)$.

- A three-party communication between a client instance π_i^m, a middlebox instance π_k^s and a server instance π_j^n: This communication is simulated as a 2-SACCE communication. Since the middlebox does not use its own certificate and parameters, but use the parameters of the server to establish the key-agreement, the communication can be simulated by a communication between client and server. Two queries $\mathsf{NewSession}^*(\pi_i^m, init, \pi_j^n, 0)$ and $\mathsf{NewSession}^*(\pi_j^n, resp, \pi_i^m, 0)$ are required to instantiate the session.

Let $\mathsf{n_P} = \mathsf{n_c} + \mathsf{n_s} + \mathsf{n_{mb}}$, the number of the party with $\mathsf{n_c}$ the number of client party, $\mathsf{n_{mb}}$ the number of middlebox party and $\mathsf{n_s}$ the number of server party. With this reduction, the game $\mathbb{G}_{M-EA}$ has to simulate $\mathsf{n_s}$ server's party and $\mathsf{n_c} + \mathsf{n_{mb}}$ client's party.

For all the others queries (Reveal, Send and Corrupt), the simulation works as in the M-EA case on the related instances. We note that a peculiarity occurs for the Send oracle, since in order to simulate correctly the record-layer transcript of the middlebox-server session between π_k^ℓ and π_j^n, the M-EA-adversary will query Reveal on this session (this is allowed in the EA game) and simulate the encryption and decryption oracles. In particular, to simulate sending: $(ek \oplus \mathsf{ck}, \Gamma_{\mathcal{S}})$, the M-EA-adversary chooses at random a value r and encrypts $r, \Gamma_{\mathcal{S}}$, sending this to the middlebox.

This simulation is perfect. In particular, this is guaranteed by the earlier reduction concerning the indistinguishability from random of the blinding keys ek. If the adversary $\mathcal{A}_{\mathbb{G}_3}$ wins for some session π_i^m, then $\mathcal{A}_1$ verifies if there exists a unique instance π_k^p such that $\pi_i^m.\mathsf{sid} = \pi_k^p.\mathsf{sid}$. If not, then $\mathcal{A}_1$ will have π_i^m as its own winning instance; else, it must be that $\mathcal{A}_3$ will find an instance π_j^n of P_j holding the pre-channel key corresponding to $\pi_i^m.\mathsf{ck}$, but such that there exists no matching, unique π_k^ℓ, also holding that pre-channel key, so that π_k^ℓ, π_j^n are 2-partnered; in that case, $\mathcal{A}_3$ wins.

This concludes the proof and yields the indicated bound.

□

Theorem 33. [**CS-security of** KeylessTLS1.2**.**] *Let Π be our 3-(S)ACCE variant of Keyless SSL,* KeylessTLS1.2*, let P be the unilaterally-authenticated TLS 1.2 handshake, P', the mutually-authenticated TLS 1.2 handhsake between middlebox and servers, P'', a mix between unilaterally-authenticated and mutually-authenticated 2-party TLS handshake, and Ψ, the transformation of P' to an AKE protocol by the computation of the export key ek. Consider a (t,q)-adversary $\mathcal{A}$ against the* CS*-security of the protocol Π running at most t queries and creating at most q party instances per party. We denote by $\mathsf{n_P}$ the number of parties in the system, and denote $\mathcal{A}$'s advantage by $\mathsf{Adv}^{\mathsf{CS}}_{\Pi}(\mathcal{A})$. If such an adversary exists, then there exist adversaries $\mathcal{A}_1$ against the* SACCE *security of P, $\mathcal{A}_2$ against the* ACCE *security of P', $\mathcal{A}_3$ against the AKE security of the EAP-TLS protocol composed as $\Psi(P')$ with resulting export key ek, $\mathcal{A}_4$ against the M-EA security of P'', $\mathcal{A}_5$ against the* AE *security of P with the computed channel keys, $\mathcal{A}_6$ against the AE security of P' given the computed keys and either: $\mathcal{A}_7$ against the existential unforgeability (EUF-CMA) of the signature algorithm used to generate* PSign *and $\mathcal{A}_8$ against the collision resistance of the hash function H (for TLS-DHE), or $\mathcal{A}_7$ against the channel security of P (for TLS-RSA), each adversary running in time $t' \sim O(t)$ and instantiating at most $q' = q$ instances per party, such that*

- *For TLS-DHE:*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{CS}}_{\Pi}(\mathcal{A}) &\leq 2\mathsf{n_P}^2\mathsf{Adv}^{\mathsf{ACCE}}_{P'}(\mathcal{A}_2) + \mathsf{n_P}^2\mathsf{Adv}^{\mathsf{SACCE}}_{P}(\mathcal{A}_1) + \mathsf{n_P}\mathsf{Adv}^{\mathsf{Unf}}_{\mathsf{Sign}}(\mathcal{A}_7) + \mathsf{Adv}^{\mathsf{C.Res}}_{H}(\mathcal{A}_8)\\ &+ \mathsf{n_P}^3(\mathsf{Adv}^{\mathsf{AKE}}_{\Psi}(\mathcal{A}_3) + \mathsf{Adv}^{\mathsf{M-EA}}_{\Pi''}(\mathcal{A}_4) + \mathsf{Adv}^{\mathsf{ae}}_{P}(\mathcal{A}_5) + \mathsf{Adv}^{\mathsf{ae}}_{P'}(\mathcal{A}_6)).\end{aligned}$$

- *For TLS-RSA:*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{CS}}_{\Pi}(\mathcal{A}) &\leq 2\mathsf{n_P}^2\mathsf{Adv}^{\mathsf{ACCE}}_{P'}(\mathcal{A}_2) + \mathsf{n_P}^2 \cdot \mathsf{Adv}^{\mathsf{SACCE}}_{P}(\mathcal{A}_1) + \mathsf{n_P}^2\mathsf{Adv}^{\mathsf{CS-SACCE}}_{P}(\mathcal{A}_7)\\ &+ \mathsf{n_P}^3(\mathsf{Adv}^{\mathsf{AKE}}_{\Psi}(\mathcal{A}_3) + \mathsf{Adv}^{\mathsf{M-EA}}_{\Pi''}(\mathcal{A}_4) + \mathsf{Adv}^{\mathsf{ae}}_{P}(\mathcal{A}_5) + \mathsf{Adv}^{\mathsf{ae}}_{P'}(\mathcal{A}_6)).\end{aligned}$$

Proof. Our proof has the following hops:
Game $\mathbb{G}_0$: This game works as the CS-game stipulated section 5.2.3.
Games $\mathbb{G}_0$-$\mathbb{G}_3$: We make similar successive reductions as in the previous proof to obtain the game $\mathbb{G}_3$ which behaves as the original game but with the restriction that P_i is a client, and for the targeted instance π_i^m it holds that: $\pi_i^m.\mathsf{pid} = \mathsf{P}_j.\mathsf{name}$ with $\mathsf{P}_j \in \mathbb{S}$ and such that there exists some middlebox P_k and an instance π_k^p such that π_k^p and π_i^m are 2-partnered (they have the same session ID).

The loss through to game $\mathbb{G}_3$ is as follows:

$$\mathbf{Pr}[\mathcal{A}_{\mathbb{G}_2}\text{ wins}] \leq \mathbf{Pr}[\mathcal{A}_{\mathbb{G}_3}\text{ wins}] + \mathsf{n_P}^2 \cdot \mathsf{Adv}^{\mathsf{SACCE}}_{P}(\mathcal{A}_1) + 2\mathsf{n_P}^2 \cdot \mathsf{Adv}^{\mathsf{ACCE}}_{P'}(\mathcal{A}_2).$$

Winning game 3: This proof goes similarly to the one before, except that in the simulation of adversaries $\mathcal{A}_1$ and $\mathcal{A}_2$ we use a simulation akin to that of the CS-game simulating party as the previous proof and additionally, in particular with respect to simulating the encryption and decryption queries which have been included in the simulation. The total success probability of the adversary is given by:

- For TLS-DHE:

$$\begin{aligned}\Pr[\mathcal{A}_{\mathbb{G}_3}\text{ wins}] &\leq \frac{1}{2} + \mathsf{n_P}^3 \cdot (\mathsf{Adv}^{\mathsf{AKE}}_{\Psi}(\mathcal{A}_3) + \mathsf{Adv}^{\mathsf{M-EA}}_{\Pi''}(\mathcal{A}_4) + \mathsf{Adv}^{\mathsf{ae}}_{P}(\mathcal{A}_5)\\ &+ \mathsf{Adv}^{\mathsf{ae}}_{P'}(\mathcal{A}_6)) + \mathsf{n_P} \cdot \mathsf{Adv}^{\mathsf{Unf}}_{\mathsf{Sign}}(\mathcal{A}_7) + \mathsf{Adv}^{\mathsf{C.Res}}_{H}(\mathcal{A}_8).\end{aligned}$$

- For TLS-RSA:

$$\begin{aligned}\Pr[\mathcal{A}_{\mathbb{G}_3}\text{ wins}] &\leq \frac{1}{2} + \mathsf{n_P}^3 \cdot (\mathsf{Adv}^{\mathsf{AKE}}_{\Psi}(\mathcal{A}_3) + \mathsf{Adv}^{\mathsf{M-EA}}_{\Pi''}(\mathcal{A}_4) + \mathsf{Adv}^{\mathsf{ae}}_{P}(\mathcal{A}_5)\\ &+ \mathsf{Adv}^{\mathsf{ae}}_{P'}(\mathcal{A}_6)) + \mathsf{n_P}^2 \cdot \mathsf{Adv}^{\mathsf{CS-SACCE}}_{P}(\mathcal{A}_7).\end{aligned}$$

□

Theorem 34. [Acc**-security of** KeylessTLS1.2**.**] *Let Π be our 3-(S)ACCE variant of Keyless SSL,* KeylessTLS1.2*, let P be the unilaterally-authenticated TLS 1.2 handshake, P', the mutually-authenticated TLS 1.2 handshake between middlebox and servers, and Ψ, the transformation of P' to an AKE protocol by the computation of the export key ek. Consider a (t,q)-adversary $\mathcal{A}$ against the* Acc*-security of the protocol Π running at most t queries and creating at most q party instances per party. We denote by $\mathsf{n_P}$ the number of parties in the system, and denote $\mathcal{A}$'s advantage by $\mathsf{Adv}^{\mathsf{Acc}}_{\Pi}(\mathcal{A})$. If such an adversary exists, then there exists adversary $\mathcal{A}_1$ against the* SACCE *security of P running in time $t' \sim O(t)$ and instantiating at most $q' = q$ instances per party, and $\mathcal{A}_2$ against the existential unforgeability (EUF-CMA) of the signature algorithm used to generate* PSign *(for TLS-DHE) such that:*

- *For TLS-DHE:* $\mathsf{Adv}^{\mathsf{Acc}}_{\Pi}(\mathcal{A}) \leq 2 \cdot {\mathsf{n_P}}^2 \cdot \mathsf{Adv}^{\mathsf{SACCE}}_{P}(\mathcal{A}_1) + {\mathsf{n_P}}^2 \cdot \mathsf{Adv}^{\mathsf{Unf}}_{\mathsf{Sign}}(\mathcal{A}_2)$.
- *For TLS-RSA:* $\mathsf{Adv}^{\mathsf{Acc}}_{\Pi}(\mathcal{A}) \leq 3 \cdot {\mathsf{n_P}}^2 \cdot \mathsf{Adv}^{\mathsf{SACCE}}_{P}(\mathcal{A}_1)$.

