Securing a trusted hardware environment (Trusted Execution Environment)

Mathieu da Silva

THÈSE POUR OBTENIR LE GRADE DE DOCTEUR DE L'UNIVERSITÉ DE MONTPELLIER

En SYAM – Systèmes Automatiques et Micro-Electroniques

École doctorale : I2S – Information, Structures et Systèmes

Unité de recherche : LIRMM – Laboratoire d'Informatique, de Robotique et de Micro-électronique de Montpellier

SECURISATION D'UN ENVIRONNEMENT MATERIEL DE CONFIANCE (TRUSTED EXECUTION ENVIRONMENT)

SECURING A TRUSTED HARDWARE ENVIRONMENT (TRUSTED EXECTION ENVIRONMENT)

Présentée par Mathieu Da Silva

Le 26 novembre 2018

Sous la direction de Bruno Rouzeyre, Giorgio Di Natale et Marie-Lise Flottes

Devant le jury composé de

Bruno Rouzeyre,	Professeur à l'Université de Montpellier, LIRMM, Montpellier	Directeur de thèse
Giorgio Di Natale	Directeur de recherche au CNRS, LIRMM, Montpellier	Co-encadrant
Marie-Lise Flottes	Chargé de recherche au CNRS, LIRMM, Montpellier	Co-encadrant
Lilian Bossuet	Professeur à l'Université Jean Monnet, Laboratoire Hubert Curien, Saint-Etienne	Rapporteur
Lorena Anghel	Professeur à Grenoble INP, TIMA, Grenoble	Rapporteur
Alberto Bosio	Professeur à l'ECL, INL, Lyon	Président
David Hély	Associate Professor à Grenoble INP, LCIS, Grenoble	Membre
Sophie Dupuis	Maître de conférence à l'Université de Montpellier, LIRMM, Montpellier	Membre

UNIVERSITÉ DE MONTPELLIER

ABSTRACT

This work is part of the Trusted Environment Execution eVAluation (TEEVA) project (French project FUI n°20 from January 2016 to December 2018) that aims to evaluate two alternative solutions for secure mobile platforms: a purely software one, the Whitebox Crypto, and a TEE solution, which integrates software and hardware components. The TEE relies on the ARM TrustZone technology available on many of the chipsets for the Android smartphones and tablets market. This thesis focuses on the TEE architecture. The goal is to analyze potential threats linked to the test/debug infrastructures classically embedded in hardware systems for functional conformity checking after manufacturing.

Testing is a mandatory step in the integrated circuit production because it ensures the required quality and reliability of the devices. Because of the extreme complexity of nowadays integrated circuits, test procedures cannot rely on a simple control of primary inputs with test patterns, then observation of produced test responses on primary outputs. Test facilities must be embedded in the hardware at design time, implementing the so-called Design-for-Testability (DfT) techniques. The most popular DfT technique is the scan design. Thanks to this test-driven synthesis, registers are connected in one or several chain(s), the so-called scan chain(s). A tester can then control and observe the internal states of the circuit through dedicated scan pins and components. Unfortunately, this test infrastructure can also be used to extract sensitive information stored or processed in the chip, data strongly correlated to a secret key for instance. A scan attack consists in retrieving the secret key of a crypto-processor thanks to the observation of partially encrypted results.

Experiments have been conducted during the project on the demonstrator board with the target TEE in order to analyze its security against a scan-based attack. In the demonstrator board, a countermeasure is implemented to ensure the security of the assets processed and saved in the TEE. The test accesses are disconnected preventing attacks exploiting test infrastructures but disabling the test interfaces for testing, diagnosis and debug purposes. The experimental results have shown that chips based on TrustZone technology need to implement a countermeasure to protect the data extracted from the scan chains. Besides the simple countermeasure consisting to avoid scan accesses, further countermeasures have been developed in the literature to ensure security while preserving test and debug facilities. State-of-the-art countermeasures against scan-based attacks have been analyzed. From this study, we investigate a new proposal in order to preserve the scan chain access while preventing attacks, and to provide a plug-and-play countermeasure that does not require any redesign of the scanned circuit while maintaining its testability. Our solution is based on the encryption of the test communication, it provides confidentiality of the communication between the circuit and the tester and prevents usage from unauthorized users. Several architectures have been investigated, this document also reports pros and cons of envisaged solutions in terms of security and performance.

RESUME

Ce travail de thèse a pour cadre le projet Trusted Environment Execution eVAluation (TEEVA) (projet français FUI n°20 de Janvier 2016 à Décembre 2018) qui vise à évaluer deux solutions alternatives de sécurisation des plateformes mobiles, l'une est purement logicielle, la Whitebox Crypto, alors que l'autre intègre des éléments logiciels et matériels, le Trusted Environment Execution (TEE). Le TEE s'appuie sur la technologie TrustZone d'ARM disponible sur de nombreux chipsets du marché tels que des smartphones et tablettes Android. Cette thèse se concentre sur l'architecture TEE, l'objectif étant d'analyser les menaces potentielles liées aux infrastructures de test/debug classiquement intégrées dans les circuits pour contrôler la conformité fonctionnelle après fabrication.

Le test est une étape indispensable dans la production d'un circuit intégré afin d'assurer fiabilité et qualité du produit final. En raison de l'extrême complexité des circuits intégrés actuels, les procédures de test ne peuvent pas reposer sur un simple contrôle des entrées primaires avec des patterns de test, puis sur l'observation des réponses de test produites sur les sorties primaires. Les infrastructures de test doivent être intégrées dans le matériel au moment du design, implémentant les techniques de Design-for-Testability (DfT). La technique DfT la plus commune est l'insertion de chaînes de scan. Les registres sont connectés en une ou plusieurs chaîne(s), appelé chaîne(s) de scan. Ainsi, un testeur peut contrôler et observer les états internes du circuit à travers les broches dédiées. Malheureusement, cette infrastructure de test peut aussi être utilisée pour extraire des informations sensibles stockées ou traitées dans le circuit, comme par exemple des données fortement corrélées à une clé secrète. Une attaque par scan consiste à récupérer la clé secrète d'un crypto-processeur grâce à l'observation de résultats partiellement encryptés.

Des expérimentations ont été conduites sur la carte électronique de démonstration avec le TEE afin d'analyser sa sécurité contre une attaque par scan. Dans la carte électronique de démonstration, une contremesure est implémentée afin de protéger les données sensibles traitées et sauvegardées dans le TEE. Les accès de test sont déconnectés, protégeant contre les attaques exploitant les infrastructures de test, au dépend des possibilités de test, diagnostic et debug après mise en service du circuit. Les résultats d'expérience ont montré que les circuits intégrés basés sur la technologie TrustZone ont besoin d'implanter une contremesure qui protège les données extraites des chaînes de scan. Outre cette simple contremesure consistant à éviter l'accès aux chaînes de scan, des contremesures plus avancées ont été développées dans la littérature pour assurer la sécurité tout en préservant l'accès au test et au debug. Nous avons analysé un état de l'art des contremesures contre les attaques par scan. De cette étude, nous avons proposé une nouvelle contremesure qui préserve l'accès aux chaînes de scan tout en les protégeant, qui s'intègre facilement dans un système, et qui ne nécessite aucun redesign du circuit après insertion des chaînes de scan tout en préservant la testabilité du circuit. Notre solution est basée sur l'encryption du canal de test, elle assure la confidentialité des communications entre le circuit et le testeur tout en empêchant son utilisation par des utilisateurs non autorisés. Plusieurs architectures ont été étudiées, ce document rapporte également les avantages et les inconvénients des solutions envisagées en termes de sécurité et de performance.

1. Introduction

Avec le nombre important d'applications dont sont chargées les smartphones, les opérations critiques, telles que les transactions bancaires ou la gestion des droits numériques (DRM), ont besoin d'être protégées. Une solution existante est d'utiliser un environnement de confiance, le Trusted Environment Execution (TEE), pour réaliser ces opérations confidentielles. Les applications communes sont quant à elles exécutées dans le système d'exploitation du smartphone, par exemple Android. L'objectif de cette thèse est d'évaluer l'architecture TEE face aux menaces liées à l'exploitation des infrastructures de test. Les infrastructures de test sont nécessaires afin d'assurer la qualité d'un produit après fabrication, ainsi que fournir un moyen de debug lorsque le circuit est opérationnel.

Le test/debug des circuits intégrés est possible grâce à des techniques de design dédiées, appelées Design-for-Testability (DfT). Les chaînes de scan sont la technique de DfT la plus commune. Cette méthode consiste à organiser les bascules du circuit en un/des registre(s) à décalage lorsque le circuit est en mode de test. Ainsi, les états internes du circuit sont contrôlables en utilisant l'entrée série de la chaîne de scan, appelée scan-in, et ils sont observables en utilisant la sortie série de la chaîne de scan, appelée scan-out. En plus des chaînes de scan, plusieurs standards ont été proposés pour faciliter le test des circuits électroniques. Le standard pour tester les cartes électroniques est l'IEEE 1149.1 [8], communément appelé JTAG. Le standard IEEE 1500 [9] a été proposé pour tester les System-on-Chip (SoC). Et le récent standard IEEE 1687 [10], aussi appelé IJTAG, facilite l'accès aux instruments de test embarqués, en définissant un réseau scan reconfigurable.

Ces techniques de conception en vue du test sont nécessaires pour fournir une observation et un contrôle total sur les états internes du système. Elles sont utilisées après la production du circuit intégré pour vérifier son bon fonctionnement. De plus lorsque le circuit est opérationnel, les infrastructures de test assurent le diagnostic en cas de dysfonctionnement du circuit intégré, ainsi qu'un moyen de debugger les applications installées sur le dispositif électronique. Cependant, accéder aux infrastructures de test peut compromettre la sécurité du système. En effet, un attaquant peut exploiter les moyens de test intégrés au circuit afin de voler des informations secrètes. Par exemple, sur des crypto-processeurs, l'observation du contenu des chaînes de scan permet de retrouver la clé secrète. Plusieurs attaques sur l'Advanced Encryption Standard (AES [12]) ont été proposées dans la littérature [13]–[16], appelées attaques scan. L'AES est composé de plusieurs rondes où des opérations de substitution et permutation sont appliquées sur un bloc de données en clair afin d'obtenir le bloc chiffré. Les attaques scan ciblent la première ronde de l'AES quand les données ne sont que partiellement chiffrées. La procédure de l'attaque consiste à passer le circuit en mode test après l'exécution de la première ronde afin de récupérer le contenu du registre de ronde en accédant aux chaînes de scan. L'attaquant procède ensuite à une attaque différentielle en appliquant des paires de texte en clair, puis en calculant la distance d'Hamming entre les deux résultats. Dans certains cas, le résultat du calcul de la distance de Hamming permet à l'attaquant d'identifier un octet de la clé. La stratégie est alors d'essayer plusieurs paires de texte en clair jusqu'à ce que la différence entre les deux résultats permette de déterminer un octet de clé. L'attaquant répète ces étapes pour tous les octets de clé afin de récupérer la clé dans sa totalité.

D'autres attaques existent exploitant les interfaces de test JTAG, IEEE 1500 et IJTAG. Une attaque possible consiste à voler le fichier de configuration d'un FPGA lorsque celui-ci est envoyé par l'interface JTAG [26]. Ce fichier de configuration contient toutes les propriétés intellectuelles du designer. Une autre fonctionnalité du JTAG est le téléchargement de mises-à-jour de firmware. Cette fonctionnalité a été exploitée dans [27] afin de télécharger un firmware corrompu. Ce firmware corrompu permettait à l'utilisateur d'accéder à tout le service payant sur son décodeur TV. Les fonctionnalités de debug offertes par le standard JTAG permettent aussi à un attaquant d'étudier le logiciel installé sur le dispositif afin d'en trouver les failles. Dans [28], une faille a été trouvée dans l'iPhone pour insérer des cartes SIM non autorisées par le fabricant. Dans [29], l'exploitation d'une erreur dans le code a permis de faire fonctionner des jeux sans la vérification DRM sur la console de jeu Xbox 360.

Les attaques exposées jusqu'ici sont réalisées par un utilisateur non autorisé extérieur au circuit. Une autre classe d'attaquant peut être considérée, elle concerne les attaques réalisées par un élément malicieux à l'intérieur du circuit. Une attaque interne au circuit est présentée en [30], consistant à étudier le cas d'un circuit malicieux inséré dans la chaîne JTAG du système électronique. Ce circuit attaquant est alors capable de voler et de modifier les données de test passant à travers lui.

Plusieurs contremesures ont été proposées pour se prémunir des attaques exploitant les infrastructures de test. Une solution utilisée dans l'industrie consiste à déconnecter les chaînes de scan en brûlant des fusibles après le test de fabrication. Cette solution a été privilégiée sur la carte électronique de référence du projet fonctionnant avec le TEE. Le principal inconvénient de cette solution est que les opérations de diagnostic et de debug ne sont plus possibles. De plus, si un attaquant est capable de reconnecter les accès déconnectés à l'aide d'une sonde, cette contremesure devient inefficace.

Une autre contremesure est basée sur une approche alternative de DfT, le Built-In Self-Test (BIST) [36]. Cette technique limite le contrôle et l'observation des chaînes de scan pour le testeur externe, prévenant ainsi contre les attaques scan. Cette solution est intéressante dans le cas des crypto-processeurs, car ils sont généralement facilement testables avec des données pseudo-aléatoires [37]–[39]. Cependant, cette contremesure compromet elle aussi la maintenance lorsque le circuit est opérationnel.

Un autre type de contremesures est basé sur un mécanisme de verrouillage des infrastructures de test. Les moyens de test sont déverrouillés après un protocole d'authentification. Ces contremesures garantissent que seuls les utilisateurs autorisés peuvent déverrouiller les fonctionnalités de test soit à l'aide d'un mot de passe [49], [50] ou soit à l'aide de paires de challenge/réponse [51]–[58]. En plus de l'important surcoût en surface et en temps de test impliqués par ces solutions, une gestion de clé est requise afin de partager avec les utilisateurs autorisés le mot de passe ou les paires de challenge/réponse utiles au déverrouillage du mécanisme de protection.

Une autre famille de protections consiste en la détection du comportement d'un attaquant [62], [63]. Au lieu de verrouiller l'accès au test, ces solutions sont basées sur des détecteurs capables de distinguer entre le testeur et un possible attaquant. Ces contremesures ne préviennent uniquement que contre les menaces extérieures au circuit. Les possibles éléments malicieux insérés dans les chaînes de test ne sont pas considérés.

Une dernière catégorie de contremesures repose sur l'encryption du canal de test, ce qui assure la confidentialité des messages échangés entre le circuit et le testeur. L'encryption protège contre les utilisateurs non autorisés ainsi que contre les possibles éléments malveillants insérés dans la chaîne de test. Les attaquants ne sont pas capables de communiquer avec un circuit protégé, ni d'intercepter les communications. Pour encrypter les données de test, deux types de chiffrement peuvent être utilisés : le chiffrement par bloc, et le chiffrement par flot. Le choix entre les deux chiffrements dépend des performances recherchées ainsi que de la sécurité désirée. Dans la littérature ([30], [60] et [61]), le chiffrement préféré pour encrypter le canal de test est le chiffrement par flot. Les chiffrements par bloc ont un coût en surface plus important, et nécessitent d'adapter leur fonctionnement afin de traiter les communications de test qui sont réalisées en série. Néanmoins, les chiffrements par flot peuvent présenter une vulnérabilité de sécurité dans le cas d'une mauvaise implémentation. Nous montrons dans la section suivante que les solutions [30], [60] et [61] présentent cette vulnérabilité.

Dans cette thèse, nous proposons deux contremesures consistant à encrypter le canal de test : une première basée sur le chiffrement par flot, qui n'est pas exposée à la vulnérabilité mentionnée plus haut, une deuxième basée sur le chiffrement par bloc. L'état de l'art des contremesures basées sur l'encryption du canal de test est présenté dans la Section 2. La Section 3 présente les deux solutions proposées. La Section 4 consiste en l'évaluation de ces nouvelles solutions en termes de performance et de sécurité, ainsi qu'à leur comparaison. Finalement, la Section 5 conclut ce résumé concernant les travaux de thèse.

2. Etat de l'art des contremesures basées sur l'encryption du canal de test

Plusieurs solutions ont été proposées pour assurer la confidentialité des communications dans les infrastructures de test. Dans ces solutions, l'interface de test intègre des chiffrements en entrée et en sortie du canal de test. Le protocole de test s'en retrouve donc légèrement modifié. Premièrement, le testeur encrypte les données de test en dehors du circuit avec la clé secrète partagée avec le circuit. Deuxièmement, les données de test chiffrées sont envoyées au circuit, qui les décrypte à l'aide du chiffrement implanté à l'entrée de son infrastructure de test. Une fois les données de test décryptées, le circuit peut poursuivre les opérations de test. Avant que les réponses de test ne soient envoyées au testeur, le chiffrement implanté en sortie du canal de test se charge d'encrypter les réponses. Le testeur reçoit alors les réponses de test chiffrées, qu'il décrypte en dehors du circuit à l'aide de la clé secrète, puis les compare avec les réponses attendues. Avec ce schéma de communication, les données et les réponses sont gardées confidentielles durant le processus de test. Sans la connaissance de la clé, il n'est pas possible d'observer les états internes du circuit, ni possible de contrôler le circuit pour le mettre dans un état souhaité. De cette façon, il n'y a pas de risques que les données chiffrées soient lues par une tierce personne non autorisée.

Puisque les données dans les infrastructures de test sont transmises en série, la plupart des schémas d'encryption sont basés sur le chiffrement par flot. Les auteurs [30] proposent d'utiliser le TRIVIUM [59] comme chiffrement par flot pour encrypter les communications JTAG. Le TRIVIUM génère un flux de nombre pseudo-aléatoire à partir d'une clé secrète et d'une valeur d'initialisation (IV). Dans [30], l'IV est codé en dur à l'intérieur du circuit grâce à des fusibles programmés au moment

de la fabrication. La clé secrète est quant à elle dérivée d'un challenge envoyé par le testeur. Seul un utilisateur autorisé connaît les paires de challenge/réponse, et donc la clé secrète utilisée pour encrypter les données de test.

La solution proposée en [60] traite le problème de possibles cores malveillants insérés dans les SoCs. La contremesure protège contre le risque qu'un core malveillant vole des données de test passant par le canal de test. Pour ce faire, un chiffrement par flux TRIVIUM encrypte les données de test. La clé secrète est générée aléatoirement par le testeur. Elle est ensuite envoyée au core protégé en utilisant une chaîne de scan dédiée et non visible des autres cores. Concernant l'initialisation du TRIVIUM, les auteurs ne spécifient pas la configuration de l'IV.

Le chiffrement par flot est aussi utilisé en [61] pour encrypter le réseau scan reconfigurable dans le standard IJTAG. Un TRIVIUM décrypte et encrypte les données envoyées et reçues du réseau scan reconfigurable. Le but est de protéger les instruments de test contre le vol de données par un instrument malicieux dans la chaîne de test, ainsi que contre les attaquants extérieurs souhaitant utiliser illégalement les instruments embarqués. Concernant l'initialisation du chiffrement par flux, les auteurs proposent des ensembles uniques de clés et d'IVs pour chaque instrument protégé, mais les auteurs ne mentionnent pas le changement de clé ou d'IV entre chaque session d'encryption.

Or il est essentiel pour assurer une bonne protection de changer la clé secrète ou l'IV entre chaque session d'encryption. Autrement, le chiffrement par flot présente une vulnérabilité appelée « *two times pad* ». On considère deux réponses de test R_1 et R_2 et un chiffrement par flux qui encrypte ces réponses en sortie de chaîne de test. Pour la première réponse R_1, le chiffrement par flot génère un flux de bits pseudo-aléatoire, noté S, à partir d'une clé secrète et d'une valeur initiale IV. Le chiffré de la réponse est alors $R_1 \oplus S$. La réponse R_2 est obtenue après un reset du circuit, réinitialisant par la même occasion le chiffrement par flot. Si la même clé secrète et la même IV sont utilisées pour encrypter R_2, alors le chiffré est $R_2 \oplus S$. Dans le cas des attaques scan telles que présentées en [13]–[16], l'attaquant procède au XOR entre les deux réponses, donnant comme résultat $[R_1 \oplus S(K, IV)] \oplus [R_2 \oplus S(K, IV)] = R_1 \oplus R_2$. L'impact du chiffrement par flot est alors retiré, rendant le circuit vulnérable aux attaques scan. Un attaquant peut donc réaliser des attaques scan même si les réponses de test sont encryptées.

Les contremesures présentées en [30], [60] et [61] sont toutes exposées à cette vulnérabilité, puisque la gestion de l'IV et de la clé secrète n'est pas correcte. La même paire (clé secrète, IV) ne doit pas être utilisée plus qu'une fois pour éviter la génération du même flux pseudo-aléatoire pour différentes encryptions.

3. Proposition de contremesures

Dans ce chapitre, nous proposons premièrement une contremesure basée sur le chiffrement par flot qui ne présente pas la vulnérabilité des contremesures de l'état de l'art. Nous proposons ensuite une méthode d'encryption de la chaîne de scan basée sur le chiffrement par bloc.