Proof. Our proof has the following hops:

Game 0: This game works as the Acc- game stipulated section 5.2.3. We say that an adversary $\mathcal{A}$ *breaks the accountability* for an instance π_i^m with $\mathsf{P}_i \in \mathcal{C}$, if the following conditions are verified:

(a) $\pi_i^m.\alpha = 1$ such that $\pi_i^m.\mathsf{pid} = \mathsf{P}_j.\mathsf{name}$ for an *uncorrupted* $\mathsf{P}_j \in \mathbb{S}$ and $\pi_i^m.\mathsf{ck} = \mathsf{ck}$;

(b) There exists no instance $\pi_j^n \in \mathsf{P}_j.\mathsf{Instances}$ such that $\pi_j^n.\mathsf{ck} = \pi_i^m.\mathsf{ck}$;

(c) There exists no probabilistic polynomial algorithm S which given the view of P_j (namely all instances $\pi_j^n \in \mathsf{P}_j.\mathsf{Instances}$ with all their attributes), outputs ck.

Our strategy in this proof is to show that, whenever condition (a) holds, then either the reverse of (b) or the reverse of condition (c) holds (except with negligible probability). Our ultimate task will be to construct a simulator that fulfills condition (c). We first rule out a few exceptional cases.
Game $\mathbb{G}_1$: The adversary begins by guessing the identities of the targetted client P_i and of the server P_j such that π_i^m is the instance for which accountability is broken, and for which it holds $\pi_i^m.\mathsf{pid} = \mathsf{P}_j.\mathsf{name}$. As a consequence, we have:

$$\mathbf{Pr}[\mathcal{A}_{\mathbb{G}_0}\ \mathsf{wins}] \leq {\mathsf{n_P}}^2 \cdot \mathbf{Pr}[\mathcal{A}_{\mathbb{G}_1}\ \mathsf{wins}].$$

There are two cases.
• **No middlebox.** In this case, we assume that there is no middlebox P_k such that there exists an instance π_k^p with $\pi_k^p.\mathsf{sid} = \pi_i^m.\mathsf{sid}$. This implies that either there is a true session between π_i^m and some instance π_j^n of P_j (no impersonation takes place), or there was a server impersonation on P_j. The latter only happens with probability $\mathsf{Adv}^{\mathsf{SACCE}}_{P}(\mathcal{A}_1)$. The former implies directly that the opposite of condition (b) is true, since the TLS 1.2 handshake has perfect correctness.
• **With middlebox.** In this case, we assume the existence of a middlebox P_k such that there exists an instance π_k^p with $\pi_k^p.\mathsf{sid} = \pi_i^m.\mathsf{sid}$. There are two options.

First the middlebox could try to run the handshake on its own (there exist no instances π_j^n, π_x^ℓ such that $\pi_i^m.\mathsf{pck}$ is in fact $\pi_i^m.\mathsf{ck}$ as per the protocol description). Note that for this first option it does not necessarily have to hold that π_x^ℓ is an instance of P_k, i.e, the middlebox talking to the client. In that case we can construct a reduction from this case to the server-impersonation security of the protocol P' (recall that the honest server P_j cannot be corrupted). In the case of TLS-DHE, this is equivalent to forging the signature PSign. For TLS-RSA, this is equivalent to decrypting the premaster secret. We lose a term $\mathsf{Adv}^{\mathsf{SACCE}}_{P}(\mathcal{A}_1)$.

The second option is that there exist instances π_j^n, π_x^ℓ such that $\pi_i^m.\mathsf{pck}$ is in fact $\pi_i^m.\mathsf{ck}$ as per the protocol description. In this case, the simulator is trivial, namely, the simulator consists in simply seeking an instance π_j^n such that the record transcript of that instance contains the transcript of $\pi_i^m.\tau$, i.e, the same tuple of nonces, key-exchange elements, and a verifying client finished message. Output the key ck sent to the middlebox by that instance as the key of π_i^m. We also note that if the client finished message does not verify, then the middlebox has to generate its own Finished message; if the adversary does that, we can construct a reduction from this game to the SACCE-security of P' (i.e, the standard TLS 1.2 handshake run between the client and the uncorrupted server), in which the adversary (possibly a collusion of all the malicious middlebox) simulates all but parties $\mathsf{P}_i, \mathsf{P}_j$ and will win by outputting the same instance and random sampling bit as the underlying adversary. Thus, we lose here another term $\mathsf{Adv}_P^{\mathsf{SACCE}}(\mathcal{A}_1)$.

□

Theorem 35. [**CSound-security of** KeylessTLS1.2**.**] *Let Π be our 3-(S)ACCE variant of Keyless SSL,* KeylessTLS1.2*, let P be the unilaterally-authenticated TLS 1.2 handshake, P', the mutually-authenticated TLS 1.2 handhsake between middlebox and servers, and Ψ, the transformation of P' to an AKE protocol by the computation of the export key ek. Consider a (t,q)-adversary $\mathcal{A}$ against the* CSound*-security of the protocol Π running at most t queries and creating at most q party instances per party. We denote by $\mathsf{n_P}$ the number of parties in the system, and denote $\mathcal{A}$'s advantage by $\mathsf{Adv}_{\Pi}^{\mathsf{CSound}}(\mathcal{A})$. If such an adversary exists, then there exist adversaries $\mathcal{A}_1$ against the* SACCE *security of P, and $\mathcal{A}_2$ against the* ACCE *security of P', each adversary running in time $t' \sim O(t)$ and instantiating at most $q' = q$ instances per party, such that*

$$\mathsf{Adv}_{\Pi}^{\mathsf{CSound}}(\mathcal{A}) \quad \leq \quad \frac{1}{2^{|\omega|}} + \mathsf{Adv}_P^{\mathsf{SACCE}}(\mathcal{A}_1) + {\mathsf{n_P}}^2 \cdot \mathsf{Adv}_{P'}^{\mathsf{ACCE}}(\mathcal{A}_2).$$

Proof. Our proof has the following hops:
Game 0: This game works as the CSound-game stipulated section 5.2.3. An adversary $\mathcal{A}$ wins this game on an arbitrarily fixed context ω an 3-(S)ACCE protocol if there exists a client-instance $\pi_i^m \in \mathsf{P}_i.\mathsf{Instances}$ for $\mathsf{P}_i \in \mathcal{C}$ and $\pi_i^m.\alpha = 1$, and there exists no partner-server-instance π_u^l such that $\pi_i^m.\mathsf{pid} = \mathsf{P}_u.\mathsf{name}$, $\mathsf{P}_u \in \mathbb{S}$ and $\pi_i^m.\mathsf{sid} = \pi_u^l.\mathsf{sid}$, and the following conditions simultaneously hold:

- $\omega \in \pi_i^m.\mathsf{rec}\tau$.
- It holds that any party P_x such that $\mathsf{RegParty}(\mathsf{P}_x, \Omega, \cdot)$ was queried, with $\omega \in \Omega$, P_x is uncorrupted.
- Furthermore, it holds that no party P_y such that $\mathsf{RegParty}(\mathsf{P}_y, \Omega,' Server')$ was queried, with $\omega \in \Omega$, is corrupted.

For our security proof, we will split this game in two games $\mathbb{G}_0^{\{1\}}$ and $\mathbb{G}_0^{\{2\}}$. The first game $\mathbb{G}_0^{\{1\}}$ considers the case where the partner identifier of the instance π_i^m is a server while the second one considers the case where this partner identifier is a middlebox. Thus we have:

$$\mathbf{Pr}[\mathcal{A}_{\mathbb{G}_1} \text{ wins}] = \mathbf{Pr}[\mathcal{A}_{\mathbb{G}_0^{\{1\}}} \text{ wins}] + \mathbf{Pr}[\mathcal{A}_{\mathbb{G}_0^{\{2\}}} \text{ wins}]$$

Game $\mathbb{G}_0^{\{1\}}$: We focus on the situation in which the instance π_i^m ends in an accepting state with partner identifier $\mathsf{P}_k \in \mathbb{S}$.
Game $\mathbb{G}_1^{\{1\}}$: This game behaves as the game $\mathbb{G}_0^{\{1\}}$ with the restriction that the partner identifier is a real server instance (which must be the owner of ω if π_i^m finished in an accepting state). Since

owners of ω cannot be corrupted, it follows that, except for the probability of server-impersonation towards a client, the true partner is indeed the indicated server.

$$|\mathbf{Pr}[\mathcal{A}_{\mathbb{G}_0^{\{1\}}} \text{ wins}] - \mathbf{Pr}[\mathcal{A}_{\mathbb{G}_1^{\{1\}}} \text{ wins}]| \leq \mathsf{Adv}_P^{\mathsf{SACCE}}(\mathcal{A}_1)$$

Winning game $\mathbb{G}_1^{\{1\}}$: We note that at this point, every time a client instance finishes in an accepting state (a pre-requisite of having the content in its record transcript), it holds that there exists a matching session π_u^s such that P_u is the owner of ω and $\pi_i^m.\mathsf{sid} = \pi_u^s.\mathsf{sid}$, contradicting the hypothesis of a winning attack. Thus,

$$\mathbf{Pr}[\mathcal{A}_{\mathbb{G}_1} \text{ wins}] = 0$$

Game $\mathbb{G}_0^{\{2\}}$: We now focus on the situation in which the instance π_i^m ends in an accepting state with partner identifier $\mathsf{P}_k \in \mathbb{MB}$. We note that all owners of ω are uncorrupted, and so are all middlebox parties that legitimately registered for that content.

Game $\mathbb{G}_1^{\{2\}}$: A middlebox party may obtain contents in two steps. First, the middlebox registers for that content; second, it requests this content on the secure channel of a middlebox-server session. In this game, we rule out the possibility that there exists a middlebox party P_x such that no $\mathsf{RegParty}(\mathsf{P}_x, \Omega, \cdot)$ was made, with $\omega \in \Omega$ was made, which successfully obtains the content ω via a secure channel with a server P_y, which is an owner of that content.

Indeed, this is bounded by the ACCE security of the established channel – first by entity authentication, then by channel security. The reduction follows the usual route, though the adversary needs to guess the identity of the content owner and that of the impersonated middlebox. This yields:

$$\mathsf{Pr}[\mathcal{A}_{\mathbb{G}_1^{\{2\}}} \text{ wins}] - \mathsf{Pr}[\mathcal{A}_{\mathbb{G}_2^{\{2\}}} \text{ wins}] \leq {\mathsf{n_P}}^2 \cdot \mathsf{Adv}_{P'}^{\mathsf{ACCE}}(\mathcal{A}_1)$$

Winning game $\mathbb{G}_1^{\{2\}}$: At this point, the adversary has no possibility of delivering the content other than guessing it, which yields a final term of $\frac{1}{2^{|\omega|}}$. □

5.4.3 Efficiency vs. Security

Our KeylessTLS1.2 proposal has several disadvantages in terms of efficiency with respect to Keyless SSL. First, the server is involved more heavily in client-middlebox handshakes. In fact, in KeylessTLS1.2 the middlebox is practically reduced to relaying the handshake between the client and the server. Most of our modifications are meant to achieve accountability. In particular, we must both *allow* the server to compute the master secret and channel keys, and at the same time *prevent* the middlebox from calculating those values itself (otherwise the msk can be used for session resumption). In addition to computing msk, ck, the server must also verify the client Finished message, and generate and encrypt $\mathsf{Fin}_{\mathcal{S}}$ on behalf of the middlebox.