Dans les deux contremesures proposées, un premier chiffrement est implanté en entrée de chaîne de scan en charge de décrypter les patterns de test. Un deuxième chiffrement est implanté en sortie

de chaîne de scan en charge d'encrypter les réponses de test. En plus de l'implémentation des chiffrements, l'encryption du canal de test requiert d'implanter un mécanisme de gestion de clé afin de partager de façon sécurisée la clé secrète entre le circuit et le testeur. Nous supposons que le dispositif électronique doit être protégé des attaques scan, et donc qu'il embarque un ou des crypto-processeur(s). Nous proposons de réutiliser la gestion de clé déjà implantée pour le crypto-processeur ainsi que son stockage sécurisé, pour gérer et stocker la clé encryptant le canal de test. De cette façon, la clé est stockée de façon sécurisée dans le crypto-processeur, et elle est partagée avec les utilisateurs autorisés sans introduire de problèmes spécifiques de gestion de clé.

a) Encryption du canal de test avec le chiffrement par flux

Concernant la contremesure basée sur le chiffrement par flot, la nouveauté de notre approche comparée à l'état de l'art réside dans le fait que l'IV n'est pas une valeur constante, mais qu'elle est générée aléatoirement par un générateur de nombre aléatoire, True Random Number Generator (TRNG). L'utilisation du TRNG garantit de ne jamais réutiliser la même IV pour la génération du flux pseudo-aléatoire. Ainsi, le chiffrement par flot ne présente pas la vulnérabilité *two times pad* décrite dans la section précédente.

La valeur aléatoire de l'IV doit être connue par le testeur extérieur afin que celui-ci puisse communiquer avec le circuit. Pour rendre la valeur de l'IV accessible à l'extérieur du circuit, une instruction JTAG, appelée *GETIV*, est ajoutée au jeu d'instructions. Lorsque le testeur exécute cette instruction, l'IV générée aléatoirement est envoyée en sortie de l'interface de test. Le testeur n'a plus qu'à lire la valeur de l'IV afin de pouvoir encrypter et décrypter les données de test en dehors du circuit. Il est important de noter que la sécurité de la solution n'est pas compromise en rendant l'IV public, puisque la clé du chiffrement par flot est toujours gardée secrète. Un attaquant n'est pas capable de déchiffrer les communications chiffrées sans la clé secrète.

L'encryption des données de test est réalisée avec le chiffrement par flot TRIVIUM. Un seul TRIVIUM est capable de générer deux flux pseudo-aléatoires : l'un pour décrypter les données arrivant en entrée du canal de test, l'autre pour encrypter les réponses sortant du canal de test. Avant la génération des flux pseudo-aléatoires par le TRIVIUM, une phase d'initialisation est nécessaire pour que le TRNG atteigne une entropie suffisante pour générer un nombre aléatoire. Le chiffrement par flot est lui aussi initialisé. Pendant l'initialisation, les chaînes de scan ne sont pas accessibles, toutes les opérations de test sont considérées comme une instruction *BYPASS*. Une fois la phase d'initialisation achevée, le jeu d'instructions est déverrouillé. Seulement un utilisateur, connaissant la clé secrète et ayant récupéré l'IV générée aléatoirement par le circuit grâce à l'instruction *GETIV*, peut communiquer avec le circuit.

b) Encryption du canal de test avec un chiffrement par bloc

Dans ces travaux de thèse, nous avons aussi proposé une contremesure basée sur le chiffrement par bloc. Nous avons choisi des chiffrements par bloc dits « légers » afin de limiter le coût en surface et en consommation de puissance. Deux chiffrements par bloc légers sont implantés, l'un pour le déchiffrement effectué en entrée du canal de test, et l'autre pour l'encryption effectuée en sortie du canal de test. Chacun des deux chiffrements possède deux registres de rondes afin de réaliser en parallèle l'encryption des données et l'acquisition série des données du canal de test. La réalisation de ces opérations en parallèle permet de ne pas perdre en temps de test.

Puisque les communications de test sont effectuées en série, l'utilisation du chiffrement par bloc implique d'ajouter des données supplémentaires aux données de test initiales afin de former des segments de la taille du bloc encrypté/décrypté. La chaîne de scan est ainsi remplie et vidée segment par segment. Chaque segment est composé d'un bloc de données sur N bits (N représentant la taille du bloc encrypté/décrypté). Quand la chaîne de scan n'est pas multiple de N, la solution fonctionne toujours mais chaque pattern est complété avec des données supplémentaires. En considérant R le reste modulo N de bascules dans la chaîne de scan, il est nécessaire de compléter les données de test par N-R bits. Les données supplémentaires demandent du temps de test additionnel pour que le testeur les envoie dans la chaîne de scan. Le résultat est que cette solution a un surcoût en temps de test dépendant du nombre de pattern nécessaire pour tester le circuit. Ce coût est évalué à N-R coups d'horloge pour chaque pattern de test.

Pour réduire ce coût, une optimisation de la solution a été proposée. Elle consiste à exploiter le temps de test additionnel ajouté à chaque pattern pour améliorer la testabilité du circuit. En effet, en ajoutant N-R bascules scan au circuit, il est possible de les utiliser en tant que points d'observation sur la logique du circuit. Les points d'observation sont une technique de test permettant de propager des signaux « difficilement » observables directement à des bascules scan, sans affecter le temps de test initial. Le but de cette technique est de réduire le nombre de patterns nécessaire pour tester un circuit, tout en préservant le même taux de couverture de fautes.

En conclusion, les deux contremesures proposées permettent de se protéger contre les menaces utilisant l'infrastructure de test, tout en préservant les fonctionnalités de debug et de diagnostic lors de possibles maintenances. C'est un principal avantage comparé à la solution consistant à déconnecter les accès de test. Un autre avantage concerne la réutilisation de la gestion de clé déjà implanté dans le circuit. Les contremesures basées sur un protocole sécurisé utilisant des primitives cryptographiques, comme [49]–[58], ont quant à elles besoin d'avoir une gestion de clé dédiée.

Les deux solutions ont fait l'objet d'une publication dans un journal [69], et à plusieurs présentations à des conférences [70]–[72] ainsi qu'à des workshops [76]–[81]. Dans la section suivante, nous comparons les deux solutions.

4. Evaluation et comparaison des solutions proposées

Les contremesures consistant à encrypter le canal de test sont évaluées par rapport à la protection apportée sur l'infrastructure de test, les coûts en termes d'implémentation et l'intégration de la solution dans un SoC.

a) Analyse de sécurité

Concernant la sécurité, le déchiffrement effectué en entrée du canal de test empêche d'écrire une valeur désirée dans la chaîne de scan sans la connaissance de la clé. Le chiffrement effectué en sortie du canal de test empêche l'observation des états internes du circuit, sans avoir réalisé le déchiffrement au préalable. Les possibles éléments malicieux insérés dans les infrastructures de test ne peuvent donc pas voler des données confidentielles. De plus, l'encryption du canal de test permet de se prémunir des attaquants extérieurs au circuit, souhaitant par exemple réaliser des attaques scan [13]–[16] ou

exploiter les fonctionnalités JTAG telles que le debug [28][29], l'accès au réseau IJTAG, ou le téléchargement de firmware [27].

La proposition de contremesure basée sur le chiffrement par flot présente un niveau de sécurité plus important que les solutions proposées dans l'état de l'art. En effet, la mauvaise gestion de clé secrète et d'IV rend les contremesures [30], [60] et [61] vulnérables aux attaques scan. Nous proposons de générer aléatoirement l'IV initialisant le chiffrement par flot afin d'éviter cette vulnérabilité, et par conséquent protéger contre les attaques scan.

Finalement, les deux propositions d'encryption du canal de test (par le chiffrement par flux et par le chiffrement par bloc) protègent contre les menaces liées à l'utilisation de l'infrastructure de test.

b) Coûts d'implémentation

En terme de testabilité, l'encryption du canal de test permet de tester le circuit original sans réduire le taux de couverture de fautes. En effet, les patterns du circuit original sont appliqués tels qu'ils sont après leur déchiffrement. Cependant, l'architecture de la solution a besoin d'être testée, sans l'aide des chaînes de scan qui exposeraient les chiffrements implantés aux attaques scan. Nous proposons de tester fonctionnellement les chiffrements en utilisant les patterns dédiés à tester le circuit original. Le test des chiffrements par bloc est facilité par les propriétés de diffusion des algorithmes cryptographiques ([37] et [38]). Concernant le test des chiffrements par flot, le flux de bits pseudo-aléatoire généré par le TRIVIUM est une combinaison des états internes du chiffrement, le rendant facilement testable. Les deux chiffrements, par bloc et par flot, propagent donc facilement les erreurs à leur sortie lorsqu'une encryption est réalisée. Pour valider cette hypothèse, nous avons mené des expérimentations consistant à appliquer la séquence de test de plusieurs circuits sur les chiffrements implantés. Pour tous les circuits considérés, le taux de couverture de fautes dans l'architecture de la contremesure est de 100%. En d'autres termes, les chiffrements implantés sont testés en même temps que le circuit original, sans application de séquence particulière.

Nous avons aussi mené des expérimentations pour évaluer le surcoût en terme de surface et de temps de test dans le cas d'une simple chaîne de scan. Si un TRNG est déjà implanté dans le circuit alors la solution basée sur le chiffrement par flot peut l'utiliser pour générer l'IV, limitant le coût en surface. Dans ce cas, le chiffrement par flot présente une surface moindre (5 408,52 µm² pour l'implémentation avec le TRIVIUM) comparée au chiffrement par bloc (9 282.52 µm² pour l'implémentation avec le chiffrement par bloc SKINNY). Il est clair aussi que le chiffrement par flot surpasse le chiffrement par bloc en terme de coût en temps de test, à cause du surcoût à chaque pattern pour ce dernier. Ce coût peut cependant être réduit en appliquant l'optimisation consistant à insérer des points d'observation sur le circuit original.

Les deux solutions ne sont pas limitées à une seule chaîne de scan. Nous avons proposé de les étendre pour protéger des chaînes de scan multiples, indépendamment de l'implémentation de techniques de compression. La contremesure basée sur le chiffrement par flot présente un coût plus faible en terme de surface (de 5 553,60 µm² à 9 999,60 µm² pour l'implémentation avec TRIVIUM) et de puissance consommée (environ 35 µW), comparé à la contremesure basée sur le chiffrement par bloc (pour l'implémentation avec SKINNY, environ 10 000 µm² pour la surface, et de 128,2 µW à 2 232,9 µW pour la puissance consommée). Cependant, le nombre limite de chaîne de scan traité par

un chiffrement est plus faible (jusqu'à 32 chaînes de scan) pour la contremesure avec le chiffrement par flot, comparé à la limite de la contremesure avec le chiffrement par bloc (jusqu'à 64 chaînes).

c) Intégration dans un SoC

Les infrastructures de test de chacun des cores composant un SoC sont souvent connectées en série les uns après les autres. L'entrée et la sortie série de l'interface de test permet alors d'accéder aux chaînes de scan des différents cores dans le SoC. Quand un core implante des chiffrements en entrée et sortie de chaîne de scan, tous les autres cores de la chaîne de test reçoivent des données encryptées, protégeant ainsi contre les potentiels cores malicieux. L'encryption avec un chiffrement par flux consiste à une opération bit à bit, n'impliquant donc aucun problème d'intégration de la solution dans un design SoC. A l'inverse, l'encryption avec le chiffrement par bloc nécessite d'ajouter des données supplémentaires afin d'avoir des segments de la taille du bloc encrypté. Ces données supplémentaires sont alors reçues par les autres cores du canal de test, pouvant créer des problèmes dans les opérations de test. Par conséquent, le concepteur doit être conscient du problème potentiel de l'intégration dans un SoC de la solution basée sur le chiffrement par bloc. Un moyen pour éviter d'envoyer des données supplémentaires dans la chaîne de test est insérer des bascules dans la chaîne de scan connectées à des points d'observation sur la logique du circuit. De cette façon, la chaîne de scan est multiple de la taille du bloc encrypté. Cependant, l'insertion des points d'observation implique de modifier le circuit original. Il n'est donc pas possible d'appliquer la solution dans le cas d'un core non modifiable.

Pour conclure la comparaison, le choix entre les deux contremesures proposées dépend du circuit original où la protection est implantée : si un TRNG est déjà implanté dans le circuit original, si la longueur de la chaîne de scan est multiple du bloc encrypté, si le circuit original peut être modifié afin d'insérer des points d'observation. Le choix dépend aussi du nombre de chaîne de scan qui doit être encrypté dans le cas de la configuration multiple des chaînes de scan.

La comparaison des deux solutions a fait l'objet d'une soumission au journal *Transactions on VLSI Systems* ainsi qu'une présentation à une conférence [74].

5. Conclusion

Les circuits cryptographiques sont largement répandus pour assurer la confidentialité dans les systèmes électroniques. C'est le cas par exemple dans les smartphones où les opérations d'encryption sont réalisées pour échanger ou stocker des informations confidentielles. Dans ce cadre, les opérations d'encryption sont réalisées dans un environnement de confiance, le TEE. Cependant, cet environnement n'est pas suffisant pour assurer la protection des crypto-processeurs contre des attaques matérielles telles que l'exploitation des infrastructures de test.

Les tests de production des circuits cryptographiques sont obligatoires afin d'assurer la qualité du produit. Il est d'autant plus important de réaliser le test car des défauts physiques sur le circuit peuvent compromettre la sécurité. L'implémentation des techniques de test (chaînes de scan, JTAG, IEEE 1500 et IJTAG) est donc essentielle. Néanmoins, ces infrastructures de test peuvent être un point d'entrée pour un attaquant souhaitant corrompre la sécurité du système. Une contribution de cette thèse

concerne la proposition de nouvelles contremesures consistant à encrypter le canal de test. Plusieurs contremesures basées sur le chiffrement par flot ont été précédemment proposées, mais présentant toutes une faiblesse dans leur implémentation. Nous avons proposé une nouvelle implémentation sans la vulnérabilité des contremesures de l'état de l'art, ainsi qu'une solution basée sur des chiffrements par bloc légers. Le principe de ces contremesures est l'insertion de modules de chiffrement/déchiffrement en amont et en aval du canal de test. L'encryption est effectuée à l'aide d'une clé secrète partagée avec les utilisateurs autorisés grâce à la gestion de clé déjà implantée dans le circuit. Un avantage important est de conserver les fonctionnalités de diagnostic et de debug pour les utilisateurs autorisés.

Dans cette thèse, nous avons aussi évalué et comparé les deux solutions. Les résultats expérimentaux ont montré que l'encryption avec le chiffrement par flot présente un coût plus faible en surface, en temps de test et en puissance consommée, dans le cas où un générateur de nombre aléatoire est déjà implanté dans le circuit. Dans le cas contraire, l'encryption avec le chiffrement par bloc est une bonne alternative, avec une possible optimisation pour réduire le coût en temps de test et faciliter l'intégration dans un design SoC.

REMERCIEMENTS

Je souhaiterais remercier plusieurs personnes qui ont contribué par leur aide et par leur soutien à la réalisation de cette thèse. Premièrement, je suis reconnaissant d'avoir eu Bruno Rouzeyre, Marie-Lise Flottes et Giorgio Di Natale pour encadrer ma thèse. Ils m'ont aidé à appréhender le contexte de la thèse, et à mener à bien ces travaux en m'apportant chacun leurs conseils et leurs idées. Les différentes contributions de cette thèse sont le résultat d'une excellente ambiance de travail, propice à réaliser d'intéressantes recherches.

Cette thèse n'aurait pas pu voir le jour sans le financement du projet FUI TEEVA. Je remercie les partenaires du projet que j'ai pu rencontrer à diverses occasions lors de réunions pour leur apport dans la réalisation de cette thèse.

Durant ces trois années, j'ai pu compter sur l'aide de plusieurs personnes en plus de mes encadrants : Sophie Dupuis, qui m'a aidé dans mes travaux et qui a participé à plusieurs publications, et Papa Sidy Ba, un ancien doctorant, qui m'a apporté sa maîtrise des outils Design Compiler et TetraMax utiles aux expérimentations menées dans cette thèse. Je n'oublie pas Emanuele Valea, mon collègue de bureau et ami, avec lequel j'ai collaboré depuis août 2017. Je me considère chanceux d'avoir pu bénéficier de son point de vue perspicace durant ma dernière année de thèse, qui a permis une entraide très efficace pour mener à bien ces travaux et pour les publier.

Je tiens également à remercier Lorena Anghel et Lilian Bossuet d'avoir accepté de rapporter cette thèse. Merci aussi à David Hély et à Alberto Bosio d'avoir accepté de faire partie du jury.

Mes années au LIRMM ont été une formidable expérience, où j'ai reçu un très bon accueil. Je voudrais remercier spécialement Caroline Lebrun, secrétaire du département microélectronique, toujours présente pour m'aider et répondre à mes questions.

Les doctorants du LIRMM ont aussi participé à ce que l'ambiance de travail soit très agréable à travers nos pauses cafés. Je peux même en compter un grand nombre en tant qu'amis, qui ont participé à enrichir ma vie montpelliéraine avec les différentes sorties organisées.

Je tiens finalement à remercier toutes les personnes de mon entourage qui m'ont soutenu durant ces trois ans, amis et famille. Malgré la distance, mes parents et mon frère Florian ont toujours été là pour m'aider dans mon quotidien. Un énorme merci aussi à Sara qui m'a apporté soutien et motivation pour mener à bien cette thèse.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

AES	*Advanced Encryption Standard*
AHB	*Advanced High-performance Bus*
AP	*Access Port*
APACC	*Access Port ACCesses*
APB	*Advanced Peripheral Bus*
ATE	*Automatic Test Equipment*
ATPG	*Automatic Test Pattern Generator*
AXI	*Advanced eXtensible Interface*
BIST	*Built-In Self-Test*
BSR	*Boundary Scan Register*
BYP	*Bypass register*
CBC	*Countermeasures Based on Block cipher*
CSC	*Countermeasures Based on Stream cipher*
CTR	*Counter mode*
CUT	*Circuit Under Test*
DAP	*Debug Access Port*
DES	*Data Encryption Standard*
DfT	*Design-for-Testability*
DfD	*Design-for-Debug*
DP	*Debug Port*
DPACC	*Debug Port ACCesses*
DRM	*Digital Rights Management*
ECC	*Elliptic Curve Cryptography*
FF	*Flip-Flop*
FPGA	*Field-Programmable Gate Array*
FSM	*Finite State Machine*
GE	*Gate Equivalent*
HMAC	*Keyed-Hash Message Authentication Code*
IC	*Integrated Circuit*
IDCode	*Identification Code register*
IJTAG	*Internal JTAG*
IP	*Intellectual Property*
IR	*Instruction Register*
IV	*Initialization Vector*
JTAG	*Join Test Action Group*

LFSR	*Linear Feedback Shift Register*
LOC	*Launch-On-Capture*
LOES	*Launch-On Extra-Shift*
LOS	*Launch-On-Shift*
LSIB	*Locking Segment Insertion Bit*
LUT	*LookUp Table*
MAC	*Message Authentication Code*
MMU	*Memory Management Unit*
n.s.	*not specified*
NS	*Non-Secure*
OEM	*Original Equipment Manufacturer*
OS	*Operating System*
PRNG	*Pseudo Random Number Generator*
PUF	*Physically Unclonable Function*
RAM	*Random Access Memory*
REE	*Rich Environment Execution*
RSA	*Ronald Rivest, Adi Shamir, Leonard Adleman*
RSN	*Reconfigurable Scan Network*
SE	*Scan Enable*
SI	*Scan-In*
SIB	*Segment Insertion Bit*
SKMU	*Secret Key Management Unit*
SO	*Scan-Out*
SoC	*System-on-Chip*
TA	*Trusted Applications*
TAP	*Test Access Port*
TCK	*Test Clock*
TDI	*Test Data Input*
TDO	*Test Data Output*
TDR	*Test Data Register*
TEE	*Trusted Environment Execution*
TMS	*Test Mode Select*
TRNG	*True Random Number Generator*
TRST	*Test Reset*
WBR	*Wrapper Boundary Register*
WBY	*Wrapper Bypass*
WCK	*Wrapper Clock*
WID	*Wrapper Identification register*
WIR	*Wrapper Instruction Register*
WPI	*Wrapper Parallel Input*

WPO	*Wrapper Parallel Output*
WSI	*Wrapper Serial Input*
WSO	*Wrapper Serial Output*

INTRODUCTION

The context of this thesis is in the Trusted Environment Execution eVAluation (TEEVA) project, which aims to give a security evaluation of the Trusted Environment Execution (TEE) technology. The TEE is mainly used in mobile platforms to provide a secure environment for the execution of critical-security application, such as bank transactions. The goal of this thesis focuses on the attacks carried out through the test infrastructures, and their countermeasures.

Testing is a mandatory task in the Integrated Circuit (IC) production process to ensure product quality. In the area of digital testing, test procedures require to implement dedicated design for test purposes, called Design-for-Testability (DfT), to reduce efforts for test pattern generation. The most popular DfT method relies on scan chains usage. This approach consists in replacing original registers by serial scan registers, and connecting these scan registers into one or several scan chains. Extra control signals allow shifting in and out test data through the scan chain(s), providing full control and observation of the circuit internal states. Scan design greatly reduces the complexity of the test pattern generation and the overall test application time.

In addition to scan chains, test standards have been proposed to reach the interface of the scan chains for each device in complex ICs. This avoids connecting each device to the external pins. The first test standard to be proposed for board testing is the IEEE Std. 1149.1, also known as JTAG. It offers an interface to the user that relies on the Test Access Port (TAP) controller. The user has the access to a scan network, which connects all the devices on the board in a daisy-chain fashion. The same principle has then been extended to System-on-Chips (SoCs) with the IEEE Std. 1500. This allows testing the internal cores of a SoC equipping them with standardized test wrappers. More recently, the IEEE Std. 1687, also known as IJTAG, has been released to facilitate the access to the hundreds of embedded instruments that are present in nowadays SoCs. The IJTAG is based on a Reconfigurable Scan Network (RSN) that can be properly set by the user according to which instruments he/she wants to reach.

The access to the test infrastructure of the whole SoC by a malicious user represents a very serious security threat. While testability is positively impacted by scan designs thanks to full control and observation of IC internal states, the confidentiality of processed data is unfortunately negatively affected for exactly the same reasons. Indeed, scan attacks exploit facilities offered by the scan chains to retrieve the secret data processed by the device. These attacks target secure circuits implementing a cryptographic algorithm and storing a secret key. They rely on the possibility for hackers to shift out the scan chain content while the circuit state is correlated with the secret, i.e., the key.

Moreover, test infrastructures based on JTAG, IEEE 1500 and IJTAG standards can be exploited for a wide range of attacks. For instance, this is the case when the JTAG interface is used to access the system memory for debugging purposes, allowing an attacker to corrupt the firmware, to steal Intellectual Property (IP), or to find a flaw in the executed program.

Many research works have been dedicated to propose countermeasures against attacks exploiting test infrastructures. The main difficulty in the scan attack prevention stems from the need to maintain

both data security and hardware quality ensured by test, diagnostic and debug activities. The cost of the scan attack prevention in terms of circuit characteristics (power, performances, area), insertion in the design flow, test quality and test time is an important issue. While several solutions have been proposed to prevent scan attacks, no proposal is available for providing both high security and quality in preserving test, diagnostic and debug facilities at low cost and at all cycles of the device lifespan.

The contribution of this thesis is to present a secure, plug-and-play, and cost-efficient mechanism for preventing scan attacks, while providing full control and observation on the scan content for authorized users. The main idea is to maintain the confidentiality of the scan content through encryption/decryption of the test data flowing through the test infrastructures. The encryption can be performed either with block ciphers, or with stream ciphers. The choice of stream vs block cipher is driven by performance and security trade-offs. Because of their smaller footprint, stream ciphers are generally preferred to block ciphers in the literature. Nevertheless, when incompletely implemented, solutions based on stream ciphers are prone to scan attacks and thus have to be completed. This mitigates their interest versus block cipher solutions. In this thesis, we propose two new solutions: one exploiting block ciphers, and another exploiting stream ciphers fulfilling security requirement. We detail the advantages of these solutions compared to the state-of-the-art countermeasures. Moreover, we give a comparison between these two scan encryption countermeasures.

This thesis is divided in six chapters. Chapter I presents the context of the thesis on the security evaluation of the Trusted Environment Execution (TEE) technology with respect to the hardware access provided by the test infrastructures.

Chapter II brings out the security threats from the misuse of test infrastructures. The threat model includes “external attackers” (with unauthorized to access the test interfaces), as well as “internal attackers” (with insertion of malicious core connected to the test infrastructure at design time). A main attack from external attackers is the scan-based attack, exploiting the observation offered by the scan chain in order to steal confidential data, such as the secret key of a crypto-processor. Chapter II also reports our experiments and analysis on the project reference board considering the scan attack, and show that it implements the most common and radical industrial countermeasure consisting in disconnecting the test accesses after manufacture testing. A discussion about limitations in terms of security and debug/diagnostic facilities conclude the chapter.

Furthers scan-attack countermeasures have been developed in the literature. In Chapter III, the state-of-the-art of such countermeasures is presented. We propose a taxonomy according to the target strategy consisting in avoiding the scan chains, modifying the scan chains structures, providing a secure test access, or detecting illegitimate behavior in the use of the test interfaces.

Chapter IV presents a new approach based on the encryption of the test communication, ensuring the confidentiality of the exchanged data between the device under test and the tester. Several architectures are explored using block or stream ciphers with the goal to ensure same testability for the protected circuit than before scan protection.

Chapter V gives a comparison of the proposed solutions in terms of implementation costs, testability evaluation, and integration in a SoC design.

Finally, in Chapter VI we draw some conclusions and perspectives.

Chapter I

Context

Summary

I.1. Introduction to TEEVA project

This thesis is part of the TEEVA (Trusted Environment Execution eVAluation) project, a FUI project led by Gemalto (FUI n°20 from January 2016 to December 2018) with the following partners Trustonic, Phonesec, EMSE, Laboratoire Hubert Curien and LIRMM. TEEVA tackles the sensitive threats on mobile platforms. Smartphones represent high-value targets for hackers since they contain personal data and are used for mobile payment. The Android platform is a main target for identity theft, or financial data theft. Application stores propose a lot and a wide diversity of applications, such as email, games, social media, and bank applications. Among this large number of applications, some malwares could execute malicious program and steal sensitive data processed by other applications.

As illustrated in Figure 1, a malware application downloaded from a store could exploit a flaw in the Operating System (OS) (e.g., Android) and steal confidential banking information processed at the same time by a banking application. The appearance of many malware on the Android market requires the Original Equipment Manufacturers (OEMs) of mobile platforms to implement appropriate security solutions.

The goal of the TEEVA project is to evaluate two security solutions for mobile platforms: one is purely software: the Whitebox Crypto solution, while the other one integrates software and hardware elements: the Trusted Environment Execution (TEE). The security evaluation of the TEE is performed specifically against hardware attacks, including the ones from an access through the test interfaces evaluated by the LIRMM. The TEEVA project also aims to propose new countermeasures against the disclosed attacks on the TEE in order to improve the security level.

The principle of a TEE is to have an extra execution environment isolated from the environment executing the main rich operating system such as Android. TEEs are developed by Trustonic and others companies, and is based on ARM TrustZone Technology [1] available on many chipsets of the

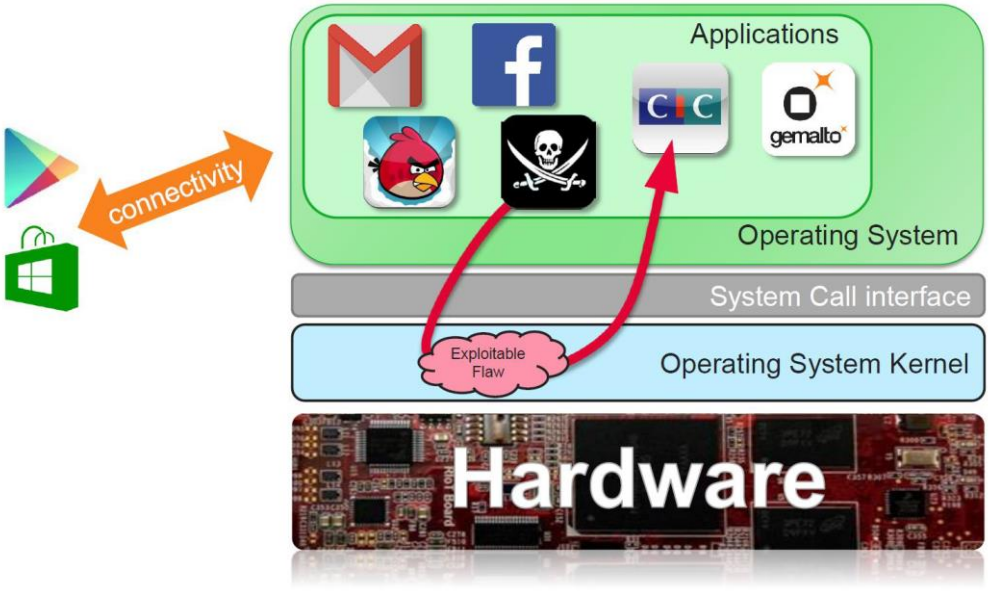


Figure 1: Security threats on mobile platforms.

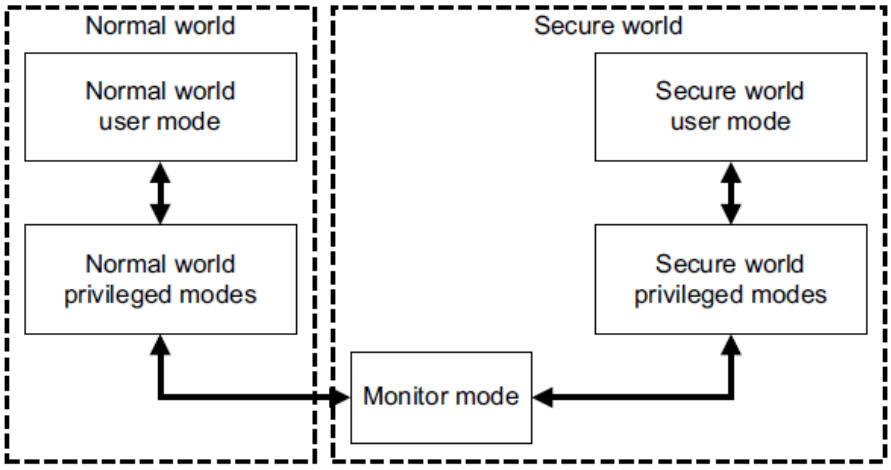


Figure 2: Modes in an ARM processor implementing TrustZone [1] technology.

smartphone and tablet market mainly. The isolation mechanism between the two environments is achieved by defining two states of the processor (Figure 2): a Secure world and a Non-Secure world. Both worlds have modes based on privileges: a user one and a privileged one. The secure state of the processor is stored in a special register, Secure Configuration Register, on one bit, called the Non-Secure (NS) bit. When the NS bit is set to 1 (respectively 0), the processor is in the Non-Secure world (resp. in the Secure world). Switching between the two worlds is ensured by the so-called Secure Monitor, which saves and restores save the context of each world. The Secure world is not restricted to the ARM processor, but extended to the whole System-on-Chip (SoC) design. In addition to the standard ARM buses signals, one additional signal propagates the NS bit from the processor to the peripherals and memories. Thus, each core on the SoC receives the processor state, in order to operate either in the Secure World or in the Non-Secure world. Concerning the memories, the physical memory is partitioned into two regions: a Secure one and a Non-Secure one. The virtual addresses are on 33 bits instead of the usual 32 bits, in order to include the “NS” bit. The Memory Management Unit (MMU) is in charge of decoding the virtual addresses to allow or not writing and reading operations on a physical memory location, given NS-bit value. The TrustZone features allow thus to define at hardware level a Secure world and a Non-Secure world.

Applications are then split into two categories: normal applications and Trusted Applications (TA). The normal applications run within the main OS (usually Android), called the Rich Execution Environment (REE), while TAs are executed within the TEE. As illustrated in the example in Figure 3, normal applications include Wi-Fi connection, messages applications, touch keyboard, and geolocation; while TAs are related to bank transactions, Digital Rights Management (DRM) and health data. When TAs are executed, the peripherals being accessed are only visible in the TEE. For instance, when the pin code of the bank account is entered on the touch keyboard of the mobile phone, the touch keyboard becomes a secure peripheral in order to be invisible from the REE side. A potential software attack carried out from the REE has no impact on the assets protected on the TEE side since the isolation between the two environments is at both hardware and software levels thanks to the TrustZone technology.

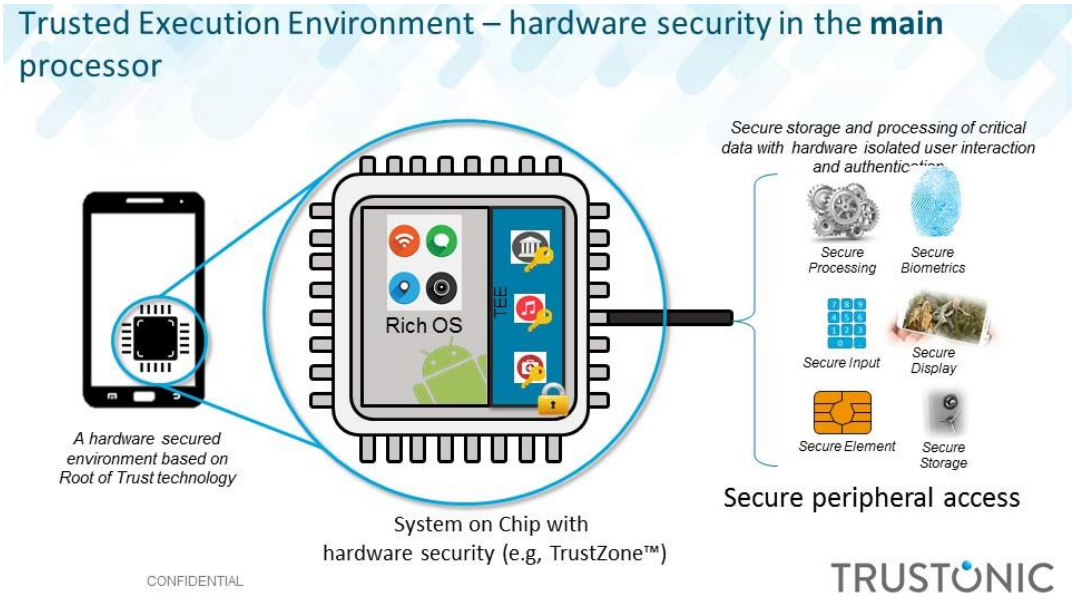


Figure 3: Mobile chip operating with the TEE.

The isolation between the TEE and the REE provides a safe environment execution for TAs at software level. However, security against hardware attacks needs to be evaluated. The attacks based on information gained from the physical implementation of a secure system are called *side-channel attacks*. Power consumption, cache timing, electromagnetic fields are sources of information on the processed data, which can be exploited by an attacker. For instance, a weak implementation of crypto-processors can leak information about the secret key used during encryption/decryption even if the implemented crypto-algorithm is secure from a mathematical point of view.

In this thesis, we evaluate specially the security of a TEE with respect to the fraudulent usage of the test infrastructure and its interface. The exploited data in that case are not side channel information such as power consumption, but processed data extracted from the chip through the test infrastructure while the crypto-processor executes encryption/decryption operations. We present in the next Section the context of testing, diagnosis and debugging, and the several test infrastructures implemented on a device.

I.2. Test, diagnosis and debug for Integrated Circuits

Test is a mandatory step in Integrated Circuits (ICs) production for weeding out bad products before they reach end-users. Indeed, physical defects at the production can imply that the IC does not behave as expected. Testing allows detecting these defects by exercising the Circuit Under Test (CUT) with specific test patterns and comparing its outputs with a golden reference. Figure 4 illustrates the process. At design time a logical simulation is used to execute test patterns on the reference (circuit model), and retrieve expected responses to the test sequence. After manufacturing, the same test patterns are applied to the produced physical CUTs. The test responses from every CUT are then collected and compared with expected ones. The comparison distinguishes the faulty circuits from correct ones as long as test patterns are able to detect potential defects (e.g. open, short-circuits, damaged doping). Because of the number and types of the defects to be considered, test pattern

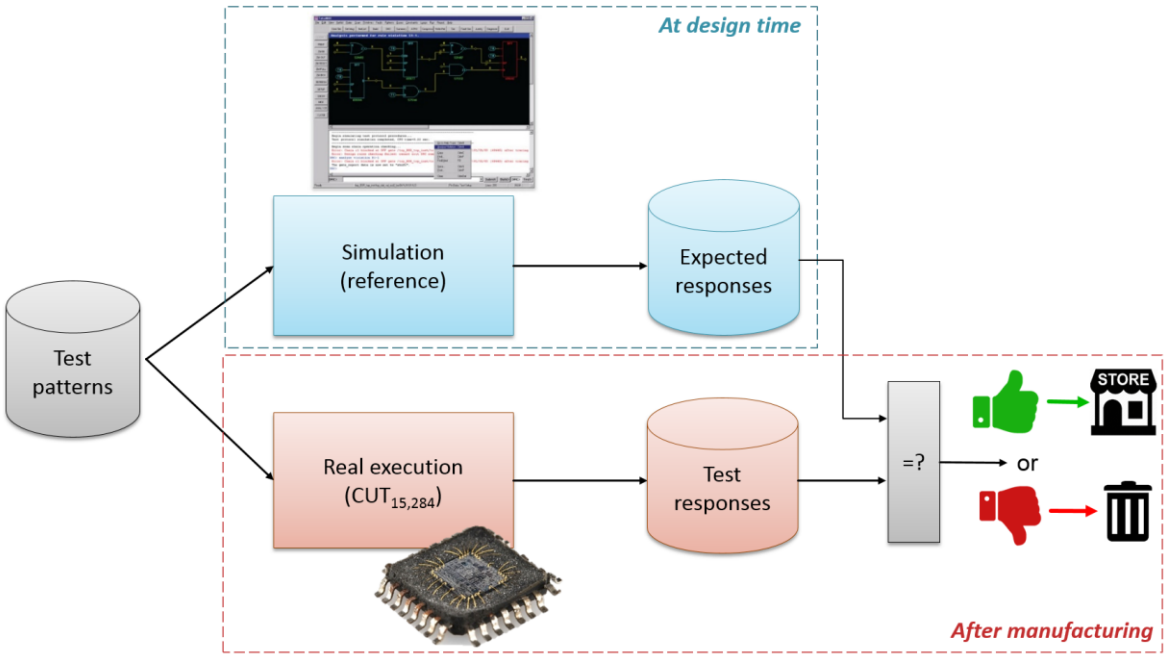


Figure 4: Testing principle.

generation relies however on fault models (e.g. stuck-at faults, delay faults). The academic state-of-the-art and the industrial practice consists in quantifying the quality of a test sequence in terms of "Fault (model) Coverage" (FC%) computed as the ratio between the number of fault detectable by the sequence, and the total number of faults in the considered model. Additionally, the test sequence length is of prime importance since it must be applied on every single produced circuit (recurrent cost).

The observed errors (faulty test responses) may also due to bad design practices that can be corrected before new productions. After few runs, tests target manufacturing defects. In order to potentially correct a design or calibrate manufacturing steps and thus improve the yield, diagnosis helps to locate defaults. Additionally, diagnosis is also useful when the device is already in the field. An effective diagnosis on physical failure location allows fast and targeted maintenance, detection of uncorrected design errors.

Automatic Test Pattern Generator (ATPG) (e.g. Synopsys Suite, TetraMAX [2]), helps for the generation of test patterns and computes expected test responses. Today's test pattern generation tools have reached their maturity, the ATPG problem is known to be NP-complete, the computation complexity increases linearly with the sequential depth of the circuit, and exponentially in the presence of sequential cycles. Industrial state-of-the-art tools are thus efficient on combinational circuits but do not perform accordingly on sequential ones. For these reasons, specific design techniques allow to deal with sequential circuits and generate high quality test sequences. Next section presents the scan chain insertion approach and its usage according to the targeted fault model.

I.2.a Design-For-Testability and test application

The most common DfT technique is the usage of scan chains. As illustrated in Figure 5, it consists in replacing original flip-flops (FFs) by so-called scan FFs organized in shift registers. The internal states of the circuit can thus be controlled from the serial input of the scan chain, Scan-In (SI), and observed

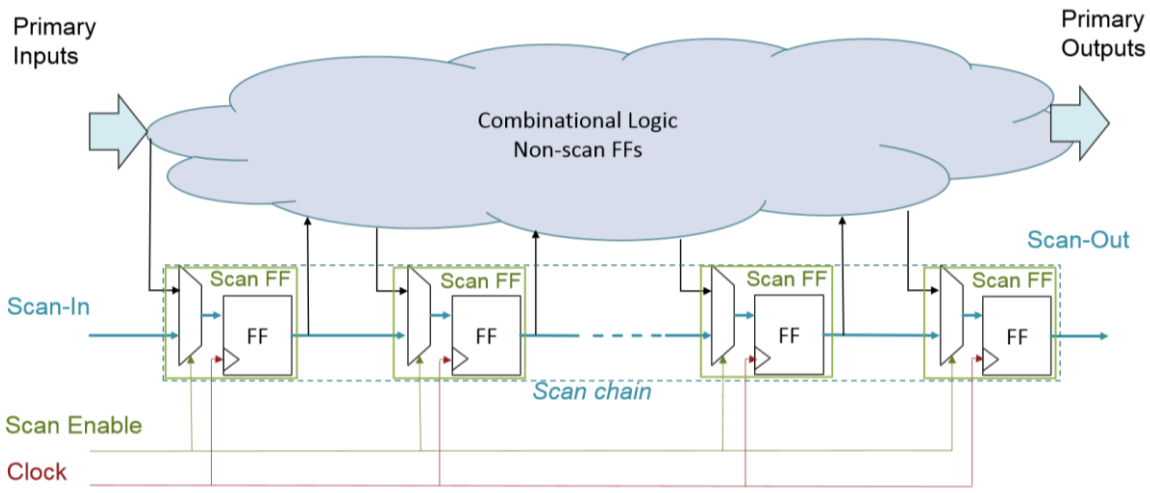


Figure 5: Single scan chain.

through the serial output of the scan chain, Scan-Out (SO). The signal Scan-Enable (SE) selects either the test mode or the functional mode for the scan FFs. The controllability and the observability offered by the scan chain allow ATPGs to treat sequential circuits as combinational ones since sequential depth and cycles are eliminated (Scan FFs are controllable and observable from pseudo-primary inputs/outputs). The patterns generation and the scan procedure then differ according to the fault models. While test generation is not useful for the comprehension of the sequel, scan test procedures are hereafter detailed according to fault models because our countermeasures to scan-based attacks will be also evaluated according to their ability to permit required procedures.