Second, we disallow session resumption. An alternative is to relax the accountability definition requiring servers to only be able to compute the msk for each client-middlebox session (and not directly the channel key ck). Using TLS 1.2 session resumption is compatible with this relaxation; however, we cannot *recommend* using it, since it weakens security against malicious middleboxes. We also discourage the use of TLS-RSA in Keyless SSL, as this mode is inherently not forward secure; although TLS-DHE involves more exchanges with the server, it offers a better protection against malicious middlebox and corruptions.

Third, we need to compute the export key ek to blind the channel-key transmission. In the proofs, this allows us to simulate the record-layer transcript to an adversary, even when the reduction does not know ck. Although not computationally-heavy, this step does deviate from the standard TLS 1.2 handshake.

Fourth, we also require a very large certificate infrastructure, with one public key/certificate per middlebox, per content. A way to reduce that number is to ensure, on the server side, that no content can be delivered by more than one middlebox. This could, however, significantly decrease the efficiency of the CDN.

Our design represents one set of trade-offs that favors security over efficiency. Other designs may choose a different balance. For example, if we want to prevent just the attacks in Section 5.3.2 without attaining full 3-(S)ACCE-security, smaller changes are probably adequate.

Chapter 6

Conclusion & Further Perspectives

This thesis is focused on the secure channel establishment between client and server when a third intermediate entity acts within the communications. This establishment is based on a correct execution of an authenticated key exchange protocol, when some session keys are agreed between endpoints of the channel. Our researches are focused on two specific contexts: the mobile networks where a secure channel has to be established across the access network and the secure HTTP communication (HTTPS) where a content delivery to a local entity is required.

In this chapter, we recall the main obtained results for each contexts and how we fulfil our objectives. We also propose some additional works to do, some others possible leads in each context.

Mobile Networks. This study starts by a complete overview of the radio access mobile networks. Proposed by the 3rd Generation Partnership Project (3GPP) as a standard for 3G and 4G mobile-network communications, the UMTS-AKA and EPS-AKA protocols are meant to provide a mutually-authenticated key-exchange between clients and associated serving networks, denoted *servers*, to establish a secure channel between clients and local servers, across the radio access network.

We have formulated requirements with respect to both Man-in-the-Middle adversaries, i.e. key-indistinguishability and impersonation security, and to local untrusted serving networks namely state-confidentiality and soundness. Additionally to these security requirements, a paramount requirement is that of client privacy, which 3GPP defines in terms of: user identity confidentiality, service untraceability, and location untraceability. Moreover, since servers are sometimes untrusted (in the case of roaming), the AKA protocols must also protect clients with respect to these third parties.

We have proved that the unmodified UMTS-AKA protocol attains the security properties as long as servers cannot be corrupted. Considering client-tracking attacks e.g. by using error messages or IMSI catchers, some variants of the UMTS-AKA have been proposed, addressing such problems. We use the approach of provable security to show that these variants, as UMTS-AKA protocol, still fail to guarantee the privacy of mobile clients.

Additionally to the study of UMTS-AKA protocol, we have proposed a specific analysis of the EPS-AKA protocol to know if the 4G variant of AKA protocol can guarantee more security and client privacy than the UMTS-AKA protocol. In this way, we consider the three following statements: (a) The EPS-AKA protocol guarantees the key-indistinguishability property considering the ability to corrupt a server, which is not the case of the UMTS-AKA protocol; (b) as the GSM and UMTS version of the AKA protocol, the EPS-AKA cannot guarantee any kind of user privacy despite the use of a new temporary identifier; (c) we propose a secure modular variant of the EPS-AKA protocol providing all the security and privacy requirements and considering some practical considerations and the cost reductions due to the pre-established EPS architecture.

We have proposed a first improvement of AKA protocols, which retains most of its structure and respects practical necessities such as key-management, but which provably attains security and user privacy with respect to servers and Man-in-the-Middle adversaries. One of the main features of this improvement is the delete of the resynchronization procedure. Such a remove can imply some DoS attack based on the desynchronization of clients/operators' states. We consider that an additional formal study in Tamarin to study the desynchronization's ability, is indispensable before a possible standardization.

Finally, we provide additional practical solution for a possible future standardization and implementation within 5G mobile networks, considering additional lawful and practical issues. Thus, we have filled two patents, a first one focus on the user identification taking into consideration lawful interceptions and a second one focusing on the impact of subscriber key leakage.

Content Delivery Networks. Due to revelations of widespread mass-surveillance by governmental agencies, natural migration appeared from HTTP to HTTPS, to delivery web services and exchange information across a secure channel. Content delivery networks were introduced as a large networks of reverse catching proxies to accelerate HTTP delivery. The secure channel established in HTTPS communications, is based on the TLS protocol. This protocol is a 2-party protocol executed between web client and web server. With the act of intermediate proxies, the TLS protocol cannot be directly implied. The basic option is to have different TLS connection between each pair of consecutive *neighbours* along the chain of links in the communication. But with such an option, proxies can modify and access to the exchanged data and it is not clear that the security of the several independent TLS handshakes imply the establishment of a secure channel between endpoints.

A first variant proposed by Cloudfare, is called Keyless SSL [101]. We provide a first full security overview of the Keyless SSL proposal in the content delivery networks. Considering the different security requirements from the CDNs context, we also propose a 3-(S)ACCE security model. Such a security model, we have studied the security of Keyless SSL pointing out different practical and security weaknesses. Thus, we propose an enhanced design of Keyless SSL guaranteeing the different required security requirements which include TLS 1.2 architecture.

Alternatively to Keyless SSL as an alternative design, multi-context TLS (mcTLS) was introduced in 2015 [91], as mentioned in Chapter 6. Where a middle entity, e.g. a proxy, can mitigate and/or relay TLS handshake, and record communication between end-users and servers for a variety of purposes like content filtering, compression, etc. This alternative has no complete security analysis. We consider as important to analyse the strengths and shortcomings of mcTLS protocol checking the AKE and ACCE security considering the specific third intermediate entity as an important part of the security model.

Bibliography

[1] 3GPP. 3G security; technical specification group (tsg) sa; 3G security; security architecture. Tech. Rep. 33.102, 3rd Generation Partnership Project (3GPP), 1999.

[2] 3GPP. 3G security; specification of the MILENAGE algorithm set: An example algorithm set for the 3GPP authentication and key generation functions f1, f1*, f2, f3, f4, f5 and f5*; document 2: Algorithm specification. Tech. Rep. 35.206, 3rd Generation Partnership Project (3GPP), June 2007.

[3] 3GPP. Technical specification group core network and terminals; sim/usim internal and external interworking aspects. Tech. Rep. 31.900, 3rd Generation Partnership Project (3GPP), 2012.

[4] 3GPP. 3G security; specification of the TUAK algorithm set: A 2nd example for the 3GPP authentication and key generation functions f1, f1*, f2, f3, f4, f5 and f5* – document 1: Algorithm specification. Tech. Rep. 35.231, 3rd Generation Partnership Project (3GPP), June 2013.

[5] 3GPP. 3G security; technical specification group services and system aspects; 3gpp network domain security (nds); ip network layer security. Tech. Rep. 33.210, 3rd Generation Partnership Project (3GPP), 2015.

[6] 3GPP. 3G security; generic authentication architecture (gaa); generic bootstrapping architecture (gba). Tech. Rep. 33.220, 3rd Generation Partnership Project (3GPP), 2016.

[7] 3GPP. 3G security; numbering, addressing and identification. Tech. Rep. 23.003, 3rd Generation Partnership Project (3GPP), 2016.

[8] 3GPP. 3G security; technical specification group services and system aspects; 3gpp system architecture evolution(sae) sa; security architecture. Tech. Rep. 33.401, 3rd Generation Partnership Project (3GPP), June 2016.

[9] 3GPP. 3G security; technical specification group services and system aspects; specification of the 3gpp confientiality and integrity algorithms. Tech. Rep. 35.201, 3rd Generation Partnership Project (3GPP), 2016.

[10] 3GPP. 3G security; technical specification group services and system aspects; specification of the 3gpp confientiality and integrity algorithms: Kasumi specification. Tech. Rep. 35.202, 3rd Generation Partnership Project (3GPP), 2016.

[11] 3GPP. 3G security; technical specification group services and system aspects; specification of the 3gpp confientiality and integrity algorithms uea1 and uia2: Document 1: Uea2 and uia2 specifications. Tech. Rep. 35.215, 3rd Generation Partnership Project (3GPP), 2016.

[12] 3GPP. 3G security; technical specification group services and system aspects; specification of the 3gpp confientiality and integrity algorithms uea1 and uia2: Document 2: Snow 3g specifications. Tech. Rep. 35.216, 3rd Generation Partnership Project (3GPP), 2016.

[13] 3GPP. 3G security; technical specification group (tsg) sa; 3G security; security architecture. Tech. Rep. 33.102, 3rd Generation Partnership Project (3GPP), June 2016.

[14] 3GPP. 4G security; general packet radio service (gprs) enhancements for evolved universal terrestrial radio access network (e-utran) access. Tech. Rep. 23.401, 3rd Generation Partnership Project (3GPP), 2016.

[15] 3GPP. 4G security; non-access-stratum (nas) protocol for evolved packet system (eps). Tech. Rep. 24.301, 3rd Generation Partnership Project (3GPP), 2016.

[16] 3GPP. Evolved universal terrestrial radio access (e-utra) and evolved universal terrestrial radio access network (e-utran); overall description; stage 2. Tech. Rep. 36.300, 3rd Generation Partnership Project (3GPP), 2016.

[17] 3GPP. Network architecture. Tech. Rep. 23.002, 3rd Generation Partnership Project (3GPP), 2016.

[18] 3GPP. Network domain security (nds); authentication framework (af). Tech. Rep. 33.310, 3rd Generation Partnership Project (3GPP), 2016.

[19] 3GPP. Radio interface protocol architecture. Tech. Rep. 25.301, 3rd Generation Partnership Project (3GPP), 2016.

[20] 3GPP. Security related network functions. Tech. Rep. 43.020, 3rd Generation Partnership Project (3GPP), 2016.

[21] 3GPP. Service requirements for the evolved packet system (eps). Tech. Rep. 22.278, 3rd Generation Partnership Project (3GPP), 2016.

[22] 3GPP. Specification of the 3gpp confidentiality and integrity algorithms eea3 and eia3; document 1: Eea3 and eia3 specifications. Tech. Rep. 35.221, 3rd Generation Partnership Project (3GPP), 2016.