I.2.a.i Scan test procedure for stuck-at faults testing

The procedure of stuck-at faults testing consists in 1/ applying the test patterns shifted serially in the scan chain at SI pin in test mode (SE = 1), 2/ potentially setting non-latched primary inputs to required values, 3/ running the circuit in functional mode (SE = 0) and storing back test responses in the scan chain, 4/ collecting the test responses on potential non-latched primary outputs and shifted serially out the scan chain at SO pin in test mode (SE = 1) while next test pattern is concurrently shifted in the scan chain. In case of multiple scan chains, the process is executed in parallel over the chains.

Figure 6 describes an example of the test procedure for testing stuck-at faults. In the example, the scan chain is composed of 3 scan FFs where 3 test patterns p_1, p_2 and p_3 are applied, and 3 test

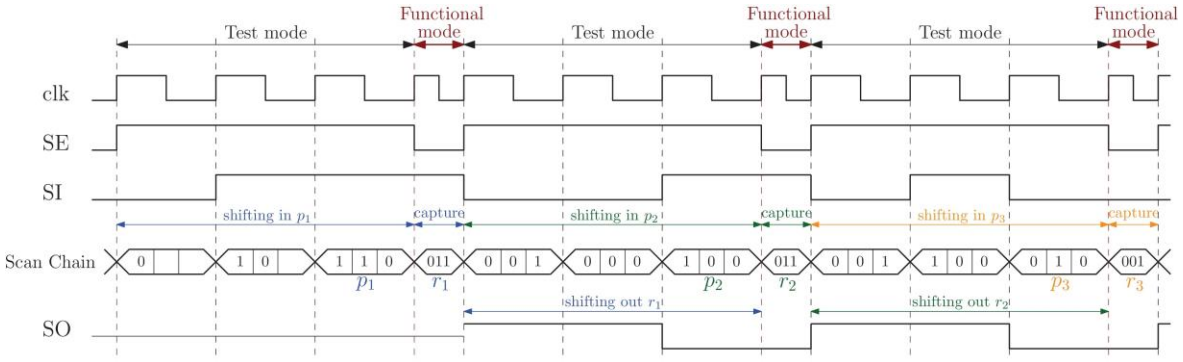


Figure 6: Example of scan procedure for testing stuck-at faults.

responses r_1, r_2 and r_3 are collected. The first pattern p_1 is applied with 3 successive shifting operations in test mode. The shifting operations are realized at a lower frequency than the nominal frequency to avoid over-heating of the circuit. When the test pattern p_1 fills the scan chain, the corresponding response r_1 is captured by switching the circuit from test mode to functional mode. The capture operation is realized at the nominal frequency of the system. The circuit is then switched from functional mode to test mode. The response r_1 is shifted out the scan chain, and concurrently, the next pattern p_2 is shifted in the scan chain during 3 shifting operations. The scan procedure follows this scheme until all the test responses are shifted out the circuit.

Formally, by considering a CUT with F scan FFs tested with K test patterns, each test vector is applied after $F + 1$ clock cycles, corresponding to F shifting operations and the application of the test vector during 1 clock cycle in functional mode. Test responses are concurrently shifted out the scan chains, except for the last test response where F extra clock cycles are needed to shift out this last response. Overall, the number of clock cycles T required to test the CUT is given in equation (1):

$$T = K(F + 1) + K \qquad (1)$$

This equation can be simplified considering that the number of F scan FFs is significantly larger than 1. In this case, equation (1) becomes: $T = F(K + 1)$.

I.2.a.ii Scan test procedure for transition-delay faults testing

The goal of transition-delay fault testing is to detect slow-to-rise and slow-to-fall faults. In order to excite these faults, at-speed transitions have to be achieved within the circuit. The transition is performed between an initialization vector V_1 and the resulting vector V_2. Two complementary methods are used to test these faults by using scan chains: Launch-On-Shift (LOS) [3] and Launch-On-Capture (LOC) [4]. In both solutions, the shifting operations to initialize the scan chain with V_1 are performed at shift frequency (usually much slower than the nominal clock frequency), while the transition to V_2 is captured at the circuit nominal frequency.

Figure 7 (a) presents the LOS scheme on a circuit having F scan FFs, which consists in shifting test data during $(F - 1)$ cycles, launching the transition on the last shift at the nominal frequency, and

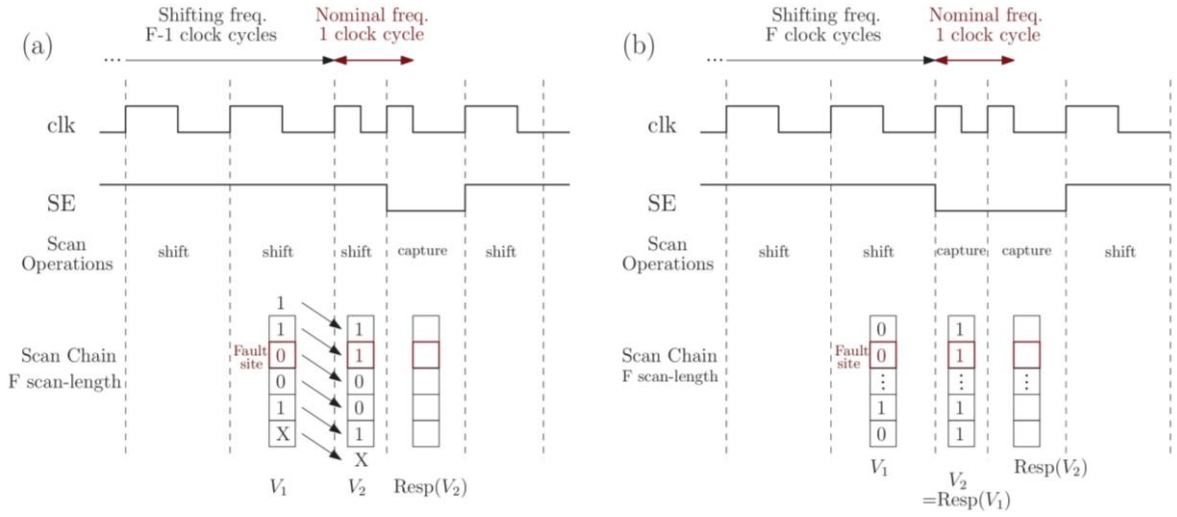


Figure 7: (a) Launch-On-Shift (LOS) scheme. (b) Launch-On-Capture (LOC) scheme.

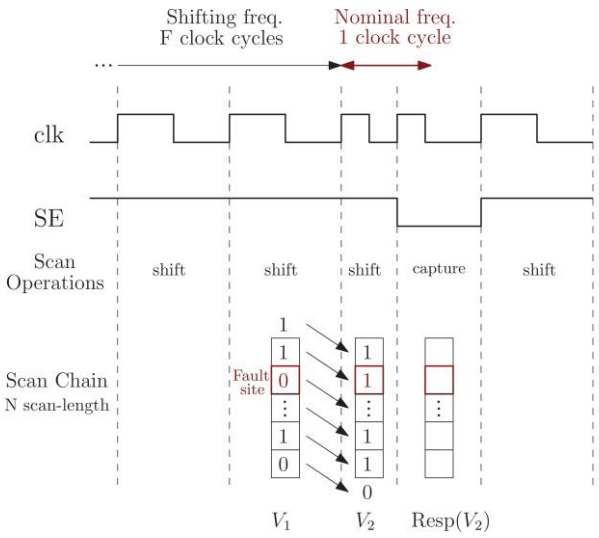


Figure 8: Launch-On Extra-Shift (LOES) scheme.

capturing the response at nominal frequency as well. In this scheme, vector V_2 is the same vector as V_1 in which every bit is shifted 1 position (following the order in the scan chain). In the case of LOC scheme, the transition is launched on the capture of the previous clock cycle where V_1 is used to exercise the circuit. In other words, vector V_2 is the circuit response to vector V_1. Figure 7 (b) details the LOC scheme.

An alternative method to transition-delay faults testing proposed in the literature is called Launch-On Extra-Shift (LOES) [5]. As illustrated in Figure 8, the LOES scheme consists in launching the transition on an extra-shift after the scan chain is completely filled. For a circuit with F scan-length, the LOES scheme applies F shifting operations (at the shift frequency) in order to initialize the scan chain, then a shift operation at the nominal frequency to launch the transition, and finally another clock cycle at the nominal frequency to capture the circuit response. Although the LOES test is proposed on some ATPG tools, the common solution to achieve high delay fault coverage is to perform the LOS test first, and then the LOC test on the faults left undetected by LOS.

I.2.a.iii Insertion of test points

Test point insertion is a classical Ad Hoc DfT procedure that consists in adding extra control or observation points to the circuit logic in order to improve its testability. It comes in addition to scan design if the combinational logic is enough "testable" and helps ATPG to find test patterns for uncovered faults. Figure 9 presents the principle of the insertion of test points. Control points consists in inserting a gate in the circuit connected to a scan cell. The control point is inserted at a node with few controllability in order to set easily the node to a desired state in test mode. Usually, control points allows improving the test coverage by controlling the untestable nodes in a circuit. Observation points allows improving propagation of test responses to observable points, i.e. scan FFs. Observability improvement usually results in reducing the test sequence length (less test patterns for same fault coverage).

Test point insertion does not affect the scan procedure for testing stuck-at faults (detailed in I.2.a.i) and for testing transition-delay faults (detailed in I.2.a.ii).

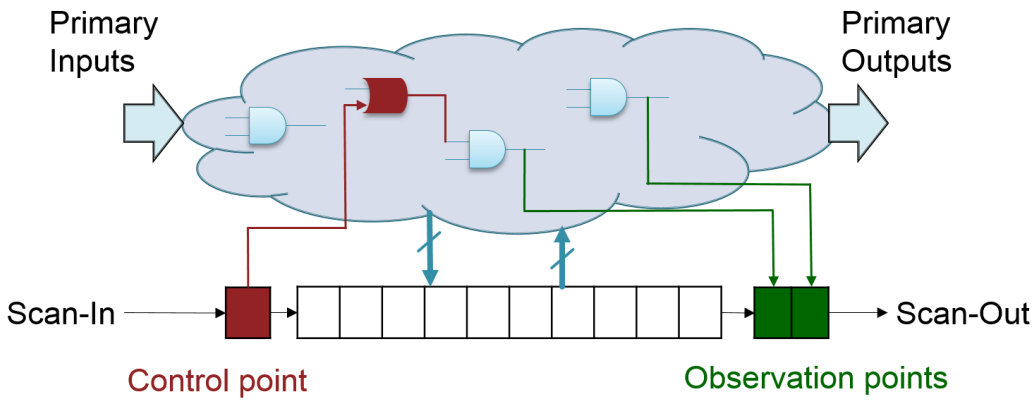


Figure 9: Insertion of control points and observation points on a circuit.

I.2.a.iv Multiple scan chains

Tests costs, including test data storage and test application time components, increases with the complexity of ICs. To overcome the test time issue, multiple scan chains can be inserted within the circuit. Figure 10 describes a multiple scan chains architecture (L scan chains). The test procedure of a classical multiple scan design consists in applying the test vectors in the L parallel SIs of the CUT. The scan chains are filled *slice* by *slice*. Slice is the set of scan FFs of the same rank within multiple scan chains (see slices $S_0 \dots S_{Z-1}$ in Figure 10). During one shifting operation, the tester, named Automatic Test Equipment (ATE), sends the content of one slice S_i to the CUT. Once all the scan chains are filled, the CUT runs in functional mode. Each scan slice contains thus a part of the test responses. The CUT is switched from functional mode to test mode in order to shift out the content of each scan slice at each shifting operations. The responses are then collected by the ATE on the L parallel SOs of the CUT. As for single scan chain, the next test pattern is shifted in the circuit at the same time as the current test response is collected.

Compared to a single scan chain with LxZ scan FFs, test time for L multiple scan chains of Z scan FFs is thus reduced by a factor L. However, the multiple scan chain scheme does not reduce the volume of data required to test a circuit. In addition, CUTs may have a limited number of inputs/outputs to control and observe many scan chains.

To overcome these issues, the current solution consists in compressing test data before application and observation. ATPG generates compressed stimuli to apply to the CUT, and compacted responses for future comparison. As illustrated in Figure 10 in the case of integration of test compression techniques, ATE stores the compressed test data, reducing the volume of data stored in the tester. As for the CUT, a decompressor is implemented in the circuit at the scan chains' inputs in order to fill scan chains with the decompressed test vectors. Before test responses are shifted out the circuit, a compressor at the scan chains' outputs compress them. The implemented decompressor and compressor within the CUT reduce the scan pins required to access the multiple scan chains.

The methods for test compression proposed by Mentor Graphics and Cadence are respectively Embedded Deterministic Test [6] and 2D Elastic Compression [7]. For both methods, the decompressor is based on a sequential circuit and a XOR network. ATE sends compressed stimuli corresponding to the seed to initialize the sequential circuit, either a Linear Feedback Shift Register (LFSR) or a ring

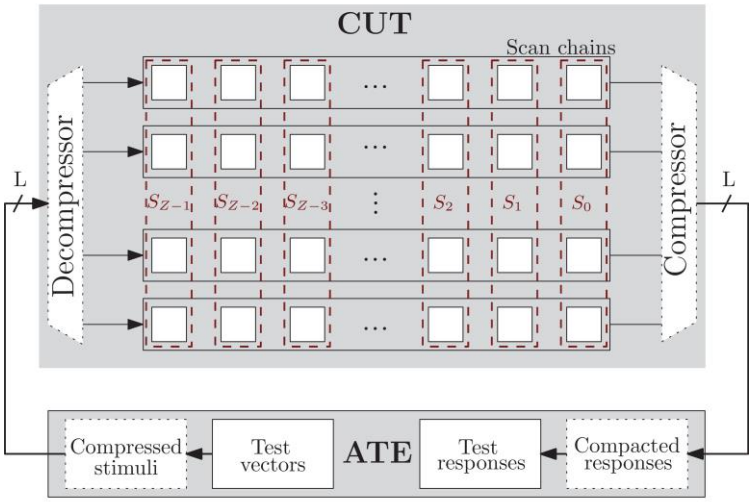


Figure 10: Multiple scan design with the possible integration of test compression techniques.

generator. The sequential circuits then generate the test vectors filling the scan chains. The test responses are compressed in a signature before shifting out at scan outputs. The signature is generated with a XOR operations between the responses contained in the scan chains. ATE needs then to compare the signatures with the expected ones in order to detect faults within the circuits.

Last, the scan chains in a single or multiple configuration are accessible through scan pins. During manufacturing, these pins are accessible by dedicated test pads placed on the die. Chip packaging and/or SoC integration usually move away the external test pins accessible during manufacturing. In the first place, test standards have been developed to address the need to access the scan chains after IC packaging. We present in the next subsection the three main test standards.

I.2.b Test standards: JTAG, IEEE 1500 and IJTAG

The first test standard to be proposed for board testing was the IEEE Std. 1149.1 [8], also known as Join Test Action Group (JTAG). Instead of using probes for testing circuit interconnections on a board, board's circuits are equipped with a wrapper and Test Access Protocol (TAP) controller allowing to set and collect test data on board interconnections (external test procedure). The standard allows to propagate test data to and from a circuit on the board too (internal test procedure). The TAP test interface includes four mandatory signals: a serial input, Test Data Input (TDI), and a serial output Test Data Output (TDO), a Test Clock (TCK) signal, and a control signal Test Mode Select (TMS); one optional signal is possible: Test Reset (TRST). The JTAG interface connects all the components mounted on the board into a test daisy-chain as described in Figure 11. Each device contains a TAP controller for interfacing internally with the test ports, i.e. a 16-bit Finite State Machine (FSM) driven by the TMS and TCK signals generally distributed to all the TAP controllers. The TDI/TDO signals of each device are connected together in order to form the daisy-chain connection. Each device also includes an Instruction Register (IR), and Test Data Registers (TDRs): a register to retrieve the component ID code (IDCode), a boundary scan register (BSR) for accessing primary inputs and internal scan chains, and a bypass register in order to shift data to the next circuit on the board. The JTAG protocol, managed by the TAP controller, is based on four basic operations on these registers: shift, capture and update for both instruction and data registers. *ShiftIR* connects the serial pins TDI/TDO to the IR register.

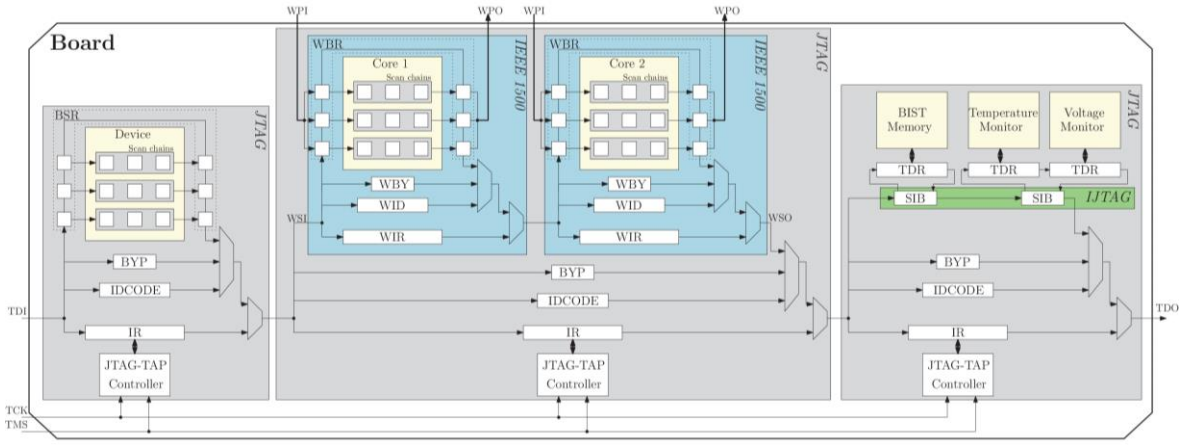


Figure 11: JTAG daisy-chained components and integration with IEEE 1500 (blue area) and IJTAG (green area) standards.

CaptureIR loads a vector into the IR register. *UpdateIR* makes the instruction contained in the IR register the current instruction. *ExitIR* finishes the executed operation on the IR. The JTAG protocol begins by loading the IRs of each device with an instruction to be executed. Some instructions are mandatory in the standard (*BYPASS, SAMPLE/PRELOAD, EXTEST*), but many other custom instructions can be added (e.g. *INTEST, RUNBIST* for execution of built-in self-test procedures). Each instruction selects an access to a specific TDR that is managed by the same basic operations as for the IR register, *ShiftDR*, *CaptureDR*, *UpdateDR* and *ExitDR*. For example, the *BYPASS* instruction connects at least one flip-flop called BYP between the TDI and TDO pins in order to make the target device transparent to the network. The *EXTEST* instruction is used to test off-chip circuits and board level-interconnections by accessing the Boundary Scan Register (BSR). The *INTEST* instruction allows a tester to shift test patterns into the circuit and collects the test responses, potentially by accessing the internal scan chains of the device. Each device has a proper identification code, accessible with the *IDCODE* instruction and stored into the identification register (IDCode).

IEEE 1500 standard has been developed later to deal with the large number of cores implemented in Systems on Chips (SoCs). For saving time during SoC testing, the test vectors cannot be applied only serially, using the 1-bit TDI/TDO pins of the JTAG interface. As illustrated in Figure 11 (blue area), the IEEE 1500 [9] provides an architecture very similar to the JTAG. The JTAG TDI, TDO and TCK signals are represented in IEEE 1500 standard as Wrapper Serial Input (WSI), Wrapper Serial Output (WSO), and Wrapper Clock (WCK). IEEE 1500 standard offers a test wrapper around each core. The test wrapper is composed of a Wrapper Instruction Register (WIR), a Wrapper Bypass (WBY), a Wrapper Identification register (WID), and a Wrapper Boundary Register (WBR), similar respectively to IR, BYP, IDCode and BSR registers in JTAG standard. A main difference with JTAG is that the IEEE 1500 has parallel test accesses mechanisms for testing cores, providing access to several scan chains and thus shorter test times. The test data are sent through the Wrapper Parallel Inputs (WPI) and test responses are collected through the Wrapper Parallel Outputs (WPO). Another difference is that the FSM mandatory in the JTAG architecture is not included in the IEEE 1500 standard. As a result, the SoC designer has the choice to either use the JTAG-TAP controller to manage the test operations on the IEEE 1500 test wrapper (as illustrated in Figure 11), or to implement a dedicated IEEE 1500-TAP controller. The TAP controller chosen by the designer manages the registers of the IEEE 1500 wrapper by the four

operations of shift, capture, update and exit. Unlike board testing, where circuits are previously and independently tested before bonding on the board, internal cores in a SoC cannot be tested in advance. The cores are manufactured all together to produce the SoC and test is mandatory after system production.

In addition to JTAG and IEEE 1500, a recent standard has been set up to deal with the important number of embedded instruments within a chip. Embedded instruments are dedicated to testing, such as Memory Built-In Self-Test (BIST), or measurement purposes, such as Temperature Monitor and Voltage Monitor. The IEEE 1687 [10] standard, called Internal JTAG (IJTAG), allows accessing the different instruments through a Reconfigurable Scan Network (RSN). The RSN offered by the IJTAG standard is accessible through a specific instruction in the JTAG standard. Concerning the RSN itself, the IJTAG defines the Segment Insertion Bit (SIB) in order to give access to a segment of TDRs connected to embedded instruments. The tester configures the RSN by choosing the SIB to open, giving the access to the desired segment. In the case of the circuit described in Figure 11 (green area), the RSN is composed of two SIBs: one given access to a Memory BIST, the other given access to 2 monitoring instruments.

Figure 11 illustrates how the three standards can be found interleaved together. Even when IEEE 1500 and IJTAG infrastructures are implemented in a system, the test interface for an external tester is always with the JTAG interface at board level.

If the JTAG standard was firstly developed for testing purposes, additional features have been integrated to the test interface. JTAG interface allows namely to update firmware (e.g. Pay-TV box). Field-Programmable Gate Array (FPGA) configuration file (bitstream) can be set through the JTAG interface. Finally, the JTAG infrastructure is also used to debug the system in the field. In the next subsection, we detail the Design-for-Debug (DfD) approach implemented in ARM designs, as well as its integration with the TrustZone technology.

I.2.c Overview of the Design-for-Debug implemented in ARM designs

The DfD implemented in ARM designs gives 3 potential debug accesses.

A first access is the system bus implemented on the SoC, either the Advanced High-performance Bus (AHB) or the Advanced eXtensible Interface (AXI).

A second access is the dedicated debug bus Debug Advanced Peripheral Bus (Debug APB), connected to CoreSight components [11]. CoreSight components include mainly Cross Trigger Interface, which allows cross triggering at the same time different cores in a SoC (e.g. halt two processors), and Embedded Trace Macrocell, which collects debug data of a specific core (e.g. read registers content). These components are configured through the Debug APB to set breakpoints to halt a core on specific activity, to examine and modify registers and memory, and to provide single-step execution.

In addition to the accesses to the ARM busses, a third potential debug access is connected to the JTAG wrappers. In this case, the debug is performed by controlling and observing the internal scan chains within the circuits with the JTAG instruction *INTEST*.

While the first two accesses to the system bus AHB or AXI and Debug APB, provide debugging on the executed program (e.g. setting breakpoints, reading the memory, single-step executing), the access to the JTAG wrappers provide debug and diagnosis on the design hardware (e.g. reading and writing on specific scan FFs).

Each debug access is not necessarily implemented, this is the choice of the SoC designer to implement a debug access.

In order that an external user access the 3 potential debug accesses, boards based on ARM processors implement a non-standard JTAG interface, as described in Figure 12, called JTAG–Debug Port (JTAG–DP). The external interface has the same ports proposed by the JTAG, including TDI, TDO, TCK and TMS, but with a different TAP controller, called Debug Access Port (DAP). The set of instructions is composed of two mandatory instructions in the standard, *BYPASS* and *IDCODE*, and custom instructions, *ABORT, D*ebug *P*ort *ACC*esses (*DPACC*) and *A*ccess *P*ort *ACC*esses (*APACC*).

The *ABORT* instruction is used to clear error in the TDRs, and returns an OK response once the clearing operation is achieved. The *DPACC* instruction is used to configure the JTAG–DP DAP, especially to select the path among the 3 possible accesses: system bus (AHB or AXI), debug bus (Debug APB) and standard JTAG wrappers. The configuration is saved on the DPACC register. The *APACC* is used to communicate with the selected Access Port (AP). The three APs ensure the data exchange between an external tester and the selected debug access. The debug data are stored into the APACC register, accessible thanks to the *APACC* instruction.

The APs related to the system bus (AHB or AXI) and the debug bus (APB) implement an extension to the TrustZone protecting the debug data saved and processed in the Secure Zone. All the data passing through the ARM buses AHB, AXI or APB, contain the NS bit introduced by the TrustZone. The associated APs control thus if the data comes from Secure Zone or Non-Secure Zone in function of the value of the NS bit on the bus. The decision to allow debugging the Secure Zone depends on the value of enable signals on the APs. The AHB–AP (or AXI–AP) has two enable signals: a first one, *global_debug_enable*, for enabling completely the debugging access through the system bus, a second one, *secure_debug_enable*, for enabling the debugging access through the system bus for the Secure Zone.

Concerning the debug APB–AP, only one enable signal, *debug_enable*, controls the debugging functions through the debug bus access. This signal enables/disables the debugging access through the debug APB either completely, or only on the Secure Zone. According to ARM specifications [11], these enable signals for both AHB–AP and debug APB–AP are defined either with fuses, enabling/disabling the debugging functions for the entire life of the IC, or with an authentication module, enabling/disabling the debugging functions only for authorized users after an authentication protocol. The implementation of the enable signals depends on the choice of the SoC designer.

Concerning the JTAG–AP, no signal controls the debug access as for the two other APs. Since the data obtained from the JTAG wrappers are not a path containing the NS bit as for the ARM buses AHB, AXI and APB, the debug data obtained through the serial interface TDI/TDO are not differentiated between the Secure Zone and Non-Secure Zone. Consequently, the access to the JTAG wrappers can be exploited in order to steal confidential data in the Secure Zone.

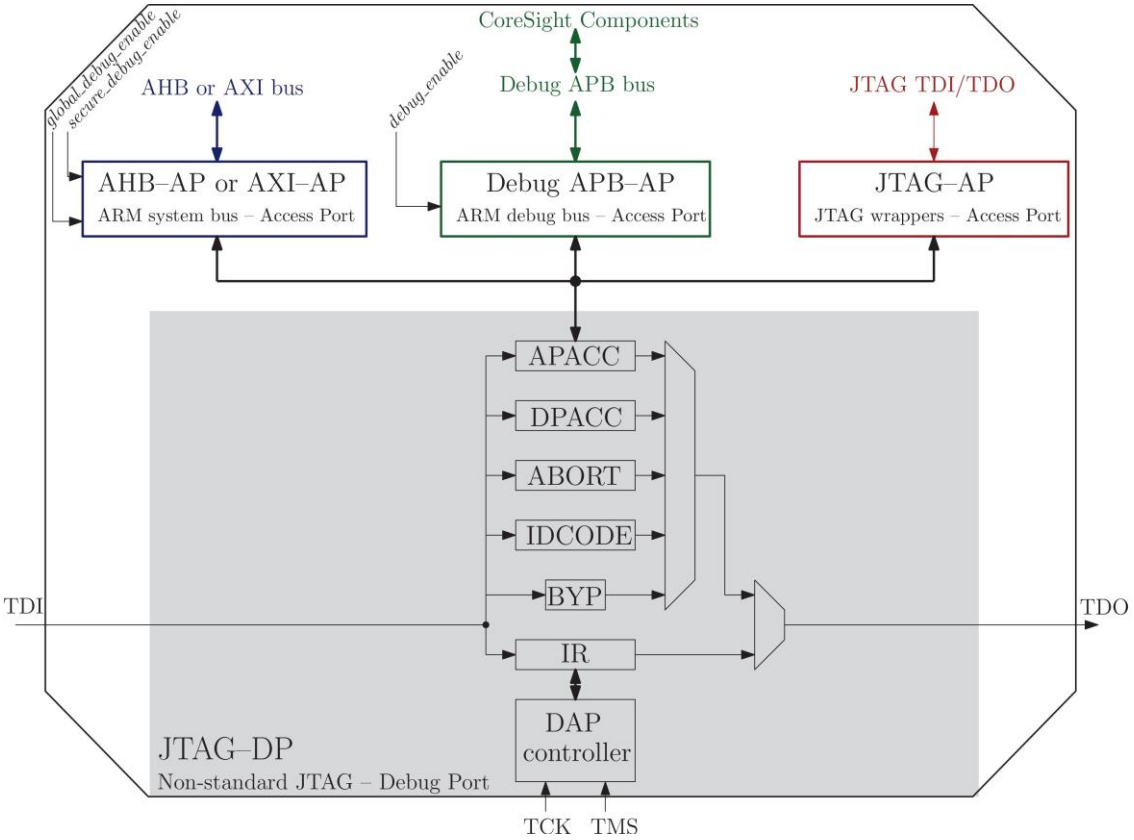


Figure 12: Overview of the non-standard JTAG–Debug Port (JTAG–DP) giving 3 accesses to (1) the ARM system bus (AHB or AXI); (2) the ARM debug bus (Debug APB); and (3) the standard JTAG wrappers.

I.3. Conclusion on test infrastructures

Testing is essential to ensure the product quality, as well as in-field debugging and diagnosis in order to provide feedback information during IC life, such as hardware failure or software error. To provide testability on a complex system, designers have to insert DfT techniques, such as scan designs, and test interfaces, defined in the test standards, JTAG, IEEE 1500 and IJTAG. This way, a tester can test individually the components of a system, as well as their interconnections. In addition to the test infrastructures, ARM proposes DfD technique in order to provide debugging facilities on boards based on ARM processors. Among the debug accesses, the access of JTAG wrappers represents a threat for board operating with TEE since the NS bit is not managed for this access. For this reason, a countermeasure against attacks exploiting test interfaces has to be implemented, such as the case for the project reference board. As detailed in next Chapter II, the test accesses on this board have been disconnected. Next chapter details the threats for every circuits implementing a test interface, including devices operating with the TEE.

Chapter II

Security threats through test infrastructures

Summary

II.1. Threat model related to test infrastructures

Although IC designer needs to implement test infrastructures within the circuit in order to provide test, debug and diagnosis facilities, these infrastructures present an important security threat since they offer control and observation on the data saved and processed by each component composing the IC. Therefore, an unauthorized user may potentially exploit the test infrastructures to carry out attacks aiming to steal confidential data (e.g. secret key, configuration files), or to obtain an elevated privileged access in the system. As described in Chapter I, the common configuration of the test infrastructures is the daisy-chain. The daisy-chain configuration implies another threat coming from inside the IC since all devices belonging to the IC are connected together. If a malicious device is implemented within the test daisy-chain, this device can interfere with the test data passing through itself. This way, the malicious device may steal confidential data from another device in the chain, or tamper the test responses of another device in order to give fake test responses to the tester.

Figure 13 depicts several threats in the test infrastructures composed of the 3 standards (JTAG, IEEE 1500, IJTAG). This chapter details these threats classed in two main threats. We consider as a first threat, an external attacker unauthorized to access the test interface. The second considered threat is internal to the circuit with the malicious and counterfeit components integrated to the IC.

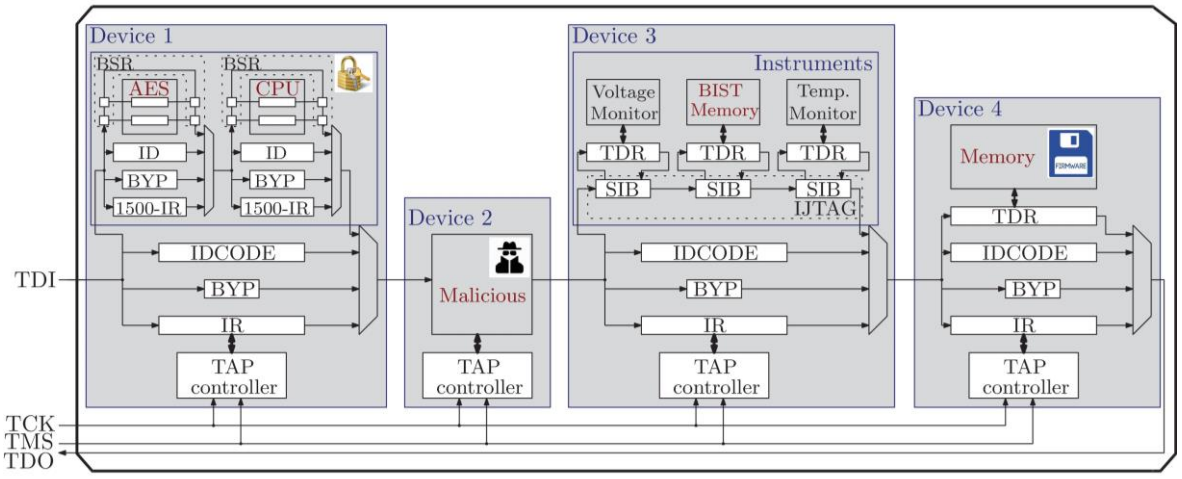


Figure 13: JTAG daisy-chained components (Device 1) a SoC containing an AES crypto-core and a CPU, (Device 2) a malicious core, (Device 3) a device with embedded instruments accessible through an RSN, and (Device 4) a memory contained a firmware.

II.1.a External attacker

As seen in Chapter I, the test interface has many features (e.g. accessing scan chain, configuring FPGA, updating firmware, debugging a system, accessing embedded instruments). All these features gives to an external attacker numerous and diverse possible attacks. For instance, he/she can exploit the observability and controllability offered by the scan chains to steal confidential information. The so-called scan attacks have been presented in the literature to steal secret keys of crypto-processors. Another attack concerns the Intellectual Property (IP) theft, such as for the IPs contained in the FPGA configuration file. Since the test interface also allows to update firmware, an external attacker can

exploit this feature to update a corrupted firmware. Debugging offered by the test interface can also be exploited in order to find a flaw in the system. The last studied attack concerns the IJTAG network. An attacker can exploit the hundreds of embedded instruments connected to the reconfigurable network in order to overheat the circuit for instance. We detail each studied attack performed by an external attacker in the following sections.

II.1.a.i Scan attacks

A main threat coming from the test infrastructures is due to the scan chains, since they offer a complete control and observation on the internal states of a circuit. Two types of attacks are possible: one exploiting the observation on internal signals, the other exploiting both control and observation. The principle of scan attacks by observation is in a first step to apply a desired value to the primary inputs. The second step involves switching from functional mode to test mode. In test mode, the attacker shifts out the responses and can thus observe intermediate states. These steps are repeated until the attacker gets enough information to steal the secret.

Scan attacks by control and observation differ from the previous ones by using scan input instead of primary inputs. The desired value is applied by shifting data on the targeted registers. The rest of the attack is the same as the one by observation: switching from functional mode to test mode and collecting responses. The analysis of the responses serves to leak secret of the circuit.

Crypto-processors are the main target of scan attacks, as depicted in Figure 13 with the Advanced Encryption Standard (AES [12]). These circuits are dedicated microprocessors for carrying out cryptographic operations, in order to transform a plain message, named *plaintext*, into a cipher message, named *ciphertext*, using a secret key. From a cryptanalysis point of view, even with the knowledge of the crypto-algorithm, the attacker is not able to rebuild the plaintext from the ciphertext, nor to find the secret key used for the encryption. However, the scan chain implemented in a crypto-processor can leak the secret key. In literature, several proposal of attacks have been studied on (1) block ciphers [13]–[16], (2) cryptography based on elliptic curves [17] and (3) stream ciphers [18]:

1. Block ciphers are made of an encryption function based on diffusion and confusion properties able to encrypt a plaintext block into a ciphertext block using a secret key. The decryption function performs the reverse operation to retrieve the plaintext block from the ciphertext. The same secret key is used for encryption and decryption, making the block ciphers belong to *symmetric ciphers*. The most used block cipher is AES composed of substitution and permutation transformations applied to a plaintext using a secret key. Several versions of the algorithms are possible in function of the secret key size: 128, 192 and 256 bits. The key size used for an AES cipher specifies the number of repetitions of transformations, called *rounds*. In the AES-128, 10 rounds transform a 128-bit plaintext into a 128-bit ciphertext. After these 10 rounds, an attacker is not able to retrieve the secret key from the knowledge of the pair plaintext/ciphertext. Below 10 rounds of AES transformations, the encryption is not completed. The secret key is not enough diffused and confused within the encrypted message that it is impossible to retrieve the secret key.

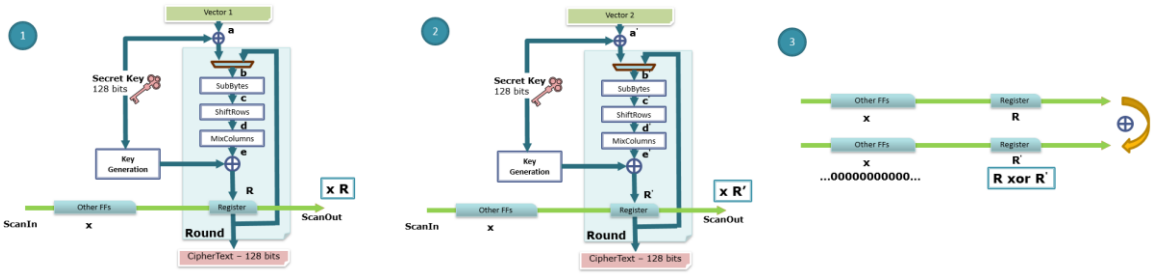


Figure 14: Differential scan attack on AES.

Yang et al. [13] apply scan attacks on AES crypto-processors when the round register belongs to the scan chain. The scan attack targets the first AES round, when the data are partially encrypted. To retrieve the secret key, the attack follows three steps, as depicted in Figure 14. Firstly, after a reset of the circuit, a plaintext is applied in functional mode during one clock cycle. The round register contains thus the result after the first round. In the second step, the attacker switches to test mode in order to collect the value of the round register. The third step consists in repeating the two previous steps for a second plaintext differing from the first plaintext by changing the least significant bit of any byte the first plaintext. The Hamming distance is calculated between the two intermediate results obtained. When the difference is equal to 9, 12, 23 or 24, a key byte is uniquely identified. Otherwise, the attacker has to try other plaintexts. In average, 32 trials are needed to retrieve one key byte. When one key byte is revealed, the attacker chooses pairs of plaintext that differ by another byte until all key bytes are identified. Overall, an attacker needs to apply on average 512 plaintexts to retrieve the 128-bit secret key (32 plaintexts for each byte). The same principle can be adapted to target different block ciphers. The same authors propose in [14] scan attacks on Data Encryption Standard (DES [19]).

The previous attack is only possible if the entire round register is accessible by scan chains. However, in the case of multiple scan chains, test compression techniques can be used such as Embedded Deterministic Test [6] or 2D Elastic Compression [7]. In this case, the 128 bits of the AES round register are observable in a compacted version. The previous attack [13] becomes inapplicable. Further scan attack has been proposed by Da Rolt et al. [15]. Instead of computing the Hamming distance between the 128 bits of the partial encrypted messages such as in [13], the principle in [15] is to study the parity of the partial encrypted results. This attack requires a preparation consisting in a simulation of the first round of the AES crypto-algorithm for chosen plaintexts. For each key byte, all the combinations are tested for chosen plaintexts, and the parity of the results after one round is saved into a database. The database is thus composed of 16 tables, one table for each key byte. The chosen plaintexts in simulation are applied to the circuit for one AES round, and the parity on the intermediate results are observed by shifting out the content of the round register. The parity on the partial encrypted results are then compared with the parity obtained in simulation for all the combination of key byte. After applying some plaintexts, one key byte comes out of the table analysis. In the case where all bits of the round register are observable, only 4 plaintexts are required to reveal a key byte. So, 64 plaintexts (4 times the 16 tables) are required to determine all the key bytes. In the case

where a response compaction is used, 24 plaintexts are needed to determine a key byte, i.e. 384 plaintexts for the entire 128-bit AES key.

The two previous scan attacks on AES ([13] and [15]) are only based on observation since plaintexts are always applied in functional mode. However, a simple countermeasure consists in forcing a reset of the round register at the switch to test mode. Ali et al. [16] circumvent this simple countermeasure by proposing a scan attack only in test mode. This scan attack uses both control and observation offered by the scan chains. Contrary to the previous scan attacks, plaintexts are applied by scanned-in bitstream. This attack implies therefore to figure out which scan FF corresponds to which input bit of AES, before trying to reveal the secret key. Authors propose to follow a protocol in order to determine the scan cells belonging to the round register. Once the AES inputs are identified, the same technique as in [15] is applied. Overall, this attack requires 375 test vectors to reveal the 128-bit AES key.

2. Scan attacks are also applied to Elliptic Curve Cryptography (ECC [20]) crypto-processors. ECC is an approach to public-key cryptography based on elliptic curves. Contrary to the symmetric cryptography where the encryption and decryption are performed with the same secret key, such as for block ciphers, the public-key cryptography is an *asymmetric cryptography*, working with a pair of keys. The encryption is performed by a public key, and the decryption by another key, named private key. This way, the owner of the private key can disseminate widely the public key to several persons. The receivers of the public key can encrypt messages, and only the owner of the private key can decrypt them. Concerning ECC cryptography, the encryption consists in several iterations where for each iteration a point multiplication on an elliptic curve is computed using the public key.

   Da Rolt et al. [17] present a scan attack on ECC. For targeted iteration, the attacker applies two distinct points and collects the parity of the results. The choice of the applied pair of points is done by simulation: the results parity is pre-calculated according to key value hypothesis. The simulation helps to find a pair of points resulting in different parity results according to key bit hypothesis. With the pair found by simulation, the attacker retrieves the key bit. This procedure is repeated for all iterations to steal the entire key. In practice for 192-bit ECC implementation, 8 points are required on average to find out the secret. In other words, all the pairs of points for each iteration are created from these 8 points.