[23] 3GPP. Study on the security aspects of the next generation system. Tech. Rep. 33.899, 3rd Generation Partnership Project (3GPP), May 2016.

[24] ADRIAN, D., BHARGAVAN, K., DURUMERIC, Z., GAUDRY, P., GREEN, M., HALDERMAN, J., HENINGER, N., SPRINGALL, D., THOMÉ, E., VANDERSLOOT, B., WUSTROW, E., BÉGUELIN, S. Z., AND ZIMMERMANN, P. Imperfect forward secrecy: How Diffie-Hellman fails in practice. In *Proceedings of ACM CCS 2015* (2015), IEEE, pp. 5–17.

[25] ALFARDAN, N., BERNSTEIN, D., PATERSON, K., POETTERING, B., AND SCHULDT, J. On the security of RC4 in TLS and WPA. In *USENIX Security Symposium* (2013).

[26] ALFARDAN, N., AND PATERSON, K. Plaintext-recovery attacks against datagram TLS. In *Network and Distributed System Security Symposium (NDSS'12)* (2012).

[27] ALFARDAN, N., AND PATERSON, K. Lucky thirteen: Breaking the TLS and DTLS record protocols. In *IEEE Symposium on Security and Privacy (SP'13)* (2013).

[28] ANTOINE DELIGNAT-LAVAUD, AND BHARGAVAN, K. Network-based origin confusion attacks against HTTPS virtual hosting. In *Proceedings of WWW'15* (2015), Springer, pp. 227–237.

[29] ARAPINIS, M., MANCINI, L. I., RITTER, E., AND RYAN, M. Privacy through pseudonymity in mobile telephony systems. In *21st Annual Network and Distributed System Security Symposium, NDSS* (2014).

[30] ARAPINIS, M., MANCINI, L. I., RITTER, E., RYAN, M., GOLDE, N., REDON, K., AND BORGAONKAR, R. New privacy issues in mobile telephony: fix and verification. In *ACM Conference on Computer and Communications Security* (2012), T. Yu, G. Danezis, and V. D. Gligor, Eds., ACM, pp. 205–216.

[31] ARAPINIS, M., MANCINI, L. I., RITTER, E., RYAN, M., GOLDE, N., REDON, K., AND BORGAONKAR, R. New privacy issues in mobile telephony: fix and verification. In *the ACM Conference on Computer and Communications Security, CCS'12, Raleigh, NC, USA, October 16-18, 2012* (2012), pp. 205–216.

[32] ARAPINIS, M., RITTER, E., AND RYAN, M. D. Statverif: Verification of stateful processes. In *Proceedings of the 24th IEEE Computer Security Foundations Symposium, CSF 2011* (2011), pp. 33–47. http://dx.doi.org/10.1109/CSF.2011.10.

[33] ATENIESE, G., HERZBERG, A., KRAWCZYK, H., AND TSUDIK, G. Untraceable mobility or how to travel *incognito*. In *Elsevier Computer Networks* (1999), vol. 31, Elsevier, pp. 871–884.

[34] AVIRAM, N., SCHINZEL, S., SOMOROVSKY, J., HENINGER, N., DANKEL, M., STEUBE, J., VALENTA, L., ADRIAN, D., J. ALEX HALDERMAN, DUKHOVNI, V., KÄSPER, E., COHNEY, S., ENGELS, S., PAAR, C., AND SHAVITT, Y. Drown: Breaking TLS using SSLv2, 2016. https://drownattack.com.

[35] BAR-ON, A. Improved higher-order differential attacks on MISTY1. In *Fast Software Encryption - 22nd International Workshop, FSE 2015, Istanbul, Turkey, March 8-11, 2015, Revised Selected Papers* (2015), pp. 28–47. http://dx.doi.org/10.1007/978-3-662-48116-5_2.

[36] BARD, G. V. Vulnerability of SSL to chosen-plaintext attack. Cryptology ePrint Archive: Report 2004/111, May 2004. http://eprint.iacr.org/2004/111.

[37] BELLARE, M., KILIAN, J., AND ROGAWAY, P. The security of the cipher block chaining message authentication code. *J. Comput. Syst. Sci. 61*, 3 (2000), 362–399.

[38] BELLARE, M., AND NAMPREMPRE, C. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In *Advances in Cryptology - ASIACRYPT 2000, 6th International Conference on the Theory and Application of Cryptology and Information Security, Kyoto, Japan, December 3-7, 2000, Proceedings* (2000), pp. 531–545. http://dx.doi.org/10.1007/3-540-44448-3_41.

[39] BELLARE, M., POINTCHEVAL, D., AND ROGAWAY, P. Authenticated key exchange secure against dictionary attacks. In *Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Proceeding* (2000), pp. 139–155. http://dx.doi.org/10.1007/3-540-45539-6_11.

[40] BELLARE, M., AND ROGAWAY, P. Entity authentication and key distribution. In *Advances in Cryptology - CRYPTO '93, 13th Annual International Cryptology, Proceedings* (1993), pp. 232–249. http://dx.doi.org/10.1007/3-540-48329-2_21.

[41] BELLARE, M., AND ROGAWAY, P. Entity authentication and key distribution. In *CRYPTO* (1993).

[42] BELLARE, M., AND ROGAWAY, P. Provably secure session key distribution: the three party case. In *Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing* (1995), pp. 57–66. http://doi.acm.org/10.1145/225058.225084.

[43] BERDOUCHE, B., BHARGAVAN, K., ANTOINE DELIGNAT-LAVAUD, FOURNET, C., KOHLWEISS, M., PIRONTI, A., PIERRE YVES STRUB, AND JEAN KARIM ZINZINDOHOUE. A messy state of the union: Taming the composite state machines of TLS. In *Proceedings of IEEE S&P 2015* (2015), IEEE.

[44] BERTONI, G., DAEMEN, J., PEETERS, M., AND ASSCHE, G. V. Keccak specification. In *NIST Specifications.* (2008). http://keccak.noekeon.org/.

[45] BERTONI, G., DAEMEN, J., PEETERS, M., AND ASSCHE, G. V. On the indifferentiability of the sponge construction. In *Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques.* (2008), pp. 181–197.

[46] BEURDOUCHE, B., BHARGAVAN, K., DELIGNAT-LAVAUD, A., FOURNET, C., KOHLWEISS, M., PIRONTI, A., STRUB, P.-Y., AND ZINZINDOHOUE, J. A messy state of the union: Taming the composite state machines of TLS. In *Proceedings of IEEE S&P 2015* (2015).

[47] BHARGAVAN, K., DELIGNAT-LAVAUD, A., FOURNET, C., PIRONTI, A., AND STRUB, P.-Y. Triple handshakes and cookie cutters: Breaking and fixing authentication over TLS. In *Proceedings of IEEE S&P 2014* (2014).

[48] BHARGAVAN, K., FOURNET, C., KOHLWEISS, M., PIRONTI, A., AND STRUB, P. Implementing TLS with verified cryptopgrahic security. In *Proceedings of IEEE S&P 2013* (2013).

[49] BHARGAVAN, K., FOURNET, C., KOHLWEISS, M., PIRONTI, A., STRUB, P., AND BÉGUELIN, S. Z. Proving the TLS handshake secure (as it is). In *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Proceedings* (2014), pp. 235–255. http://dx.doi.org/10.1007/978-3-662-44381-1_14.

[50] BHARGAVAN, K., AND LEURENT, G. Transcript collision attacks: Breaking authentication in TLS, IKE, and SSH. In *Accepted at NDSS 2016, to appear* (2016).

[51] BIHAM, E., DUNKELMAN, O., AND KELLER, N. A related-key rectangle attack on the full KASUMI. In *Advances in Cryptology - ASIACRYPT 2005, 11th International Conference on the Theory and Application of Cryptology and Information Security, Proceedings* (2005), pp. 443–461. http://dx.doi.org/10.1007/11593447_24.

[52] BLANCHET, B. An efficient cryptographic protocol verifier based on prolog rules. In *14th IEEE Computer Security Foundations Workshop CSFW-14* (2001), pp. 82–96. http://dx.doi.org/10.1109/CSFW.2001.930138.

[53] BLANCHET, B. Automatic verification of security protocols in the symbolic model: The verifier proverif. In *Foundations of Security Analysis and Design VII - FOSAD 2012/2013 Tutorial Lectures* (2013), pp. 54–87. http://dx.doi.org/10.1007/978-3-319-10082-1_3.

[54] BRZUSKA, C., FISCHLIN, M., SMART, N., WARINSCHI, B., AND WILLIAMS, S. Less is more: Relaxed yet composable security notions for key exchange. *International Journal of Information Security 12(4)* (2013).

[55] BRZUSKA, C., KON JACOBSEN, H., AND STEBILA, D. Safely exporting keys from secure channels: on the security of EAP-TLS and TLS key exporters. In *EuroCrypt* (2016).

[56] CANETTI, R., AND KRAWCZYK, H. Universally composable notions of key exchange and secure channels. In *Advances in Cryptology - EUROCRYPT 2002, International Conference on the Theory and Applications of Cryptographic Techniques, Proceedings* (2002), pp. 337–351. http://dx.doi.org/10.1007/3-540-46035-7_22.

[57] CANETTI, R., AND KRAWCZYK, H. Universally composable notions of key exchange and secure channels. In *Proceedings of EUROCRYPT'02* (2002), vol. 2332 of *LNCS*.

[58] CANVEL, B., HILTGEN, A., VAUDENAY, S., AND VUAGNOUX, M. Password interception in a SSL/TLS channel. In *Proceedings of CRYPTO 2003* (2003), vol. 2729 of *LNCS*.

[59] CREMERS, C. J. F. Key exchange in ipsec revisited: Formal analysis of ikev1 and ikev2. In *Computer Security - ESORICS 2011 - 16th European Symposium on Research in Computer Security, Leuven, Belgium, September 12-14, 2011. Proceedings* (2011), pp. 315–334. http://dx.doi.org/10.1007/978-3-642-23822-2_18.

[60] DIERKS, T., AND RESCORLA, E. The transport layer security (TLS) protocol version 1.2. RFC 5246, August 2008.

[61] DOWLING, B., FISCHLIN, M., GÜNTHER, F., AND STEBILA, D. A cryptographic analysis of the TLS 1.3 handshake protocol candidates. In *ACM CCS* (2015), pp. 1197–1210.

[62] D.STROBEL. IMSI catcher. In *2007, Seminar Work, Ruhr-Universitat Bochum* (2007).

[63] DUNKELMAN, O., KELLER, N., AND SHAMIR, A. A practical-time related-key attack on the KASUMI cryptosystem used in GSM and 3g telephony. In *Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Proceedings* (2010), pp. 393–410. http://dx.doi.org/10.1007/978-3-642-14623-7_21.

[64] DWORKIN, M. Recommendation for Block Cipher Modes of Operation : Methods and Techniques. NIST Specifications 800-38a.

[65] FIPS.180.4. Secure hash standard (SHS). Federal Information Processin Standards Publication 180.4, 2015. http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf.

[66] FIPS.197. Specification for the advanced encryption standard (AES). Federal Information Processin Standards Publication 197, 2001. http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf.