3. Streams ciphers are also the target of scan attacks. Stream ciphers belong to the symmetric ciphers, such as the block ciphers. The same secret key is used for encryption and decryption. The encryption function is based on the XOR bitwise between the plaintext and a pseudo-random sequence of bits, called *keystream*. The decryption realizes the XOR operation between the ciphertext and the same keystream. The keystream is generated by a Pseudo Random Number Generator (PRNG) that takes as inputs the secret key and an Initialization Vector (IV). Some stream ciphers rely on LFSR as PRNG for the keystream generation, such as DECIM [21], Pomaranch [22], A5/1 and A5/2 [23], W7 [24], and LILI II [25]

   Liu et al [18] present an attack to apply on stream ciphers relying on LFSR. The principle is to let the LFSR running during several functional clocks, then to switch to test mode to collect LFSR content. After collecting results at different times, algorithms developed by the authors bring

to light relations between internal states of LFSR. The attacker is thus able to discover the LFSR structure. The number of scan-out vectors required is less than 100 for all the ciphers attacked in this paper [21]–[25].

The only condition to perform the scan attacks is that the external attacker has the access to the scan chains of the targeted crypto-processors. The common attack scenario is on a board implementing a JTAG interface. The attacker has just to send an *INTEST* instruction in order to access to the scan chains within the circuit. The JTAG interface offers even more possibilities to carry out attacks.

II.1.a.ii Intellectual property theft

A JTAG feature is the configuration of FPGA. The configuration is done by sending a file called bitstream at the FPGA power-up. This file contains confidential information about IPs used in the FPGA design. Altera [26] brings out the threat of bitstream theft. The FPGA configuration is done by sending bitstream by JTAG port, or the configuration file is contained in a memory. All that the attacker needs is an access to JTAG: either he intercepts bitstream sent by JTAG, or he reads the FPGA memory containing bitstream by using JTAG interface. After stealing the configuration file, the attacker can do reverse-engineering on the decrypted file to steal IPs.

Altera countermeasure is to encrypt the bitstream. Reverse-engineering on bitstream is not possible anymore for an attacker without the knowledge of the encryption key. The user programs the FPGA with a chosen key. The key is securely stored in a tamper-resistant memory. User configures FPGA with encrypted bitstream. When user wants to configure the FPGA, an implemented AES crypto-module decrypts the bitstream.

II.1.a.iii Updating a corrupted firmware

JTAG interface is also used to upload firmware updates, as depicted in Figure 13 with the device 4. This feature brings security issues with the update of corrupted firmware by an attacker. An example is the firmware corruption of set-top box [27]. A set-top box allows seeing subscribed TV channels. The TV video stream is encrypted by asymmetric cryptography, the decoder decrypts video stream according to the user subscription. The private key is stored in a smartcard inserted into the decoder. The public key is shared between set-top box and TV provider.

The attack allows getting all TV channels without subscribing paid service. Using JTAG interface, an attacker is able to read the firmware managing decryption TV stream in the memory of the microcontroller. He/she can steal information closely related to the private key. He/she can then replace the smartcard by a microcontroller configured with the stolen private key and with an unlimited access to the TV channels. He/she also replaces the actual firmware with a corrupted one including the authorization to access all TV channels. With the private key in possession of the attacker, TV provider does not detect the corrupted firmware.

TV providers have taken a countermeasure against this attack. In the video stream, a command is sent to the decoder in order to detect illegal devices connected to the set-top box such as a

microcontroller replacing the smartcard. If an illegal device is detected, new keys are used for TV video stream decryption and the decoder becomes inoperative.

II.1.a.iv Exploiting the debug facilities

An important feature of JTAG interface is debugging. A developer of an application can read and write into registers, to dump memory, and to set breakpoints on a program, using the JTAG interface. However, a malicious user can exploit the debugging features to find security flaw on a secure system.

A first example is with first iPhone OS, named iOS. The attacker aims to eliminate restriction and security of Apple OS imposed to the user. This kind of attack is called *jailbreak*. In particular, George Hotz wanted to use a different SIM card from AT&T telephone operator. This operator had the iPhone exclusivity on American telecommunications network. Hotz relates the attack steps on his blog [28]. After phone disassembly, he was able to identify JTAG interface in the baseband module. Baseband module controls telecommunications on the phone. Using the debug offered by the JTAG interface, the author read in memory the code starting communication services. The code indicates that AT&T SIM card is checked during the phone boot, and then only the presence of a SIM is checked every 5 seconds. Hotz proceeded as follows: (i) after phone boot, a halt command is sent to the processor through JTAG interface, (ii) AT&T SIM card is exchanged with one from another telephone operator, and (iii) processor exits halted state. Thereby, after a start-up with AT&T SIM card, the phone works with another operator SIM card. The JTAG interface helped to identify the security flaw in the code starting communication services.

Debug on Xbox 360 game console also brings to an attack described in [29]. Microsoft DRM protects video games by encryption of their content. Before the video game execution, console decrypts and executes the game content in a hypervisor mode. Memory space is divided into two regions: one for unprivileged code, one for privileged code containing video game code. From the user mode, privileged code is read-only and encrypted. The user executes a privileged code by calling the hypervisor function. Before executing the program sent by the user, the function checks if the program address is in the secure memory region. The JTAG interface allows dumping the memory of the game console in order to obtain the assembly code of the hypervisor function call. Some instructions lines of this function are presented below:

```
13D8 : cmplwi   R0,0x61
13DC : bge      illegal_syscall
…
13F0 : rldicr   R1,R0,2,61
13F4 : lwz      R4,syscall_table(R1)
13F8 : mtlr     R4
…
1414 : blrl
```

These few lines realize the following operations:

1. The R0 register contains the program address that the user wants to be executed as privileged code. The first instruction *cmplwi* (Compare Logical Low Immediate) compare the 32 least significant bits of R0 to the value 0x61.
2. If the comparison result is superior or equal, the function “illegal_syscall” is called. The role of the two first instruction lines is to verify that the program address (contained in R0) belongs to the secure memory region.
3. After verification, the rest of the code performs the program call. Firstly, an operation *rldicr* (Rotate Left Double Word Immediate then Clear Right) is performed on the 64 bits of R0 and the result is saved to the register R1.
4. The next instruction *lwz* (Load Word and Zero) loads the program address into the register R4.
5. The *mtlr* (Move to Link Register) instruction writes R4 content into the Link Register.
6. Finally, the branch instruction *blrl* returns to the Link Register address resulting program execution in hypervisor mode.

The hypervisor function call presents a flaw that can be exploited by an attacker to perform a privileged escalation attack. Address checking is done incorrectly: only 32 bits of register R0 are checked in *cmplwi* instruction. However, following instructions operates on the 64 bits of R0. This way, an attacker is able to bypass the address checking in order to execute an unprivileged code in hypervisor mode. The attacker can thus execute a video game without considering the Microsoft DRM protection. As countermeasure, Microsoft updates Xbox 360 to patch this exploit by correcting the code of hypervisor function call.

II.1.a.v Exploiting the IJTAG network

Another threat from the use of the test infrastructures by an external attacker is the access to the IJTAG reconfigurable network connected to hundreds of embedded instruments. As depicted in Figure 13 with device 3, these instruments have different functions: some instruments store confidential data (e.g. SoC configuration), some other instruments are dedicated for in-field testing (e.g. BIST engines). These two types of instruments are a target from an external attacker. Firstly, an attacker needs to figure out the architecture of the IJTAG network in order to find which SIB(s) to open. Once the IJTAG network revealed, the attacker can access to the targeted TDR(s) associated to the security-critical instruments. Either the attacker reads out the TDR for instruments storing confidential data, or the attacker exploits the in-field test instruments to overheat the circuit. Indeed, BIST engines provide the testing of specific modules (e.g. memories) in a SoC, and require a high amount of power. Therefore, the BIST engines can be activated at the same time as the circuit operates in order to perform in-field testing. Nevertheless, the activation of the BIST engines has to be well scheduled in order to avoid overheating. An attacker accessing the TDRs of the BIST engines can bring about the overheating making the circuit inoperative.

II.1.b Internal attacker

External attacker has not the only threat using the test infrastructures. The attacker can also be within the chip, such as the insertion of a malicious component in the test daisy-chain or counterfeit chips due to overproduction by an untrusted foundry.

II.1.b.i Insertion of a malicious component in the test daisy-chain

Test infrastructures connect together test wrappers components: on a board for JTAG standard, and on a SoC for IEEE 1500 standard. The common connection between these test components is the daisy-chain. This way, the data shifting in and out the test daisy-chain pass through every circuits belonging to the chain. A SoC designer often uses third-party component whose the detailed architecture is unknown from the designer. If a third-party component inserted in the test daisy-chain integrate a malicious part within its architecture, it becomes an internal threat for the other circuits in the system, such as the device 2 in Figure 13.

Rosenfeld et al. [30] present this internal threat with a malicious device within a JTAG daisy-chain. The malicious component can observe or control other components in the system, as an untrusted user accessing the test interface. The attacker can either sniff the test communication in order to steal confidential data from a victim device, or tamper the test data sent by a victim device in order to give wrong feedback to the tester.

An attack scenario is for malicious devices set to BYPASS mode. Usually, when the tester wants to send data to a target device, he/she firstly sends a specific instruction to the target device, and the *BYPASS* instruction to the others. A malicious device can be designed in order to store a copy of the data that are shifted through the BYP register. These data are then sent to the attacker entity in order to retrieve the confidential data (e.g. secret keys, configuration file, and firmware). A possible way to send the data is to send them to a remote server.

Another attack is for a circuit designed to modify the test responses of another circuit shifted through its BYP register. This way, the tester receives fake test responses from the victim circuit. In view of the obtained responses, the tester considers that the tested device presents defects, even if in fact the device can be without defects.

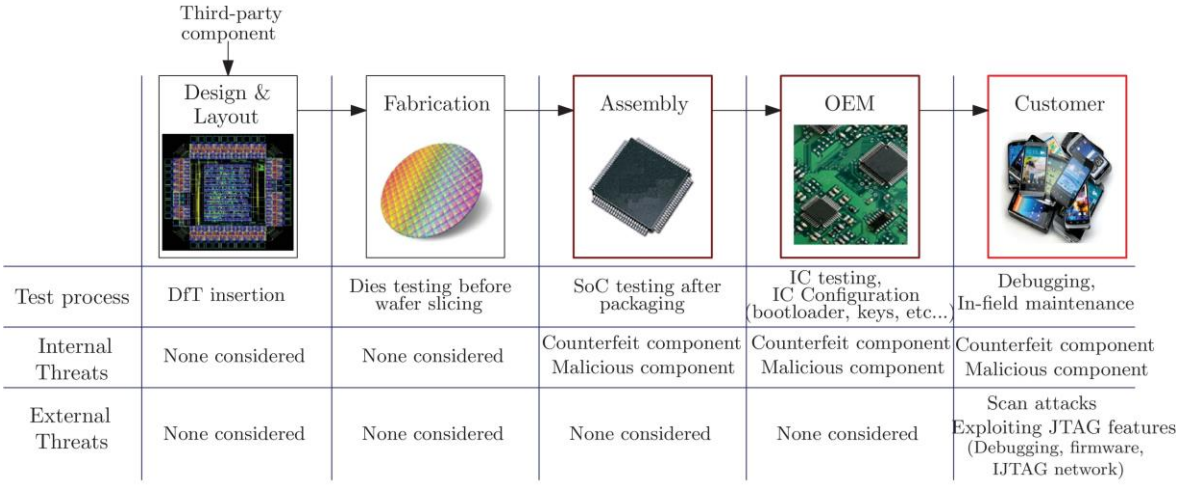
II.1.b.ii Counterfeit components

Another internal threat comes from the overproduction of untrusted foundry in order to produce counterfeit components. Counterfeit represents an important challenge in the electronics industry [31]. A counterfeit component is an unauthorized copy, that does not fulfill the specifications of the original circuit, and that is distributed in violation of the intellectual property rights of the original manufacturer. A SoC designer can integrate a counterfeit component in the design without knowing that the component is not the original one. This may result to a decrease of the performance of the whole SoC due to the counterfeit component. For this reason, the SoC designer may want to identify the counterfeit components possible implemented. A counterfeit component can also receive new version of the firmware from the original manufacturer. The original manufacturer may want to avoid uploading new version to a counterfeit. A counterfeit component may also embed a malicious circuit

in addition of the copy of the original circuit. This malicious component can be integrated into the test daisy-chain, able to sniff and tamper the test communication, such as the other internal threat presented in this section.

II.2. Threats throughout IC supply chain

The various threats can be present at every stage of the IC supply chain. Figure 15 presents the different actors in the supply chain and the threats that they represent, based on the testing procedures that are performed at each stage.



Test process	DfT insertion	Dies testing before wafer slicing	SoC testing after packaging	IC testing, IC Configuration (bootloader, keys, etc...)	Debugging, In-field maintenance
Internal Threats	None considered	None considered	Counterfeit component Malicious component	Counterfeit component Malicious component	Counterfeit component Malicious component
External Threats	None considered	None considered	None considered	None considered	Scan attacks Exploiting JTAG features (Debugging, firmware, IJTAG network)

Figure 15: IC supply chain with the threats of the different actors using the test facilities.

The first actor is the IC designer who is in charge of inserting the DfT components into the design, such as the scan chains and the test standard interfaces. According to the threat model defined in Section II.1, no attack is possible at this level since the circuit is still at the layout stage. However, third-party component can be inserted into the design at this stage. These components can come from an untrusted source, presenting a future threat if it is inserted in the test network.

After the design and layout stage, the circuit is sent to a foundry in order to produce several samples. Before wafer slicing, all the dies are tested independently or in parallel, depending on the ATE at disposal. No threat is considered at this stage. An external attacker cannot carry attacks from this stage. JTAG wrapper are not inserted yet within the circuits. Additionally, scan attacks on crypto-cores would be useless at this stage, because an attacker could only steal the key used during the manufacturing test, since the key used in mission mode is not configured yet.

The circuit is then packaged, during the assembly stage. When the SoC is mounted on the board, the considered threats are a malicious core inserted in the scan network, as well as the counterfeit component due to overproduction from an untrusted foundry. Concerning scan attacks, the key of the crypto-core is still the test key, not representing a meaningful secret to steal.

The next stage consists in mounting the IC on the board by the OEM. The OEM performs the test of the IC and sets its configuration. The configuration goes through loading the bootloader and the firmware in the memory, configuring the crypto-processor with the key used in mission mode and

configuring the microprocessors. The threats at this stage are the same as during the assembly stage. The secret key of the crypto-core and the configuration files (bootloader, firmware) are configured by the OEM, thus not representing a threat at this stage.

The final stage is when the final user owns the device. The test interface of the device can be used to debug the system or perform in-field maintenance. All the studied attacks represent a threat at this stage. A malicious user can exploit the scan chain to steal the secret key configured by the OEM, and exploit the JTAG features such as debugging, uploading corrupted firmware, or accessing the IJTAG network.

II.3. Attack report on the reference board of TEEVA project

As seen in this chapter, the test interface presents several threats for secure circuits. This thesis in the TEEVA project aims to identify specifically the threats on board operating with TEE. Considering the threat model, the security analysis using the test infrastructures has been conducted on the reference board of TEEVA project. In this section, we present firstly the discovered attacks on TEE in literature, then the reference board operating with TEE. Finally, we evaluate the security using the test infrastructures of the reference board.

II.2.a Discovered attacks on TEE in the literature

No discovered attacks on a device operating with TEE exploit directly the test infrastructures. However, attackers in [32]–[34] found vulnerabilities in the TEE by exploiting flaws in the executed code. In order to obtain the program code, one way is to dump the memory using the debugging features offered by JTAG to get binary files. Once the binary files obtained, the attacker can use a code reverse-engineering tool in order to exploit the potential flaw in the code.

Attack described in [32] targets HTC mobile phones and it exploits a flaw in the function copying data in memory. This attack looks like the attack described for the Xbox 360 game console in [29]. An attacker is able to copy data from Secure world to Non-Secure world due to an incorrect address memory checking in the function copying data. An attacker can thus exploit this flaw to export out all data contained in the Secure world, i.e. all the code related to the TEE.

In [33], the Secure Monitor Call, the function switching between Non-Secure world and Secure world, present a vulnerability for all Qualcomm-based Motorola Android phones. In these devices, a user in Non-Secure world is able to execute program contained in a specific Secure region of memory, but write operations are prohibitive on the Secure memory corresponding to the region where is saved the TEE code. The specific command of the Secure Monitor to execute TEE program from the Non-Secure world works by copying the read data in Secure region to a Non-Secure region of the memory in order to execute it after the copy. Two arguments are passed to the Secure Monitor: the first argument is the address to be read in Secure world, the second argument is the address where the value is copied. The flaw comes from the second argument; the code does not check that the provided address corresponds to a Non-secure region in memory. Therefore, the attacker is able to write the read value in a chosen address in Secure region. This way, the flaw is exploited in [33] to overwrite

security flags such as bootloader locking. The attacker can thus boot kernel and system images not signed by Motorola.

HTC and Motorola phones are not the only ones presenting flaws in their TEE code. In [34], an attack has been presented on Huawei phones. Attacker exploits vulnerabilities on HiSilicon TEE to get in the memory the fingerprint image used for authentication, normally not readable for a user in Non-Secure world.

For each mobile phone operating with TEE, the discovered vulnerabilities are no longer exploited. The TEE code has been updated with a patch to correct the security flaws.

II.2.b Overview of the reference board operating with TEE

The security evaluation has been performed on the reference board of the TEEVA project, HIKEY [35] board operating with the Trustonic TEE in parallel to the Android OS. The board implements the DfD offered by ARM, described in Chapter I. The experiments have been conducted thanks to the debugging materials of Lauterbach. The debugger hardware is the Power Debug USB 2 connecting the JTAG interface of the board to the USB of the computer. The user interface on the computer is given by the debugger software TRACE32.

The Lauterbach debugging materials allow detecting the debug accesses implemented on the HIKEY board among the potential accesses (system bus, debug bus, JTAG wrappers) offered by the debug architecture of ARM. Firstly, the debugger identifies the JTAG–DP DAP with its identification code (IDCode), as shown in Figure 16.

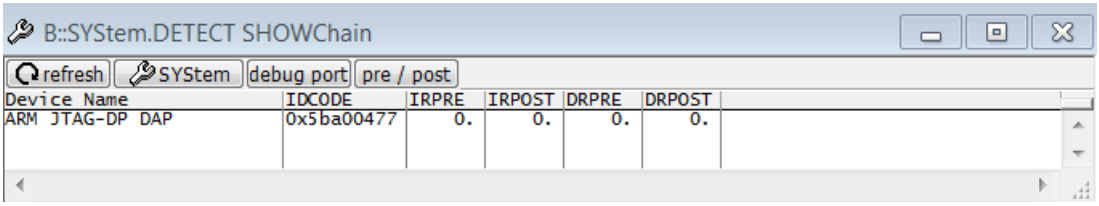


Figure 16: Detection with TRACE32 of the JTAG–Debug Port (DP) implemented on the HIKEY board.

Secondly, the debugger identifies the APs available through the DAP interface. As shown in Figure 17, two debug accesses are identified: the first one (AP#0) corresponds to the access port for the debug APB bus, the second one (AP#1) corresponds to the access port for the system bus AHB. A third access (AP#2) is not identified, recognized as unknown with an identification code equal to 0x23000000. This AP does not match with the JTAG–AP connected to the JTAG wrappers, which has normally 0x10 on the Least Significant Bits of the identification code, according to the ARM specifications [11].

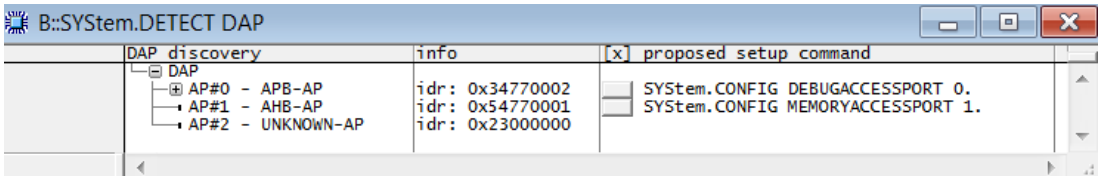


Figure 17: Detection with TRACE32 of the Access Ports (APs) implemented on the HIKEY board.

The DfD implemented on the HIKEY board allows debugging the system through the ARM buses (either AHB or debug APB) by setting breakpoints on the executed code, by dumping and modifying the memory, and by reading the processors registers. The HIKEY board being a development board,

the enable signals for the debugging functions on the Secure world are hardwired to be always enabled. This way, a developer of a TA can debug his application uploaded on the reference board running in the TEE.

II.2.c Security analysis through JTAG access on the reference board

The discovered attacks on TEE are based on software flaws in some critical program of the TEE. Therefore, these attacks target specific devices presenting the identified flaw. In addition, when a flaw is identified, the mobile manufacturer patches the TEE code with an update in order to secure its devices.

Contrarily to these attacks targeting software flaws in specific TEE implementation, we analyze the security through the test infrastructures in a general TEE implementation. A main threat from an external attacker using the test interface is the scan attacks, such as described in Section II.1. However, the access to the scan chains possible with the JTAG wrappers are not implemented in the reference board of the TEEVA project, or at least there is no connection with this access port. Since the scan chains are required for post-manufacturing testing, it is very likely that the connections to the JTAG wrappers (and thus to the scan chains) are disconnected from the external JTAG interface after manufacturing testing. It is a simple countermeasure to prevent against scan attacks. Disconnecting the test accesses after manufacturing is the most common industrial practice. An issue with this technique is in-field debug and diagnosis offered by the scan chains are not possible anymore, implying maintenance issues during the IC life. Moreover, this countermeasure still presents a security threat from probing techniques. Indeed, an attacker able to reconnect the access to the scan chains can then perform scan attacks in order to steal secrets within the circuit. The probing attacks require nevertheless specialized lab materials in order to identify the disconnected signals and to probe them. These types of attacks are out of the scope in this project, and have not been conducted on the reference board.

Assuming the probing on the scan chains of the reference board, an attacker could scan out the content of the processor's registers during a TA execution. Considering a TA executing an AES encryption, the attacker could target the execution of the first AES round after identifying the first round among the whole TA execution. The first round could be identified either by code reverse-engineering of the TA, or by analyzing the power consumption/electromagnetic emission on the reference board during the TA execution. Once the first round identified, the attacker could launch the TA and he/she could switch the circuit from functional mode to test mode after the first AES round. In test mode, the attacker could scan out the content of the processor's registers, and he/she could thus carry out the scan attack.

This attack scenario shows that scan attacks are a threat for boards operating with TEE, such as for every circuits embedded a secret. A countermeasure needs to be implemented. The reference board disconnects of the test accesses, as countermeasure. This simple solution prevents against scan attacks, but it still represents a risk against an attacker able to reconnect the scan chain accesses.

II.4. Conclusion on attacks using test infrastructures

Testing is essential to ensure the product quality, as well as in-field debugging and diagnosis in order to provide feedback information during IC life, such as hardware failure or software error. Test infrastructures are thus mandatory to deals with complex systems on board and on chip. However, as detailed in this chapter, attacks are possible exploiting the test interface by considering both external attackers as well as internal attackers.

For instance, devices implementing cryptographic primitives can be the target of scan attacks, where the content of the scan chains are observed in order to retrieve data strongly correlated to the secret key, thus allowing discovering the key itself. In addition to the test functions provided by the scan chains, test interface includes additional features, such as debugging, updating firmware or configuring FPGA. A malicious user can exploit these features to debug illegally the system, to update a corrupted firmware, and to steal IP contained in the FPGA bitstream. Considering internal attacker, the daisy-chain fashion in test infrastructures is also a security threat. Indeed, a malicious component is able to sniff and to tamper test data shifting through its architecture.

For these reasons, every ICs where the security is critical, such as devices operating with TEE, need to implement countermeasures. A simple countermeasure, implemented in the project reference board, consists in disconnecting the test accesses in order to avoid any attack through test infrastructures. However, an attacker able to reconnect the test accesses can easily circumvent this countermeasure. Moreover, test, diagnosis and debug are not possible anymore during IC life. Further countermeasures have been developed in the state-of-the-art to ensure security on the test infrastructures without affecting the test, diagnosis and debug facilities.

Chapter III

Existing countermeasures against attacks using test interfaces

Summary

III.1. Classification of the existing countermeasures

In this chapter, we survey countermeasures that have been proposed in literature and in industry. As depicted in Figure 18, we classify them in four main groups: scan chain avoidance, secure scan designs, secure test access and detection of illegitimate behavior. *Scan chain avoidance* ensures the protection by limiting or even disconnecting the test accesses. *Secure scan designs* ensure protections by modifying the scan network, either on scan chains or on RSN defined in IJTAG standard. *Secure test access* ensures a secure protocol to access the test interface using a password, a challenge/response or the encryption of the test communication. *Solutions detecting illegitimate behavior* analyze the test operations in order to recognize the use of the test interface by an attacker.

Figure 18: Classification of the countermeasures against the attacks using the test interfaces.

According to the attacks presented in Chapter II, each countermeasure is analyzed with respect to a protection against:

(i) an attacker carrying out scan attacks [13]–[18],

(ii) an attacker exploiting JTAG features (e.g. stealing IP design [26], updating a corrupted firmware [27], exploiting debugging facilities [28][29]),

(iii) users with a restricted access to the test interface (e.g. fine-grained access to specific instructions, to specific memory areas, to specific cores, and to specific instruments in IJTAG network),

(iv) an attacker able to probe the test signals within the circuit, requiring high resources to locate and probe precisely the test signals on the layout of the circuit,

(v) a malicious component integrated in the test daisy-chain [30],

(vi) counterfeit components due to an overproduction from an untrusted foundry [31].

The first four points represent the threats from an external attacker, while the two last points the threats from an internal attacker. We evaluate the existing countermeasures against these six types of attacks.

In the evaluation of the state-of-the-art countermeasures, we also consider the costs in terms of testability and applicability. Countermeasures applied to the test infrastructures can have an impact on the testability, considering the additional test time due to the protection, the potential diminution of the fault coverage, the potential limitation on debug and diagnosis features, and the modification of the test procedure. In addition to the costs on testability, the integration of the countermeasures into an original circuit implies area and power consumption overheads, modification of the insertion flow of DfT, applicability on non-modifiable circuit, and sharing method of the possible embedded secret in the circuit to the authorized tester.

We provide in this chapter a presentation of each countermeasure in the stat-of-the-art regarding the classification into the four main groups (avoiding scan chains, secure scan design, secure test access, and detection of illegitimate behavior). Moreover, we evaluate the existing solutions regarding the ensured security and the costs on testability and applicability.

III.2. Scan chain avoidance

The first class of countermeasures, named scan chain avoidance, limits or even the test accesses to the scan chains. Two solutions are presented thereafter.

1. *Blowing fuses* is the most common solution in industry, consisting in disconnecting the test interface after manufacturing. For instance, this solution is implemented in the reference board in the TEEVA project described in Chapter II, preventing the threats using the test interface. The main issues are that the maintenance in the field is not possible anymore, and this solution is vulnerable against an attacker able to probe on the disconnected test access.
2. *Built-In Self-Test*, commonly named as BIST [36], is an alternative DfT technique to the scan chains, limiting the test interactions between the tester and the circuit. A test pattern generator based on PRNG is used to generate the test vectors. An output response analyzer generates a signature from the test responses. Tester has limited control and observation on the scan chains: he/she shifts the seed of the PRNG at SI in order to launch the test pattern generator, and he/she collects test signatures at SO in order to compare with expected ones.

   BIST prevents an attacker to apply chosen test patterns and to observe the whole test responses. In the same way, a malicious component in the test daisy-chain can only sniff the test signatures. The security level ensured by BIST technique depends on the level of control on the test sequence generation (e.g. dynamic or static seed for the PRNG, control on the generated sequence size) and the level of the test responses compaction (e.g. spatial or

temporal signature generation). Less control and observation are offered to the external user, more BIST technique is secure against scan attacks.

However, the opposite statement is applied for guaranteeing diagnosis. In order to perform diagnosis, the tester must be able to either control the BIST sequence applied (seed and sequence size), or to observe signature of specified group cells. In the first case, tester can apply shortcut test sequences in order to determine which test vector results a faulty response. In the other case, a mask on a group of scan cells is applied before signature generation, and thus tester can determine which group of cells is faulty. In all cases, diagnosis requires either more control or more observation, decreasing the security level ensured by the solution. Concerning fault coverage, an analysis is performed by the ATPG in order to determine the best seed covering a maximum of faults in a minimum test time. However, it is not always possible to find a sequence achieving the same fault coverage achieved than classical scan design. The implementation of the test pattern generation and the output response analyzer also implies an area and power consumption costs. These costs can be reduced when BIST is implemented on crypto-processors, such as an AES block cipher. The test of circuits implementing cryptographic algorithms is effective even by using a short pseudo-random sequence. Indeed, possible errors are easily propagated through typical operations involved in such encryption algorithms thanks to their confusion and diffusion properties [37] and [38]. Doulcier et al. [39] present a BIST implementation on AES crypto-core testing all stuck-at faults by executing 2600 rounds. Area cost represents an overhead of 507 GE corresponding to an increase of 3.31% compared to the original core. Compared to a JTAG controller of 180 GE, this cost represents an overhead of 282%.

Scan chain avoidance is the class of countermeasure limiting test, diagnosis and debug facilities, implying potential troubles. This way, other classes of countermeasure have been developed in literature to preserve test, diagnosis and debug.

III.3. Secure scan design

The second class defined as secure scan design gathers the countermeasures protecting the scan chains and the ones protecting the RSN defined in the IJTAG standard.

III.3.a On scan chains

The protection ensured on the scan chains implies some modifications on the scan architecture in order to insert some components securing the scan chains. We present thereafter the proposed solutions in literature, and evaluate the ensured security and the costs.

1. *Mirror-Key Register* [13] solution aims to protect crypto-processors against scan attacks. During the functional mode of the crypto-processor, the scan-in and scan-out pins are nonfunctional, preventing to shift in and out the scan content. To switch to test mode, a global reset has to be applied. In test mode, the crypto-processor loads a test key from a mirror-register, different

from the secret key used in functional mode. If an attacker tries to carry out scan attacks in test mode, he is only able to retrieve the test key, not compromising the security of the system.

This solution aims to protect only against scan attacks on crypto-processors. Other threats are not considered.

Considering the impact on testability, the countermeasure preserves test, diagnosis and debug without additional test time, and it does not reduce the test coverage of the original circuit. However, the added mirror-register containing the secret key of the crypto-processor cannot be tested, without compromising the security. In-field debug is still possible with this solution, but it requires a global reset of the system before any debugging operations. Concerning the applicability of the solution, the integration in the DfT flow involves some modification on the FSM managing the test operations in order to support the mirror-key register architecture. Authors evaluate the area cost of the solution on an AES crypto-processor to 412 Gate Equivalent (GE), representing an overhead of 229% compared to a JTAG controller of 180 GE.

2. *Obfuscation* on scan design [40] consists in replacing the original scan FFs forming a simple shift register into a complex LFSR. The secret shared with the tester is the combinational and sequential function of the LFSR. Authors study LFSR structure in order to build a secure architecture difficult to identify for an attacker. Authors state that adding inverters and at most one dummy FF within the scan design helps to make the LFSR structure difficult to identify. The tester has thus to be aware of the specific hidden procedure and test data has to be processed before being compared to expected data.

   This solution is based on the assumption that the attacker has no way to get the information on the scan chain's implementation, preventing against scan attacks. However, such 'security-by-obscurity' approach goes against Kerckhoffs' doctrine and is not considered as strong.

   Test, debug and diagnosis are preserved without constrains. However, the secure LFSR structure proposed by the authors requires the insertion of dummy FFs in the scan chain, implying a test time and area costs. These costs depend on the number of added FFs and XOR gates in the scan design. No experimentations have been conducted to evaluate the overheads. The LFSR insertion within the scan design also requires an important post treatment on netlist file after regular scan insertion by the synthesis tool, significantly impacting the DfT flow.

3. *Scan Enable Tree* [41] propose to check the Scan Enable (SE) integrity in order to control the scan chain activity. Indeed, the SE signal, which drives each scan FF, is built up into a buffers tree during design synthesis. This tree is driven by the SE signal issued from the test interface and tree branches drive different portions of the scan path. The proposed protection consists in checking that the SE signal is not enabled, while scan operations are not allowed. If an illegal shift operation occurs, a controller detects it thanks to the tree configuration, and resets thus the whole system.

   This countermeasure aims to protect only against probing, others threats are out-of-the-scope in this solution. The protection prevents an attacker probing the SE of specific FFs to shift out their content. Since this solution considers an attacker able to probe specific signals on the circuit, the level of protection relies on the reset efficiency. Indeed, in addition to probe on SE

signal, the attacker could also interfere with the reset signal in order to counteract the action of the controller. Therefore, the controller has to perform a "robust" reset in order to reset the system even under stress conditions.

Concerning the costs to implement this solution, experimentations have been conducted on a DES crypto-processor with the implementation of a scan enable tree composed of 8 branches. Some additional patterns are needed to test the tree controller, introducing a test time overhead of 1% compared to the original test procedure. The implementation of the security controller of the scan enable tree represents an area overhead of 36% compared to a JTAG controller of 180 GE. Authors indicate without giving numerical results that the solution induces no significant power consumption increase. Concerning the regular DfT flow, SE signals are connected automatically to the scan FFs during design synthesis. The solution is thus performed after design synthesis, implying efforts on the DfT flow to integrate the solution. The designer has to modify the netlist file to connect manually the SE branches of the tree to the controller.

4. *Spy cell* [42] is another countermeasure aimed to protect against probing on SE signals, consisting in the insertion of "spy cells" within the san chain. A spy cell is a scan FF, which has its functional input set to a fixed value. Reading an opposite value on the FF output indicates the activation of the scan operations. A controller detects if an illegal shift operation occurs, and applies a global reset to the circuit to counteract the probe attack.

   As the previous countermeasure, this solution protects only against probing attacks, and the security level depends on the reset efficiency.

   The costs have been evaluated on a DES crypto-processor with the insertion of 6 spy cells within the scan chain. This way, the test time increases by 5% due to extra patterns needed to test the spy cell controller, and due to extra shifting operations introduced by the added scan FFs. The 6 spy cells and their controller results to an area cost of 27% compared to a JTAG controller of 180 GE. Concerning the power consumption, authors indicate a marginal increase compared to the power consumption of the original DES circuit. Concerning the DfT flow, the designer has to do an effort to adapt the synthesis command files in order to insert the spy cells.

5. *Scrambling* [43] is another technique against probing attacks, consisting in changing dynamically and randomly the order of the scan FFs in the scan chain. The scan path is divided into segments, dynamically re-ordered during functional mode. While in test mode, the scan path is in a fixed configuration in order to perform the test procedure.

   Attackers probing signals on the scan chain during the functional mode of the circuit is not able to analyze the scanned-out data since the order of scan FFs changes randomly. Concerning the observation of the scan content in test mode, an attacker without the knowledge of the scrambling order is not able to reorder the scan output bitstream. However, the scrambling order is fixed in test mode. This protection is thus not sufficient to protect against scan attacks, since scan attacks are based on a differential analysis of the obtained test responses.

   Test time and test generation effort are not affected by scan chain re-ordering. Indeed, the segmentation of the scan path is transparent for ATPG tool, test is performed as if there are only one scan path. Debug and diagnosis are preserved, but a management policy is needed to

share the scrambling order in test mode to the authorized tester. Scrambling method also induces additional power consumption due to multiplexer commutations. Experimentations have been conducted on a DES crypto-processor with a scrambling between 6 segments of scan chain. The power consumption increases by 7% for a DES encryption. Area increases by 156% compared to a JTAG controller of 180 GE.

6. *Test key in scan chain* [44] is an authentication method consisting in shifting a test key within the scan chain. The test key is inserted into the test patterns sent to the circuit. Some scan FFs are added to the scan chain, and are dedicated to receive the test key. A controller checks the value contained in the specific scan FFs to control if the shifted test key is correct. A LFSR generates a pseudo-random sequence to alter the test responses when the wrong test key is shifted into the dedicated scan FFs.

   The threat model for this countermeasure includes only the scan attack performed by an external attacker. The level of security ensured by this solution relies on the key size, and on the randomness added to the response. An attacker can circumvent the protection by intercepting the test key, or by corrupting the pseudo-random sequence generated by the LFSR (e.g. corrupting the seed, stressing the device).

   Test, debug and diagnosis are preserved, but test time is affected by the additional scan FFs in the scan chain. Authors evaluate the test time cost on S35932 benchmark circuit for a test key on 80 bits, resulting to an increase of 5.6% compared to original test time. The area cost is evaluated on the same circuit, S35932 benchmark, but for a key on 10 bits, resulting to an increase of 0.8% compared to the original circuit area. The area evaluation does not correspond to a realistic implementation, since the security ensured by a key on 10 bits is very weak. This method requires establishing a key management policy in order to share the test key of the circuit to the authorized users. The insertion of the additional scan FFs also implies to modify the DfT flow. However, the authors have developed tools in order to automate the whole process.

7. *Secure comparator* [45] is based on the embedded comparison of the test responses inside the CUT. When the test responses are shifted out the scan chain of the CUT, the tester scan in the CUT at the same time the expected test responses using an additional test pin. A comparator implemented at the CUT scan output compares the actual test responses with the expected ones furnished by the tester. The tester can only observe the comparison result.

   The scan content is therefore not observable, preventing scan attacks. A malicious circuit inside the chip cannot also sniff the test communication since the comparator is implemented at CUT scan output. Others threats are not considered in this countermeasure.

   The implementation of the secure comparator is applied after the insertion of the scan chain. Therefore, the solution does not affect the standard DfT flow and can be applied on a non-modifiable component. However, the test procedure is modified since the expected test patterns have to be shifted in the CUT. Moreover, a specific test sequence has to be applied in order to test the stuck-at faults within the comparator. Considering a circuit composed of a number $\#SFF$ of scan cells, the additional test sequence implies a test time overhead of

6. $(\#SFF + 1)$. The area cost is evaluated for a circuit with 32 scan chains of 10 000 FFs. The associated comparator is composed of 32 FFs, 98 combinational gates, 64 buffers and 14-bit counter. The main issue with this technique is due to the debug and diagnosis limitations. Since only the result of the comparison is observed, the identification of the faults within a circuit is more difficult. The diagnosis is possible by trying to compare on-chip the obtained test responses to the possible faulty ones, implying additional time to diagnosis.

The presented countermeasures deal with the protection on a simple scan design, the scan chain. Nevertheless, scan chain can be organized in a scan network defined in the IJTAG standard. The next section deals with the countermeasures proposed for securing RSN.

III.3.b On Reconfigurable Scan Network (IJTAG)

IJTAG scan network is a valuable tool for accessing on-chip instruments during test, diagnosis and debug. RSN must implement protections for security-critical instruments such for the BIST engines, potential source of overheating in the circuits, or for instruments containing secret information. For accessing the security-critical instruments, an attacker needs firstly to discover the RSN architecture in order to open specific SIBs before performing the attacks. Some countermeasures [46] and [47] aim to make difficult the exploration of the scan network for an attacker without any knowledge on the architecture. Another countermeasure [48] relied on an authentication protocol to access specific instruments.

1. *Locking Segment Insertion Bit* (LSIB) [46] is a modified SIB that can only be opened when pre-defined values, corresponding to a key, are present in particular bits in the chain. The security level resides in the key size and the number of LSIB inserted within the RSN. Indeed, an attacker without the knowledge of the architecture has to try the different combinations of the key in order to unlock the SIB and access the targeted instrument. The same authors have proposed in [47] to introduce trap bits in the scan network in order to reduce the effectiveness of brute force attacks. When a wrong value is written on a trap bit, the LSIB is totally locked and only a global reset can set the LSIB effective again.

   The considered threat model is an attacker exploiting the IJTAG network. These countermeasures are based on the assumption that the attacker has no way to get information on the scan network implementation However, this obfuscation approach is not considered as strong protection, according to the Kerckhoff's doctrine.

   Authors propose to protect instruments with a 48-bit key. Test time cost represents thus 48 additional clock cycles for each LSIB implemented. Concerning area cost, it is evaluated in [48] on the t512505 benchmark to an overhead from 0.07% for protecting one instrument with a 48-bit key to 16.2% for protecting 256 instruments. An issue not mentioned by the authors concerns the sharing of the key required to unlock the protected instruments to the authorized users.

2. *Authentication protocol to access instruments* [48] is a method consisting in sending a challenge/response in order to unlock SIB associated to security-critical instruments. The challenge/response is based on keyed-hashes: (i) device sends a random value as challenge, (ii)

user responds with the hash of the challenge and the secrets of each protected instruments wanting to unlock, and (iii) same operation is done by the device to compare with the received response. The random value used as challenge avoids replay attacks on the authentication protocol.

This solution protects against an attacker exploiting the IJTAG network. This countermeasure does not protect against other threats.

The authentication module requires the implementation of a TRNG and a hash engine. The chosen hash engine is SHA-3. Concerning TRNG implementation, authors propose to re-use a TRNG available on the chip if possible. Otherwise, a very cost-efficient implementation is preferred such as TRNG based on ring oscillators. Area cost is evaluated on the t512505 benchmark. Protecting 1 instrument represents an overhead of 2.8%, while protecting 256 instruments represents an overhead of 4.9%. Compared to the countermeasures [46] and [47], this solution is more dedicated to protect several instruments. Test time is increased by the number of cycles required to communicate the challenge/response, and to open authorized instruments, representing the number of protected SIB and 532 clock cycles. Concerning the time to compute the hash function, authors assume that the hash engine offers sufficient throughput to compute the hash computation during the shift operations. This assumption are verified during authors' experimentations. The choice for key management and the potential use of a server storing the secrets is left to the designer.

The presented secure scan designs on IJTAG network protect specifically against an attacker exploiting the RSN, while the secure scan designs applied on scan chains protect either against scan attacks, or against probe attacks. To protect against the other considered threats, such as the exploitation of the JTAG features (e.g. debugging, uploading of firmware) and the counterfeit and malicious devices potentially inserted within the circuit, the countermeasure has to be coupled with another class of countermeasure, named secure test access.

III.4. Secure test access

Secure test access offer protections against another threat model compared to secure scan designs countermeasure. Moreover, this class of countermeasures deals with some troubles considering the modification of the scan structure. Indeed, if a third-party IP, legacy core or hard core is inserted in a design, the designer cannot modify the scan design to secure them. Even when the scan chains of the circuit are modifiable, the proposed solutions require usually efforts in the DfT integration with the modification of the netlist file or the synthesis command. A designer can thus privilege secure test access. Nevertheless, secure test access countermeasures also imply a tradeoff between several aspects: security level, method used for authentication, area cost, test time cost, and key management policy. These solutions do not reduce the fault coverage on the original circuit. However, the additional secure elements introduce new faults into the design needing to be tested.