[67] GEORGIEV, M., IYENGAR, S., JANA, S., ANUBHAI, R., BONEH, D., AND SHMATIKOV, V. The most dangerous code in the world: Validating SSL certificates in non-browser software. In *Proceedings of ACM CCS'12* (2012).

[68] GERO, C., SHAPIRO, J., AND BURD, D. Terminating ssl connections without locally-accessible private keys, June 20 2013. WO Patent App. PCT/US2012/070,075.

[69] GIESEN, F., KOHLAR, F., AND STEBILA, D. On the security of TLS renegotiation. Cryptology ePrint Archive Report 2012/630, 2013.

[70] GILBERT, H. The security of one-block-to-many modes of operation. In *Fast Software Encryption, 10th International Workshop, FSE* (2003), pp. 376–395.

[71] HE, C., SUNDARARAJAN, M., DATTA, A., DEREK, A., AND MITCHELL, J. A modular correctness proof of IEEE 802.11i and TLS. In *Proceedings of ACM CCS'05* (2005).

[72] HERMANS, J., PASHALIDIS, A., VERCAUTEREN, F., AND PRENEEL, B. A new RFID privacy model. In *Computer Security - ESORICS 2011 - 16th European Symposium on Research in Computer Security, Leuven, Belgium, September 12-14, 2011. Proceedings* (2011), pp. 568–587.

[73] JAGER, T., KOHLAR, F., SCHAGE, S., AND RG SCHWENK, J. On the security of TLS-DHE in the standard model. In *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Proceedings* (2012), pp. 273–293. http://dx.doi.org/10.1007/978-3-642-32009-5_17.

[74] JAGER, T., KOHLAR, F., SCHÄGE, S., AND SCHWENK, J. On the security of TLS-DHE in the standard model. In *Proceedings of CRYPTO 2012* (2012), vol. 7417 of *LNCS*.

[75] JONSSON, J., AND KALISKI JR., B. On the security of RSA encryption in TLS. In *Proceedings of CRYPTO 2002* (2002), vol. 2442 of *LNCS*.

[76] KAUFFMAN, C. Internet Key Exchange (IKEv2) Protocol, RFC 4306. In *RFC Editor* (2005). https://tools.ietf.org/html/rfc4306.

[77] KENT, S. IP Encapsulating Security Payload (ESP), RFC 4303. In *RFC Editor* (2005). https://tools.ietf.org/html/rfc4303.

[78] KOHLWEISS, M., MAURER, U., ONETE, C., RN TACKMANN, B., AND VENTURI, D. (de-)constructing TLS. *IACR Cryptology ePrint Archive 2014* (2014), 20. http://eprint.iacr.org/2014/020.

[79] KRAWCZYK, H. The order of encryption and authentication for protecting communication (or: How secure is SSL?). In *Proceedings of CRYPTO 2001* (2001), vol. 2139 of *LNCS*.

[80] KRAWCZYK, H., BELLARE, M., AND CANETTI, R. HMAC: Keyed-Hashing for Message Authentication, RFC 2104, updated by RFC 6151. In *RFC Editor* (1997). http://www.ietf.org/rfc/rfc2104.txt.

[81] KRAWCZYK, H., PATERSON, K. G., AND WEE, H. On the security of the TLS protocol: A systematic analysis. In *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Proceedings* (2013), pp. 429–448. http://dx.doi.org/10.1007/978-3-642-40041-4_24.

[82] KRAWCZYK, H., AND WEE, H. The OPTLS protocol and tls 1.3. In *Proceedings of Euro S&P* (2016), IEEE. https://eprint.iacr.org/2015/978.

[83] LEVILLAIN, O., GOURDIN, B., AND DEBAR, H. TLS record protocol: Security analysis and defense-in-depth countermeasures. In *Proceedings of ACM ASIACCS 2015* (2015), ACM.

[84] MAURER, U., AND TACKMANN, B. On the soundness of Authenticate-then-Encrypt: Formalizing the malleability of symmetric encryption. In *Proceedings on ACM CCS'10* (2010), ACM.

[85] MAURER, U., TACKMANN, B., AND CORETTI, S. Key exchange with unilateral authentication: Composable security definition and modular protocol design. Cryptology ePrint Archive, Report 2013/555, 2013. http://eprint.iacr.org/.

[86] MCGREW, D. A., AND VIEGA, J. The security and performance of the galois/counter mode (GCM) of operation. In *Progress in Cryptology - INDOCRYPT 2004, 5th International Conference on Cryptology in India* (2004), pp. 343–355. http://dx.doi.org/10.1007/978-3-540-30556-9_27.

[87] MING-FENG LEE AND NIGEL P. SMART AND BOGDAN WARINSCHI AND GAVEN J. WATSON. Anonymity guarantees of the UMTS/LTE authentication and connection protocol. *Int. J. Inf. Sec. 13*, 6 (2014), 513–527.

[88] MJØLSNES, S. F., AND TSAY, J. Computational security analysis of the UMTS and LTE authentication and key agreement protocols. *CoRR abs/1203.3866* (2012). http://arxiv.org/abs/1203.3866.

[89] MORRISSEY, P., SMART, N., AND WARINSCHI, B. A modular security analysis of the TLS handshake protocol. In *Proceedings of ASIACRYPT 2008* (2008), vol. 5350 of *LNCS*.

[90] MORRISSEY, P., SMART, N. P., AND WARINSCHI, B. The TLS handshake protocol: A modular analysis. *J. Cryptology 23*, 2 (2010), 187–223. http://dx.doi.org/10.1007/s00145-009-9052-3.

[91] NAYLOR, D., SCHOMP, K., VARVELLO, M., LEONTIADIS, I., BLACKBURN, J., DIEGO R. LÓPEZ, PAPAGIANNAKI, K., PABLO RODRIGUEZ RODRIGUEZ, AND STEENKISTE, P. Multi-Context TLS (mcTLS): Enabling Secure In-Network Functionality in TLS. In *Proceedings of SIGCOMM 2015* (2015), ACM.

[92] NYBERG, K., AND NIEMI, V. UMTS security, 1st edition. In *Wiley* (2003).

[93] PAISE, R.-I., AND VAUDENAY, S. Mutual authentication in RFID: Security and privacy. In *Proc. on the 3rd ACM Symposium on Information, Computer and Communications Security (ASIACCS)* (2008), ACM, pp. 292–299.

[94] PATERSON, K., RISTENPART, T., AND SHRIMPTON, T. Tag size does matter: Attacks and proofs for the TLS record protocol. In *Advances in Cryptology — ASIACRYPT 2011* (2011), vol. 7073 of *LNCS*, Springer-Verlag.

[95] PFITZMANN, A., AND HANSEN, M. Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management – a consolidated proposal for terminology. https://dud.inf.tu-dresden.de/literatur/Anon_Terminology_v0.31.pdf.

[96] RHOLZ, U. D., FISCHLIN, M., KASPER, M., AND ONETE, C. A formal approach to distance bounding RFID protocols. In *Proceedings of the 14th Information Security Conference ISC 2011* (2011), vol. 7001 of *LNCS*, Springer, pp. 47–62.

[97] SHAFIUL, M., KHAN, A., AND MITCHELL, C. J. Another look at privacy threats in 3G mobile telephony.

[98] SHAIK, A., BORGAONKAR, R., ASOKAN, N., NIEMI, V., AND SEIFERT, J.-P. Practical attacks against privacy and availability in 4g/lte mobile communication systems.

[99] SHOUP, V. On formal models for secure key exchange. *IACR Cryptology ePrint Archive 1999* (1999), 12.

[100] SHOUP, V. A proposal for an iso standard for public key encryption. Cryptology ePrint Archive, Report 2001/112, 2001. http://eprint.iacr.org/.

[101] STEBILA, D., AND SULLIVAN, N. An analysis of TLS handshake proxying. In *Trustcom/BigDataSE/ISPA, 2015 IEEE* (Aug 2015).

[102] TSAY, J., AND MJØLSNES, S. F. A vulnerability in the UMTS and LTE authentication and key agreement protocols. In *Computer Network Security - 6th International Conference on Mathematical Methods, Models and Architectures for Computer Network Security, MMM-ACNS. Proceedings* (2012). http://dx.doi.org/10.1007/978-3-642-33704-8_6.

[103] ULRIKE MEYER AND SUSANNE WETZEL. A man-in-the-middle attack on UMTS. In *Proceedings of the 2004 ACM Workshop on Wireless Security, Philadelphia, PA, USA, October 1, 2004* (2004), pp. 90–97.

[104] VAN DEN BROEK, F., VERDULT, R., AND DE RUITER, J. Defeating IMSI catchers. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, USA, October 12-6, 2015* (2015), pp. 340–351.

[105] VAUDENAY, S. Security flaws induced by CBC padding – applications to SSL, IPSEC, WTLS. In *Proceedings of EUROCRYPT 2002* (2002), vol. 2332 of *LNCS*.

[106] VAUDENAY, S. On privacy models for RFID. In *ASIACRYPT '07* (2007), vol. 4883, pp. 68–87.

[107] WAGNER, D., AND SCHNEIER, B. Analysis of the SSL 3.0 protocol. In *USENIX Workshop on Electronic Commerce* (1996).

[108] ZAHRA AHMADIAN AND SOMAYEH SALIMI AND AHMAD SALAHI. New attacks on UMTS network access. In *2009 Wireless Telecommunications Symposium, WTS 2009, Prague, Czech Republic, April 22-24, 2009* (2009), pp. 1–6.

[109] ZHANG, M. Provably-secure enhancement on 3GPP authentication and key agreement protocol. *IACR Cryptology ePrint Archive 2003* (2003), 92. http://eprint.iacr.org/2003/092.

Appendices

Appendix A

Relevant Security Definitions

We recall in this chapter the different security definitions that have been used in our different security analysis detailed in the different chapters.

A.1 Primitives

A.1.1 Generic functions: pseudo-random functions and permutations.

A pseudo-random function (PRF) is a family of functions with the property that the input-output behavior of a random instance of the family is computationally indistinguishable from that of a random function. Consider a function $f : \{0,1\}^k \times \{0,1\}^m \to \{0,1\}^n$. This property is defined in terms of the following security game $\mathbb{G}^{\mathsf{prf}}$:

Security Game $\mathbb{G}^{\mathsf{prf}}$:
$K \xleftarrow{\$} \{0,1\}^k$;
$b \in \{0,1\}$;
If $b = 0$ then assigns $\mathsf{F_0}$ to a random function $\mathsf{Rand} : \{0,1\}^m \to \{0,1\}^n$;
If $b = 1$ then assigns $\mathsf{F_1}$ to the function $f(K, .)$;
$b' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{F_b}}(.)}$.
$\mathcal{A}$ wins iff $b = b'$.

Definition 20. [Pseudo-Random Function.] *A function f from $\{0,1\}^k \times \{0,1\}^m$ to $\{0,1\}^n$ is (t,q)-PRF-secure if any probabilistic polynomial time adversary $\mathcal{A}$ running in time t and making at most q queries to the PRF oracle, cannot distinguish f from a random function* Rand *with a non-negligible advantage. We can evaluate the PRF-advantage of an adversary against f, denoted by* $\mathsf{Adv}_f^{\mathsf{prf}}(\mathcal{A})$ *as follows, for a random function denoted by* $\mathsf{Rand} : \{0,1\}^m \to \{0,1\}^n$*:*

$$\mathsf{Adv}_f^{\mathsf{prf}}(\mathcal{A}) = \big|\Pr[\mathcal{A} \text{ wins } \mathbb{G}^{\mathsf{prf}} | b \leftarrow 1] - \Pr[\mathcal{A} \text{ wins } \mathbb{G}^{\mathsf{prf}} | b \leftarrow 0]\big|.$$

A.1.2 Secure Symmetric Encryption Algorithms

An encryption scheme $\mathcal{SE}$ is a tuple of probabilistic polynomial-time algorithms $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ and sets $\mathcal{K}, \mathcal{M}, \mathcal{C}$ such that:

- The **random-key-generator algorithm** KeyGen takes in input a security parameter l and outputs a key $K \leftarrow \mathsf{KeyGen}(1^l)$, with $K \in \mathcal{K}$ and $|K| \geq l$;

- The **encryption algorithm** Enc maps a key K and a plaintext $M \in \mathcal{M} = \{0,1\}^m$ to a ciphertext $C \leftarrow \mathsf{Enc}_K(M)$;
- The **decryption algorithm** Dec maps a key K and a ciphertext $C \in \mathcal{C} = \{0,1\}^n (n \geq m)$ to either a plaintext $M \leftarrow \mathsf{Dec}_K(C)$ or a special symbol $\perp$;

The encryption scheme is *perfectly correct* if, and only if, for all $l, K \leftarrow \mathsf{KeyGen}(1^l)$ and $M \in \mathcal{M} = \{0,1\}^m$: $M = \mathsf{Dec}_K(\mathsf{Enc}_K(M))$.

The security of encryption schemes is given in terms of a left-or-right indistinguishability notion which ensures that the adversary does not learn even one bit of a plaintext from the ciphertext. The most common definitions used to capture the security of symmetric encryption schemes are:

- Indistinguishability under Chosen Plaintext Attack (IND-CPA), in which the adversary is limited to just having access to the Encryption oracle to win its game.
- Indistinguishability under adaptive Chosen Ciphertext Attack (IND-CCA2) , in which the adversary is additionally given access to a Decryption oracle.

For a symmetric encryption scheme $\mathcal{SE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$, the indistinguishability under adaptive chosen ciphertext attack (IND-CCA2) is defined in terms of the following security game:

Security Game $\mathbb{G}^{\mathsf{ind-cca2}}$:
$K \leftarrow \mathsf{KeyGen}(1^l)$;
$b \in \{0,1\}$;
$(m_0, m_1) \leftarrow \mathcal{A}^{\mathsf{Enc}_K(.),\mathsf{Dec}_K(.)}$, with $|m_0| = |m_1|$;
$C \leftarrow \mathsf{Enc}_K(m_b)$;
$(b' \leftarrow \mathcal{A}^{\mathsf{Enc}_K(.),\mathsf{Dec}^*_K(.)},$ with $\mathsf{Dec}^*_K(.)$ works just like $\mathsf{Dec}^*_K(.)$, except $\mathsf{Dec}_K(C) = \perp$.
$\mathcal{A}$ wins iff $b = b'$.

An encryption scheme is indistinguishable under an adaptive chosen ciphertext attack if every probabilistic polynomial time adversary has only a negligible advantage over random guessing, i.e. it wins the below game with a probability $1/2 + \epsilon(l)$, where $\epsilon(l)$ is a negligible function in the security parameter l.

Definition 21. [General security of IND-CCA2-scheme.] *A scheme $\mathcal{SE}$ is considered as IND-CCA2-secure if any probabilistic polynomial time adversary $\mathcal{A}$ running in time t, making at most q_{enc} queries to the encryption oracle, and respectively at most q_{dec} queries to the decryption oracle, given an encryption of a message randomly chosen from a two-element message space, cannot distinguish efficiently the encryption of one of both messages. We can evaluate the IND-CCA2-advantage of a such adversary, denoted $\mathsf{Adv}^{\mathsf{ind-cca2}}_{\mathcal{SE}}(\mathcal{A})$:*

$$\mathsf{Adv}^{\mathsf{ind-cca2}}_{\mathcal{SE}}(\mathcal{A}) = \big|\Pr[\mathcal{A} \text{ wins } \mathbb{G}^{\mathsf{ind-cca2}}] - \frac{1}{2}\big|.$$

We note that the security game of the indistinguishability under chosen plaintext attack (IND-CPA) behaves similarly as the previous one, except that the adversary has only access to the encryption oracle and cannot used the results of the encryption to select subsequent plaintext after the test.

A.1.3 MAC using pseudo-random function

Let F be a pseudo-random function. A message authentication code scheme is a tuple of probabilistic polynomial-time algorithms $\mathcal{SM} = (\mathsf{KeyGen}, \mathsf{Mac}, \mathsf{Vrf})$ and sets $\mathcal{K}, \mathcal{M}, \mathcal{T}$ such that:

- The **random-key-generator algorithm** KeyGen receives a security parameter l and outputs a key $K \rightarrow \mathsf{KeyGen}(1^l)$, with $K \in \mathcal{K}$ and $|K| \geq l$;
- The **message authentication code algorithm** Mac maps a key K and a message $M \in \mathcal{M} = \{0,1\}^m$ to a tag value $T \leftarrow F_K(M) \in \{0,1\}^n$;
- The **verification algorithm** Vrf maps a key K, a message $M \in \mathcal{M} = \{0,1\}^m$, and a received tag value $T \in \mathcal{T} = \{0,1\}^n$ to check the equality $T = F_K(M)$.

The security of MAC schemes $\mathcal{SM}$ is defined in terms of the existential unforgeability of tags against chosen message attacks (EUF-CMA) if no probabilistic polynomial time adversary can forge a tag of any freshly-chosen message. It is formalized in terms of the following security game (denoted $\mathbb{G}^{\mathsf{euf-cma}}$):

Security Game $\mathbb{G}^{\mathsf{euf-cma}}$:
$K \leftarrow \mathsf{KeyGen}(1^l)$;
$(M,T) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Mac}}(\cdot)}$;
$\mathcal{A}$ wins iff $\mathsf{Vrf}_K(M,T) = 1$ and M was not queried to the Mac oracle.

A message authentication code scheme $\mathcal{SM}$ is secure in terms of unforgeability if every probabilistic polynomial time adversary has only a negligible advantage to output a correct mac value, i.e. a couple (T, M) such that $\mathsf{Vrf}_K(M,T) = 1$ which has not been output by the Mac Oracle.

Definition 22. [Security of MAC-scheme.] *A message authentication code scheme* $\mathcal{SM} = (\mathsf{KeyGen}, \mathsf{Mac}, \mathsf{Vrf})$ *is considered as existentially unforgeable against chosen message attacks if any probabilistic polynomial time adversary* $\mathcal{A}$ *running in time* t *and making at most* q *queries to its Mac oracle, cannot forge with a non-negligible success probability, a couple* (M, Mac) *with* M *which had not been previously input by the Mac oracle. We can evaluate the EUF-CMA-advantage of a such adversary, denoted* $\mathsf{Adv}^{\mathsf{mac}}_{\mathcal{SM}}(\mathcal{A})$*:*

$$\mathsf{Adv}^{\mathsf{mac}}_{\mathcal{SM}}(\mathcal{A}) = \mathsf{Adv}^{\mathsf{euf-cma}}_{SSM}(\mathcal{A}) = \Pr[\mathcal{A} \text{ wins } \mathbb{G}^{\mathsf{euf-cma}}].$$

As mentioned in [37], pseudorandom functions make good message authentication codes. The authors have determined the exact security of a such reduction by the following proposal, which has been useful for our AKA results:

Proposition 1. *Let* $f : \{0,1\}^k * \{0,1\}^m \rightarrow \{0,1\}^n$ *be a family of functions. Consider a* (t,q)*-adversary* $\mathcal{A}$ *against the prf-security of the function* f*, running in time* t *and making at most* q *queries to its challenger. Denote the advantage of a such adversary* $\mathsf{Adv}^{\mathsf{prf}}_f(\mathcal{A})$*. then, there are a* $(t' \sim t + O(s+d), q' = q)$*-adversary* $\mathcal{A}$*' against the mac-security of the MAC scheme* $\mathcal{SM} = (\mathsf{KeyGen}, f, \mathsf{Vrf})$ *with an advantage* $\mathsf{Adv}^{\mathsf{mac}}_{\mathcal{SM}}(\mathcal{A}')$ *such as:*

$$\mathsf{Adv}^{\mathsf{mac}}_{\mathcal{SM}}(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{prf}}_f(\mathcal{A}') + \frac{1}{2^n}.$$

A.1.4 Key derivation functions

A classical key derivation function is a basic and essential cryptographic function permitting to obtain some new pseudorandom keys from a source of initial keying material. A KDF is secure if the keys it outputs are *indistinguishable* from random output of the same length. In particular, the output keys reveal nothing about the input keying material. Consider a key derivation function $\mathsf{KDF} : \{0,1\}^k \times \{0,1\}^m \rightarrow \{0,1\}^n$.

The security of key derivation function is defined as a particular case of the PRF security game described above.

Definition 23. [Security of a KDF-scheme.] *A key derivation function f from $\{0,1\}^k \times \{0,1\}^m$ to $\{0,1\}^n$ is (t,q)-KDF-secure if any probabilistic polynomial time adversary $\mathcal{A}$ running in time t and making at most q queries to the KDF oracle, cannot distinguish f from a random function* Rand *with a non-negligible advantage. We can evaluate the KDF-advantage of an adversary against f, denoted by $\mathsf{Adv}_f^{\mathsf{kdf}}(\mathcal{A})$ as follows, for a random function denoted by $\mathsf{Rand} : \{0,1\}^m \to \{0,1\}^n$:*

$$\mathsf{Adv}_f^{\mathsf{kdf}}(\mathcal{A}) = \big|\Pr[\mathcal{A} \text{ wins } \mathbb{G}^{\mathsf{kdf}} | b \leftarrow 1] - \Pr[\mathcal{A} \text{ wins } \mathbb{G}^{\mathsf{kdf}} | b \leftarrow 0]\big|.$$

A.1.5 Authenticated Encryption functions

An authenticated encryption scheme is a tuple of probabilistic polynomial-time algorithms $\mathcal{SAE} = (\mathsf{KeyGen}, \mathsf{AE.Enc}, \mathsf{AE.Dec})$ and sets $\mathcal{K}, \mathcal{A}, \mathcal{M}, \mathcal{C}$ such that:

- The **random-key-generator algorithm** KeyGen takes in input a security parameter l and outputs a key $K \leftarrow \mathsf{KeyGen}(1^l)$, with $K \in \mathcal{K}$ and $|K| \geq l$;
- The **encryption algorithm** AE.Enc maps the key K, the output length n, additional data A, and a message $M \in \mathcal{M} = \{0,1\}^m$ to a ciphertext $C = \mathsf{AE.Enc}_K(M, A, n) \in \mathcal{C} = \{0,1\}^n$.
- The **decryption algorithm** AE.Dec maps the key K, additional data A and a ciphertext $C \in \mathcal{C} = \{0,1\}^n (n \geq m)$ to either a plaintext $M \leftarrow \mathsf{AE.Dec}_K(C, A)$ or a special symbol $\perp$;

The main problem around the authenticated encryption is to establish a generic construction permitting to guarantee confidentiality and integrity. Bellare and Namprempre detailed in [38] the security of the three main construction of authenticated encryption.

We consider the following definition of the security of the AEAD scheme. An AEAD-scheme $\mathcal{SAE} = (\mathsf{KeyGen}, \mathsf{AE.Enc}, \mathsf{AE.Dec})$ is AE-secure if every probabilistic polynomial time adversary has only a negligible advantage of the Left-or-Right security game using presented [94] and detailed as follows:

Security game $\mathbb{G}^{\mathsf{ae}}$:	Oracle $\mathcal{O}_{\mathsf{Enc}}(.)$:	Oracle $\mathcal{O}_{\mathsf{Dec}}(.)$:
$K \leftarrow \mathsf{KeyGen}(1^l)$	$C_0 \leftarrow \mathsf{AE.Enc}_K(M_0, A, n)$.	If $b = 1$ & $C \notin$ List then
$b \leftarrow \{0,1\}$	$C_1 \leftarrow \mathsf{AE.Enc}_K(M_1, A, n)$.	return $AE.Dec_K(C, A)$.
$b' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Enc}}(.), \mathcal{O}_{\mathsf{Dec}}(.)}$	If $C_0 = \perp$ or $C_1 = \perp$ then	return $\perp$.
$\mathcal{A}$ wins iff $b = b'$.	then return $\perp$	
	List $\xleftarrow{U} C_b$	
	return C_b.	

Definition 24. [Security of AE-scheme.] *An AEAD-scheme Π is a tuple of algorithms $\mathcal{SAE} = (\mathsf{KeyGen}, \mathsf{AE.Enc}, \mathsf{AE.Dec})$ as detailed above. We consider that such a scheme is* ae*-secure if any probabilistic-polynomial-time adversary $\mathcal{A}$ running in time t and making at most q queries to the encryption and decryption oracles, cannot success with a non-negligible advantage. This advantage, denoted $\mathsf{Adv}^{\mathsf{ae}}_{\mathcal{SAE}}(\mathcal{A})$, is defined as follows:*

$$\mathsf{Adv}^{\mathsf{ae}}_{\mathcal{SAE}}(\mathcal{A}) = \big|\Pr[\mathcal{A} \text{ wins } \mathbb{G}^{\mathsf{ae}} | b \leftarrow 1] - \Pr[\mathcal{A} \text{ wins } \mathbb{G}^{\mathsf{ae}} | b \leftarrow 0]\big|.$$

A.2 ACCE- and AKE-security models

In the following, we will recall how the classic ACCE (authenticated and confidential channel establishment) and AKE (authenticated key exchange) security properties have been defined respectively in [73] and [40]. These properties are applicable to two-party protocols, run between

a *client* and a *server*. In order to distinguish these notions from our own security models, as defined in Chapters 3 and 5, we denote traditional AKE and ACCE security as 2-AKE and 2-ACCE security, respectively.

Informally, in the AKE security definitions, an authenticated key-exchange protocol is considered secure if it outputs pseudorandom keys. If, in addition, the primitives used for authenticated encryption provide suitable confidentiality, authenticity, and integrity properties, compositional results such as [54] guarantee that a secure channel is obtained as a result. However, in the case of the TLS protocol, the AKE security notion is not guaranteed, since the Finished messages act as trivial real-from-random distinguishers for the session keys. In 2012, Jager et al. [74] proposed a relaxation of AKE security to a monolithic definition, which captures two demands : entity authentication and channel security. Thus, ACCE security essentially guarantees only that the keys output by the AKE protocol are sufficient to establish a secure channel; by contrast, AKE security guarantees that the keys themselves are good, thus offering better composability guarantees.

Parties and Instances. Both models consider a set $\mathcal{P}$ of parties, which is partitioned in two: the set $\mathcal{C}$ of *clients* and the set $\mathbb{S}$ of *servers*. Each party $\mathsf{P}_i \in \mathcal{P}$ is associated with a long-term secret key sk and, possibly with a long-term public key pk encapsulated in a certificate.

Parties can behave honestly, or can be adversarially controlled. One party is associated with several *instances*, each corresponding to a single execution of the protocol or, in other words, a protocol-session. For a party $\mathsf{P}_i \in \mathcal{P}$, we denote the m-th instance of that party as π_i^m. Each such instance π_i^m is described by the following attributes and their values:

- The **long-term, secret** key $\pi_i^m.\mathsf{sk} := \mathsf{sk}_i$ of party P_i and the **long-term, public** key $\pi_i^m.\mathsf{pk} := \mathsf{pk}_i$ of P_i, with $\mathsf{sk}_i, \mathsf{pk}_i \in \{0,1\}^*$. For clients, in the case that the TLS handshake is unilaterally authenticated, the public key $\pi_i^m.\mathsf{pk}$ and secret key $\pi_i^m.\mathsf{sk}$ are set to the special symbol $\perp$, which simply means that the party-instance does not hold any effective values for these attributes.

- The **role** of P_i in the session, $\pi_i^m.\rho \in \{init, resp\}$, denoting that the party can be the *initiator* or the *responder* in the session.

- The **session identifier**, $\pi_i^m.\mathsf{sid} \in \{0,1\}^* \cup \{\perp\}$; the session identifier is $\perp$ when the session does not exist.

- The **partner identifier**, $\pi_i^m.\mathsf{pid} \in \mathcal{P} \cup \{\text{“}Client''\}$. In particular, the partner ID indicates the alleged interlocutor of an user, as this user ascertains this partner during its running session $\pi_i^m.\mathsf{pid}$. In the case of TLS with unilateral authentication, servers cannot authenticate their partners, so in that case we say that the corresponding partner identifier is set to a generic string “Client”.

- The **acceptance-flag** $\pi_i^m.\alpha \in \{1, 0, \perp\}$; the value $\perp$ denotes that the session is still ongoing and the acceptance-flag has not been set. If for some instance π_i^m it holds that $\pi_i^m.\alpha = 1$, then we say the instance has terminated in an accepting state.

- The **established channel-key**, $\pi_i^m.\mathsf{ck} \in \{0,1\}^* \cup \{\perp\}$; the key $\pi_i^m.\mathsf{ck}$ is initially set to a special symbol $\perp$, but if the instance terminates in an accepting state, them $\pi_i^m.\mathsf{ck}$ is set to a concrete bitstring.

- The **revealed-bit** $\pi_i^m.\gamma \in \{0,1\}$; a left-or-right bit of value 1 denotes that the channel key was revealed to the adversary, otherwise the bit has value 0.

- The **sampling-bit** $\pi_i^m.\mathsf{b} \in \{0,1\}$; a bit sampled uniformly at the beginning of the session for purposes of security-games.

- The **transcript** $\pi_i^m.\tau$ of the instance; this is essentially formed of the random coins of this instance, its public knowledge prior to the session, as well as the ordered suite of messages received and sent by this party-instance from the beginning of the handshake, until it sets its acceptance bit to either 1 or 0.

- The **authentication flag** $\pi_i^m.\mathsf{auth}$ of the instance is a bit, initialized to 0 or 1 depending on whether mutual or unilateral (server-authentication) authentication is required in the studied protocol. We say $\pi_i^m.\mathsf{auth} = 0$ if unilateral server-authentication is required (as in the 2-SACCE protocols) or $\pi_i^m.\mathsf{auth} = 1$ if the mutual authentication is required (as in the 2-ACCE and AKE protocols). We note that this flag is not used in the traditional security model but will be useful in our analysis.

The execution environment in the two-party setting is formed of a polynomial number (in the security parameter) of party-instances or sessions being executed concurrently.

To define security, we need to define –as it is customary– some extra properties over the notion of sessions. Two sessions $\pi_i^m.\mathsf{pid}$ and $\pi_j^n.\mathsf{pid}$ are *partners* if $\pi_i^m.\mathsf{sid} = \pi_j^n.\mathsf{sid}$ and these session identifiers are not $\perp$. For a fixed session $\pi_i^m.\mathsf{pid}$, if its partner does not exist as per the above, then we say that $\pi_i^m.\mathsf{pid}$'s is $\perp$.

The AKE Threat model. We recall the *AKE security experiment* with the related notations used to consider the 2-ACCE security. Adversarial parties can control the network, up to an interaction with the aforementioned party-instances, also called session-oracles, described by the following queries:

1. $\mathsf{NewSession}(\mathsf{P}_i, \rho, pid)$: This query creates a new session π_i^m executed by party P_i with the role ρ, having the partner identifier pid. Moreover, the session is set to be in the running state and the associated party is its initiator, i.e., $\pi_i^m.\alpha = \perp$ and $\pi_i^m.\rho = init$. If the considered protocol provides mutual authentication, the authentication flag of all the instances π_i^m is instantiated to $\pi_i^m.\mathsf{auth} = 1$. If the protocol provides only unilateral authentication, the authentication flag of all the instances is instantiated to $\pi_i^m.\mathsf{auth} = 0$.

2. $\mathsf{Send}(\pi_i^m, n)$: This query allows the adversary to send any message n within the session π_i^m. If the session is in the running state, then the session-oracle π_i^m will answer to n as per the protocol, otherwise it will return a fixed error message, $\perp$.

3. $\mathsf{Corrupt}(\mathsf{P}_i)$: Via this query P_i's secret key is returned to the adversary.

4. $\mathsf{Reveal}(\pi_i^m)$: This query returns the channel key $\pi_i^m.\mathsf{ck}$ and the revealed bit of the session, $\pi_i^m.\gamma$, is set to 1.

5. $\mathsf{Test}(\pi_i^m)$: If $\pi_i^m.\alpha \neq 1$, then this query returns $\perp$. Otherwise, the oracle of this query samples k_0 uniformly from the domain of channel keys and set k_1 to be $\pi_i^m.\mathsf{ck}$. Of the two keys k_0 and k_1, then returns $k_{\pi_i^m.\mathsf{b}}$. This key is called the *test-challenge*.

AKE security game. The AKE-security is now defined by requiring that (i) the protocol is a secure mutual authentication (EA-security) based on their *matching conversation* notion (ii) the key-indistinguishability (K-IND-security) of the exchanged session key, i.e. any adversary cannot in polynomial time the established session key to random value. A 2-party protocol is said to be a *correct AKE protocol*, if the following holds:

$$\pi_i^m.\alpha = 1 \Rightarrow \pi_i^m.\mathsf{ck} \neq \perp \text{ and } \pi_i^m.\mathsf{sid} \neq \perp.$$
$$\pi_i^m.\alpha = \pi_j^n.\alpha = 1, \pi_i^m.\mathsf{auth} = \pi_j^n.\mathsf{auth} \text{ and}$$

$$\pi_i^m.\mathsf{sid} = \pi_j^n.\mathsf{sid} \Rightarrow \begin{cases} \pi_i^m.\mathsf{ck} = \pi_j^n.\mathsf{ck} \\ \pi_i^m.\mathsf{pid} = \mathsf{P}_j; \pi_j^n.\mathsf{pid} = \mathsf{P}_i. \end{cases}$$

In relation to the adversarial queries (1-4) above and especially so with the $\mathsf{Corrupt}(\cdot)$ query, considered in the context of the state of a session, the following notion appears. A session $\pi_i^m.\mathsf{pid}$ is said to be *fresh*, with intended partner P_j, if:

1. $\pi_i^m.\mathsf{pid} = \mathsf{P}_j$ when the adversary $\mathcal{A}$ issued its first query;
2. $\pi_i^m.\gamma = 0$ and P_i is uncorrupted;
3. for any π_j^n which is partner with π_i^m, we have that $\pi_j^n.\gamma = 0$ and P_j is uncorrupted.

A session $\pi_i^m.\mathsf{pid}$ is said to be *accepted maliciously in the AKE security* experiment with intended partner P_j, if:

1. $\pi_i^m.\alpha = 1$ and $\pi_i^m.\mathsf{pid} = \mathsf{P}_j$ when the adversary $\mathcal{A}$, playing in the AKE experiment, issued its first query;
2. P_i and P_j are not corrupted;
3. there is no unique session π_j^s such that π_j^s is the partner of π_i^m.

We use $\mathsf{Adv}_{\Pi}^{\mathsf{EA}}(\mathcal{A})$ to denote the probability that an adversary $\mathcal{A}$, (playing the AKE security game and trying to break the entity authentication property) makes a session accept maliciously on the protocol Π.

An adversary $\mathcal{A}$, that queries $\mathsf{Test}(\pi_i^m.\mathsf{pid})$ during the AKE security experiment, answers the test-challenge correctly if it outputs a bit b', such that $\pi_i^m.\mathsf{pid}$ is fresh with some intended partner P_j and $\pi_i^m.\mathsf{b} = b'$. We use $\mathsf{Adv}_{\Pi}^{\mathsf{K-ind}}(\mathcal{A})$ to denote the advantage $\mathcal{A}$ (trying to break the key-indistinguishability property) in outputting the correct b' in the above game, notably how much better $\mathcal{A}$'s winning probability is compared to a basic guessing probability.

An adversary $\mathcal{A}$ wins the AKE security experiment if he makes a session to accept maliciously or he answers the test-challenge correctly. We use $\mathsf{Adv}_{\Pi}^{\mathsf{AKE}}(\mathcal{A})$ to denote his advantage in winning the AKE security experiment, which is the probability that he does make a session accept maliciously, plus the absolute probability-difference between a non-trivial output of the test-challenge and guessing the test-challenge at random, i.e., $\mathsf{Adv}_{\Pi}^{\mathsf{AKE}}(\mathcal{A})$ is the sum of $\mathsf{Adv}_{\Pi}^{\mathsf{EA}}(\mathcal{A})$ and $\mathsf{Adv}_{\Pi}^{\mathsf{K-ind}}(\mathcal{A})$ for any $\mathcal{A}$ in the AKE security experiment.

The 2-ACCE Threat model. We define the 2-ACCE model including the SACCE model which a variant of the ACCE model when only the server is authenticated. The traditional 2-party ACCE security will be noted 2-ACCE to be clearly. These models include both 2-ACCE model considering protocol which requires mutual authentication and 2-SACCE model considering protocol which requires unilateral server authentication.

The 2-ACCE experiment is described via the following game. A challenger plays against an adversary with access (during the game and outside the game) the subset of queries formed with the first four types of queries (1-4) plus one additional query (available just during the game):

5. $\mathsf{Encrypt}(\pi_i^m, l, n_0, n_1, H)$: This query takes as input a ciphertext-length l, two messages n_0, n_1, and a header H. The query uses the sLHAE encryption stE correctly such that in an accepting session π_i^m, i.e., where $\pi_i^m.\alpha = 1$, it will encrypt under $\pi_i^m.\mathsf{ck}$ the message n_b where $b := \pi_i^m.\mathsf{b}$. The details of this are given in Figure A.1.

6. $\mathsf{Decrypt}(\pi_i^m, C, H)$: This query takes as input a ciphertext C and a header H. The query uses the sLHAE encryption stE correctly such that in an accepting session π_i^m, i.e., where $\pi_i^m.\alpha = 1$, it will decrypt under $\pi_i^m.\mathsf{ck}$ the ciphertext C. The details of this are given in Figure A.1.

$\mathsf{Encrypt}(\pi_i^s, l, m_0, m_1, H)$:

$C, H \leftarrow$ empty list;
$u, v \leftarrow 0$;
If$(\pi_i^s.\alpha \neq 1)$ then Return $\perp$;
$u \leftarrow u + 1$;
$(C^{(0)}, st^{(0)}{}_E) \leftarrow stE.Enc(k, \pi_i^s, m_0, H, st_E)$;
$(C^{(1)}, st^{(1)}{}_E) \leftarrow stE.Enc(k, \pi_i^s, m_1, H, st_E)$;
If($(C^{(0)} = \perp)$ Or $(C^{(1)} = \perp)$) then Return $\perp$;
$(C[u], H[u], st_E) \leftarrow (C^{(b)}, H, st^{(b)})$;
Return $C[u]$;

$\mathsf{Decrypt}(\pi_i^s, C, H)$:

$C, H \leftarrow$ empty list; $u, v \leftarrow 0$; $\mathtt{in\text{-}sync} = \mathsf{True}$;
If$(\pi_i^s.\alpha \neq 1)$ then Return $\perp$;
If$(\pi_i^s.\mathsf{b} = 0)$ then Return $\perp$;
$\pi_i^t \leftarrow$ partner of π_i^s;
$v \leftarrow v + 1$;
$(m, st_D) \leftarrow stE.Dec(k, C, H, st_D)$;
If($(v > \pi_i^t.u) \mid (C \neq \pi_i^t.C[v]) \mid (H \neq \pi_i^t.H[v])$) then $\mathtt{in\text{-}sync} \leftarrow$ False;
If $(\pi_i^t = \perp)$ then $\mathtt{in\text{-}sync} \leftarrow$ False;
If ($\mathtt{in\text{-}sync} =$ False) then Return m;
Return $\perp$

Figure A.1: With the respect to the algorithms above, we consider that the instance $\pi_i^s.\alpha$ has $k, b, st_E, st_D, C, H, u$ and v as variables local to its internal state.

Authenticated Encryption for 2-ACCE. As per [74], we now henceforth assume, suitably for TLS 1.2., that the established channel key is used in *stateful Length-Hiding Authenticated Encryption (sLHAE)* schemes, specified as the tuple of algorithms $stE = (st.Gen, stE.Init, stE.Enc, stE.Dec)$. Clearly, we also assume that the established channel key is used to encrypt/decrypt messages with stE (via $stE.Enc$ and $stE.Dec$) using key(s) generated by $st.Gen$ in such a way that it respects the correct sequences of states produced by stE and commenced via $stE.Init$. The security of such scheme is approximately the same as an indistinguishability property and is detailed in [73] as follows:

2-(S)ACCE security game. The *2-(S)ACCE security experiment* is described just as the AKE security experiment except for the fact that the adversary $\mathcal{A}$ has access to the query-types 1-6. and no Test query. All the other, related notions (e.g., session freshness and correctness) are defined in the (S)ACCE case as in the AKE case. In the ACCE security experiment we use $\mathsf{Adv}_{\Pi}^{2-\mathsf{ACCE}}(\mathcal{A})$ to denote his advantage in winning the 2-ACCE security experiment, which is the probability that he does make a session accept maliciously, plus the absolute probability-difference between a non-trivial strategy of outputting the sampling bit of a fresh session and that of guessing its sampling bit, i.e, $\mathsf{Adv}_{\Pi}^{2-\mathsf{ACCE}}(\mathcal{A})$ is the sum of $\mathsf{Adv}_{\Pi}^{\mathsf{SC}}(\mathcal{A})$ and $\mathsf{Adv}_{\Pi}^{\mathsf{EA}}(\mathcal{A})$ for any $\mathcal{A}$ in the 2-ACCE security experiment. We note that the 2-SACCE security game is a similar game as the 2-ACCE security game but considering only the unilateral authentication.

2-(S)ACCE Entity Authentication (EA). In the EA game, the adversary queries the first four oracles above and its goal is to make one instance, π_i^m of an uncorrupted P_i *accept maliciously*. That is, π_i^m must end in an accepting state, with partner ID P_j, also uncorrupted, such that no other unique instance of P_j partnering π_i^m exists. The adversary's advantage in this game is its winning probability.

2-(S)ACCE Security of the Channel (SC). In the SC-game, the adversary $\mathcal{A}$ can use all the oracles except Test and must output, for a fresh instance π_i^m, the bit $\pi_i^m.\mathsf{b}$ of that instance. The adversary's advantage is the absolute difference between its winning probability and $\frac{1}{2}$.

Résumé

Dans cette thèse, nous nous sommes intéressés à la sécurité des protocoles d'authentification et de dérivations de clefs dans le cas où une troisième entité intermédiaire, partiellement de confiance, est requise pour différentes raisons pratiques.

Dans un premier temps, nous nous sommes focalisés sur le protocole AKA, dont les différentes versions sont utilisées pour établir un canal sécurisé sur la voix radio au sein des réseaux mobiles 3G et 4G. Nous avons d'abord fait état des faiblesses de sécurité et celles concernant le respect de la vie privée des clients mobiles durant l'établissement d'un tel canal sécurisé. Différentes solutions pratiques ont été proposé afin d'assurer les propriétés de sécurité et de vie privée requises par le 3GPP au sein des réseaux 3G, 4G. Dans un second temps, nous avons analysé le protocole Keyless SSL utilisé au sein des CDNs afin d'établir le canal sécurisé requis pour les communications HTTPS. Nous avons proposé un modèle de sécurité calculatoire recoupant l'ensemble des besoins de sécurité et ainsi pointé les différentes faiblesses de sécurité de la proposition Keyless SSL. Par conséquent, une variante basée sur TLS 1.2 a été proposé.

Mots-clés: Sécurité Prouvable, Protocoles AKA, Réseaux Mobiles, Keyless SSL, CDN, TLS.

Abstract

In this thesis, we study the security of authentication and key exchange protocols when they are proxied through a semi-trusted third party is required.

We begin by focusing on the security of the UMTS/LTE AKA protocol, when the different versions of this protocol are used to establish a secure channel across a radio access link in 3G and 4G mobile networks. We first describe some security and privacy weaknesses during the execution of the EPS- and UMTS-AKA protocols. Then, several practical solutions are proposed, guaranteeing better security and privacy for this protocol in both 3G and 4G scenarios.

Secondly, we focus on computer networks, more precisely on the use of the Keyless SSL in proxying over HTTPS. A security model including the different various, specific security requirements from the web delivery context has been established. We also identify and discuss various weaknesses in the structure of Keyless SSL. Finally, we propose an improvement of Keyless SSL over TLS 1.2, and describe how Keyless SSL could work securely for the new TLS 1.3 protocol version.

Keywords: Provable Security, AKA Protocols, Mobile Networks, Keyless SSL, CDN, TLS.