Two different authentication methods have been proposed in the literature to ensure a secure test access, using either a password, or a challenge/response protocol. In addition to the authentication

methods, the encryption of the test communication has also been studied. We present thereafter both authentication methods and the encryption of the test communication, applied on the test standards (JTAG, IEEE 1500 and IJTAG).

III.4.a Password

The secure test access based on password allows protecting against a malicious use of the test interface, while preserving the test features for authorized users knowing the password.

1. *Password on JTAG standard* has been proposed in [49] to allow only users knowing the password to execute the JTAG instructions set. Two additional instructions, named *LOCK* and *UNLOCK*, ensure the lock and unlock of the TAP interface. When JTAG wrapper is locked, every instructions sent to the IR are decoded as *BYPASS* instruction (except the *UNLOCK* instruction). An authorized user has to shift the password after sending the *UNLOCK* instruction in order to execute the JTAG instructions. The password shifted during the *UNLOCK* instruction is compared with the one defined during the *LOCK* instruction.

   The user authentication with password on the JTAG interface ensures the protection against scan attacks and the exploitation of JTAG features.

   Concerning testability, the test of the added register is not specified, but test, debug and diagnosis capabilities are preserved. Nevertheless, before sending any test operations, the user needs to be authenticated, implying a test time overhead. Authors present experimentations with a password on 32 bits. Test time overhead represents 32 test clock cycles. The area cost is evaluated to 144% compared to a JTAG controller of 180 GE.

2. *Password on IEEE 1500 standard* [50] proposes a secure test wrapper on cores in a SoC. The secure test wrapper controller locks core's scan chains inputs and outputs by adding an AND gate with an *Unlock* signal set to '0' value until tester authentication. It also controls the instruction decoder: all instructions for the core are *BYPASS* until the tester authentication. The authentication consists in shifting a password. Each protected core has a unique password generated thanks to its boundary scan cells. Indeed, authors propose to modify the BSR into a LFSR. During the authentication phase, the LFSR seed is applied to the BSR during a number of clock cycles pre-defined by the designer.

   The considered threat model are the scan attacks, and the exploitation of the JTAG features. The protections against these threats models are limited to the protected cores.

   This solution requires some modification on IEEE 1500 standard. Authors choose an implementation of password on 256 bits. This cost represents an increase of 20% compared to a JTAG controller of 180 GE. Test time increases with 256 clock cycles required to shift the password. Authors propose a special test sequence to cover all stuck-at faults introduced by secure test wrappers.

The security level of the secure scan access using a password relies on the password size as well as the non-divulgation of the password to an attacker. The main issue with the password authentication is the share and the storage of the password to authorized users. The password has to be securely shared and stored, otherwise the security is compromised. However, authors in [49] and [50] do not

mention how to establish a secure key sharing. Concerning the password storage in [49], a register is dedicated to store the password, but no details are given about secure storage. In [50], the storage of the LFSR seed is not specified neither. Since the protection is based on the non-disclosure of the password, the password or the seed used to generate the password must be contained in a secure non-volatile memory. Another issue with the use of password is the *replay attacks*. In the case of an attacker able to sniff the password during the authentication of an authorized user, the attacker can send the same password to unlock the secure test access, circumventing the protection. For this reason, the protection is not a suitable against malicious component within the chip, able to read the password when it is shifted in the test daisy-chain.

III.4.b Challenge/Response

Secure test access using challenge/response protocol can ensure user authentication more secure. Indeed, randomness are added in the challenge/response protocol to avoid replay attacks. In addition to user authentication, some countermeasures also propose to authenticate the device in order to detect counterfeit components.

1. *Challenge/Response based on keyed-hashes* [51] propose to protect the JTAG instructions with a distinct key on each protected instruction. To use the protected instruction, tester has to send a challenge/response with the hashed secret key. The protocol is as follow: (i) when the tester sends a request to protected instruction, the device reads the associated key from a secure non-volatile memory and the device sends to the tester a random number, (ii) the tester generates a challenge based on a hashed message of the random number and the secret key, (iii) the device compute the same hash message, and (iv) if the comparison between the challenge and the hash computation, the device unlocks the protected instructions and notifies it to the tester. A protected instruction is unlocked until a reset or a power-off of the circuit. The hash computation is performed with the SHA-256 algorithm, and the random number is generated thanks to a True Random Number Generator (TRNG) based on rings oscillators.

   The considered threats with this countermeasure are an external attacker wanting to access the scan chains or other JTAG features. This solution proposes also a fine-grained access on the instructions. Indeed, since each protected instruction is unlocked using a dedicated secret key, a user can only be allowed to access a limited set of protected instructions.

   The area cost of TRNG, SHA-256 and secure memory is not specified by the authors, but Da Rolt et al. in [52] evaluate this solution to an overhead larger than 500% compared to a JTAG controller of 180 GE. Test, debug and diagnosis are preserved, but additional test time is required for the test operations. The test time overhead is due to the shifting of challenges and responses on 128 bits, and the SHA-256 computation. This test time cost cumulates for each unlocked instructions. Concerning the key management, authors indicate that the test/debug software delivered to the authorized users contains the keys to unlock the protected instructions.

2. *Challenge/Response based on symmetric cryptography* [53] present a user authentication with a challenge/response protocol using AES cryptography and SHA-1 hash function. The

authentication operates in two phases: (i) credential issue phase consisting in giving user accreditations after a verification of user identity by an online server, (ii) user authentication phase where the user unlocks the JTAG features using his/her accreditations obtained in the first phase. The challenge/response protocol establish a secure communication between the tester and the device thanks to a TRNG in order to have randomness in the challenge/response pairs, to a SHA-1 module in order to compute the hash of the secret key used for encrypting the communication, and to an AES crypto-processor for the encryption of the challenge/response communication.

When the JTAG interface is locked, all the instructions are decoded as *BYPASS* instruction, preventing an external attacker to carry out scan attacks, and to exploit JTAG features. Moreover, according to the accreditations given by the server to a user, the solution permits to have a fine-grained access to specific instructions.

Adding AES crypto-core, SHA-1 module, TRNG and an authentication controller implies an area cost of 2246% compared to a JTAG controller of 180 GE. Test time cost is composed of additional time to shift challenge/response, additional time to encrypt and additional time to compute hash. Concerning key management, an online server is only needed for the credential issue phase. Once this phase is achieved, authorized users can unlock JTAG offline. This method simplifies key management, requiring an online connection only one time.

3. *Challenge/Response based on asymmetric cryptography* [54] propose a user authentication using a challenge/response protocol based on ECC algorithm. The JTAG interface is locked, i.e. all instructions are decoded as *BYPASS*, until the authentication of the user. An authorized user is authenticated by a server owning the secret keys embedded in the protected devices and the user accreditations in a database. In addition to ECC crypto-core, a TRNG is implemented to include randomness in the challenge/response pairs in order to avoid replay attacks.

   The threat model considered in this countermeasure is against an external attacker wanting to carry out scan attack or exploiting the JTAG features. To ensure a sufficient security level, ECC operations are computed on 320 bits.

   The area cost to implement ECC crypto-processor and TRNG is evaluated in [52] to an overhead superior to 2000% compared to a JTAG controller of 180 GE. Authors do not specify the test time needed to authenticate the user before any test operations. Moreover, an online connection is required for each user authentication.

4. *Challenge/Response based on Schnorr protocol* [55] provide mutual authentication between user and device using challenge/response protocol based on an enhanced version of ECC, named Schnorr protocol. Authors consider a verifier wanting to authenticate a prover (the couple verifier/prover is either device/user or user/device), k_a the secret key, P_a the public key on the ECC curve, P a point on the same curve relying k_a and P_a with the relation $P_a = k_a.P$. Schnorr principle is the verification by the verifier of the possession of k_a by the prover, without sharing the plain value. The protocol is as follows: (i) prover generates a random number n_a and sends to verifier the point $T_a = n_a.P$, (ii) verifier sends a random number n_b, (iii) prover sends a message $s = n_a + k_a.n_b$, (iii) server verifies that the computation of $s.P$ and $T_a +$

$n_b.P_a$ gives the same point (indeed, $sP = (n_a + k_a.n_b).P = n_a.P + k_a.n_b.P$ and $T_a + n_b.P_a = (n_a.P) + (k_a.P)n_b = n_a.P + k_a.n_b.P$). To ensure mutual authentication between user and device, the described protocol is applied two times by exchanging the role of verifier and prover.

The user authentication ensured by the Schnorr protocol prevents against an external attacks wanting to perform scan attacks or exploit JTAG features. The device authentication prevents against counterfeit chips. However, a malicious device is not included in the threat model of this countermeasure, since it is still able to sniff and tamper the test communication when the JTAG interface is unlocked.

In order to ensure a sufficient security level, authors choose to implement ECC curve on 192 bits. The generation of random number is ensured by the implementation of a PRNG based on LFSR with a seed changing at each authentication. The cost of the additional implemented components represents an overhead of 1648% compared to a JTAG controller of 180 GE. To ensure mutual authentication between user and device, the test time cost represents 781 clock cycles, corresponding to the time required to shift the challenge/response, and 1,014,370 clock cycles, corresponding to the time of ECC computations. Concerning the key management, a server stores the secret key and realizes the protocol, facilitating the key management policy for the user, but it requires server maintenance and an online connection for the user on the protected board.

5. *Fine-grained access using challenge/response protocol* [56] details the implementation of a secure controller guaranteeing a secure fine-grained access for different group of users. This solution is combined with a challenge/response protocol giving accreditations to the user, such as the solutions in [53] and [54] for instance. The secure controller presented in [56] is unlocked with the authentication protocol, giving access to the circuit according to the accreditations of the authenticated user. Once a user is authenticated, the authorized JTAG instruction set is unlocked, while the other instructions are locked, i.e. decoded as *BYPASS* instruction. In addition, the secure controller monitors also the address bits shifted in the JTAG BSR in order to prevent an unauthorized read or write operations using the test interface.

   This technique ensures a protected fine-grained access for instruction as well as for memory address, providing the secure access for different group of authorized users.

   For the implementation of the solution, in addition to the secure controller, authors choose to implement the challenge/response protocol established in [53]. The area cost is evaluated to 2300% compared to a JTAG controller of 180 GE. Concerning testability, authors do not propose a way to test the added secure controller. Test time is only impacted in this solution by the challenge/response authentication in [53].

6. *Challenge/Response based on Physically Unclonable Function (PUF)* [57] is an authentication method protecting the access to the IEEE 1500 test wrappers on cores in a SoC. PUFs exploit the physical characteristics of a circuit. For instance, the variations in the propagation delays of logic paths are exploited in the Arbiter PUFs. The principle is to apply a challenge to two identical logic paths. The transition on one logic path occurs first due to the propagation delays

introduced at manufacturing. An arbiter captures the transition occurred on the fastest logic path, resulting to the response of the challenge. The PUF challenge/response pairs are obtained after manufacturing and stored in a server. The PUF output is only accessible during the database creation, and then fuses deactivate the output. The authentication follows a protocol: (i) in the SoC, a PRNG generates a random number, (ii) the server finds in the database two challenges such as the Hamming distance between the responses is equal to the received random number, and (iii) the two challenges are applied to the PUF and the verification of Hamming distance between responses is performed. Once authenticated, the test wrapper of the targeted core is unlocked. An issue with the use of PUF is the variability on responses with the environmental noise. Two queries of the same challenge may give different responses, affecting the authentication mechanism. For this reason, an error-correction module is implemented to eliminate the variability on the PUF responses.

This countermeasure protects against the scan attacks and the exploitation of JTAG features on a specific core. Moreover, the use of PUF ensures also an intrinsic device authentication, preventing against overproduction. Indeed, PUFs are based on physical characteristics of the device, making non-reproducible the secure test access mechanism. The security level of this solution relies on the size of the challenge/response database for avoiding predicting the response or replaying existing challenge/response pairs by an attacker. This way, authors propose to implement 64-bit PUF challenges and to implement a PRNG based on a 32-bits LFSR reseeding after every authentication.

The area cost of this implementation represents an overhead of 60% compared to a JTAG controller on 180 GE. Test time overhead is composed of the time to shift the random number (32 test clock cycles) and time to shift two challenges (2×64 test clock cycles) to the PUF. The authors do not specify how to test the authentication modules. A server facilitates the sharing of the challenge/response pairs to users, but needs to have a large database since challenge/responses pairs differs for each SoC.

7. *Challenge/Response for secure DfD* [58] propose a fine-grained access through the CoreSight architecture. The enable signal on the debug bus allows only filtering of the data processed and saved in the Secure world or in the Non-Secure world. However, OEM and OS debug teams normally have access to debug the Secure world, but protect different assets (e.g. secure boot for OEM team, and DRM cryptographic keys for OS team). Considering a rogue insider in the debug OS team, the debug bus can be exploited to read trace data on OEM firmware containing the secure boot. In the opposite case, an engineer from the OEM team can leak DRM cryptographic keys setting up by the OS vendor. Instead of using only an enable signal filtering the secure data passing through the ARM debug bus, the proposed solution consists in an enhancement of the CoreSight components in order to filter thinner the data with tags. During SoC production, tags are created, stored in a LookUp Table (LUT) and associated to memory regions. Thus, tagged memory regions will contain assets (e.g. cryptographic keys, firmware etc…) belonging to one specific group (OEM, OS or SoC integrator for example). An authentication module based on PUF verifies the permissions of a user accessing the test interface. Once the user is authenticated, the authentication module sends the accreditations

of the user to the asset filtering. When data are accessing through the debug bus, this module compares the accreditations of the user to the tags associated to the data. The asset filtering lets data readable and writable only with the tag corresponding to the user.

This countermeasure targets a precise threat, a rogue insider in a debug team. Others threats are not considered. The main issue with this solution concerns the post-production updates of the asset (e.g. update of a new version of firmware). Since the tags are defined during SoC production, the asset is no longer protected if it is updated into a memory region associated to another tag.

Experimentations have been conducted on an ARM9 processor. This solution represents an area overhead of 6.3%, and a power consumption overhead of 6.2% compared to original processor. The test time cost is due to the authentication phase, representing 300 clock cycles. Challenge/response pairs are shared with the tester using a server, containing a large database with the challenge/response pairs for each SoC.

Overall, the authentication using challenge/response protocol ensures the security against external attacker carrying out scan attacks or exploiting the JTAG features. Moreover, some countermeasures propose a fine-grained access to permit the use of specific instructions, instruments, memory region, or debug bus according to the authenticated user. Device authentication using challenge/response allows also to detect counterfeit devices. Nevertheless, malicious circuits inserted within the test daisy-chain still represents a threat.

III.4.c Encryption of test communication

The encryption of the test communication is another category in the class of secure test access countermeasures. Many solutions have been proposed in order to guarantee the confidentiality of communications within the test infrastructures. Solutions proposed so far rely on a modified interface of the test infrastructure that combines both test data transmission and encryption. Figure 19 presents the core scheme of countermeasures based on test communication encryption. Test vectors are first encrypted off-chip and stored into the test equipment. At test time, encrypted test vectors are sent to the target device, and then decrypted on-chip using the encryption key. Then test operations are performed. Before scanning out a test response, the data is encrypted on-chip by the device under test. The tester collects encrypted responses and decrypts them off-chip using the encryption key for further comparison with expected responses. Test vectors and responses, are thus kept confidential during the testing process. Without knowing the key, there is neither a possibility to control the device to a specific state nor the opportunities to observe the device state. Encrypted test data can thus flow safely through the entire system containing the device under test without risking to be read or written by an unauthorized third party. This mechanism guarantees that the other devices, sharing the same test infrastructure of the target one, are not able to 'understand' any content. Moreover, a user that does not know the secret key used by the stream cipher inside the circuit, is not able to have a successful communication with the target device. Both the controllability and observability capabilities are dimmed by the decryption and encryption process applied respectively on the input and output interface of the device.

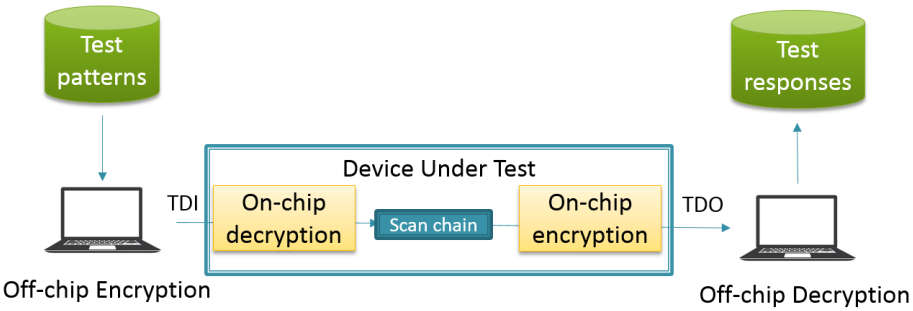


Figure 19: Scheme of test communication encryption.

In literature, the stream cipher has been preferred so far to encrypt the test data, due to its easy adaptability to the serial interface offered by the TDI/TDO signals of the test infrastructures. Among all the stream ciphers, TRIVIUM [59] is the one used in the state-of-the-art due to its low implementation. The encryption scheme provides that the TRIVIUM is initialized with a secret key and an initialization vector IV.

1. *Encryption of JTAG communication* [30] propose to ensure the device authentication and the confidentiality and integrity of the test communication between the tester and the device, using a challenge/response protocol. The first level of protection ensuring the device authentication consists in sending a challenge to the stream cipher. After the initialization of the stream cipher, the response is sent back to the tester, verifying the challenge/response pair in his/her database. The second level of protection ensures the confidentiality of the test communication in addition to the device authentication. In the second level of protection, the response obtained from the stream cipher to authenticate the device is not sent directly to the tester, but used to initialize a second stream cipher. This way, the second stream cipher encrypts the test communication using the response to the challenge as the secret key. The third level of protection ensures the integrity of the test communication in addition to the other levels of protection, thanks to a SHA-1 module computing Keyed-Hash Message Authentication Code (HMAC) on the test messages.

   The first level of protection allows detecting counterfeit chips, thanks to the device authentication. The second level of protection prevents malicious devices to sniff the test communication, and protects *a priori* against an external attacker communicating with the device without knowing the secret key used for encryption. The third level of protection prevents malicious devices to tamper the test data.

   This countermeasure preserves test, debug and diagnosis facilities, but requires additional test time for test operations. The test time overhead corresponds to: (i) shift in the challenge to the first TRIVIUM, (ii) extract the response from the first TRIVIUM after its setup, (iii) send the response to initialize the second TRIVIUM, and (iv) compute the HMAC for every test vectors. Overall, to ensure the three protection levels, test time increase represents an initial overhead of 2,464 clock cycles and the HMAC computation at each test vector. Area cost represents an overhead of 400% compared to JTAG controller of 180 GE. Authors do not specify the test of added stream ciphers and hash function, as well as the share of the challenge/response database with authorized tester.

2. *Encryption of test data in SoC* [60] consists in encrypting test data on the IEEE 1500 test wrapper to protect specific cores, using the TRIVIUM stream cipher. For that, the tester generates a random key for the TRIVIUM cipher and shifts it to the core under test via a dedicated scan chain, non-visible from other cores. A single TRIVIUM stream cipher is used to decrypt the data shifting in the parallel WPI pins and encrypt the data shifting out the parallel WPO pins.

   The proposed solution addresses the threat posed by untrustworthy cores in SoCs. The countermeasure eliminates the risk of a malicious SoC core sniffing test data, as well as a fake device not knowing the secret key used for the encryption. Since the secret key using for encryption is sent by the user, external threats are not considered.

   The protection on one core in a SoC implies an area overhead of 200% compared to a JTAG controller of 180 GE, and a test time cost of 1232 clock cycles composing of the time to shift the secret key and the TRIVUM setup. Area and test time costs have to be multiplied by the number of protected cores in the SoC.

3. *Encryption of the RSN* [61] proposes to encrypt and decrypt the data shifted in and out of the IJTAG network. In addition to the confidentiality ensured on the test communication, the solution proposes to use a security checker in order to detect an attacker trying to guess the RSN structure. In this case, the attacker will perform an incorrect number of shift cycles and this is detected by the security checker. Stub chains (i.e. fake scan chain segments of random length) are also integrated in the RSN, making difficult to reverse engineer the network structure.

   The goal of this solution is to protect items under test against malicious embedded instruments sniffing the communication, and against external attackers who want to illegally use the embedded instruments.

   The implementation costs vary in function of the number of encrypted scan network and the number of stub chains inserted within the RSN. In average, the area cost is about 2800% compared to a JTAG controller of 180 GE. Test time overhead is composed of the TRIVIUM setup, corresponding to 1152 clock cycles, and the additional time required for shifting into the stub chains.

The encryption of test data is a solution aimed to bring confidentiality between tester and protected device, ensuring a protection against malicious devices within the circuit as well as external attacker without the knowledge of the secret key. Nevertheless, the implementation of the stream cipher in these countermeasures presents a vulnerability detailed in Chapter IV, which can be used by attackers for carrying out scan attacks.

III.5. Detection of illegitimate behavior

All the countermeasures seen so far aim to avoid the execution of the attacks using the test infrastructures. Instead, the last presented class of countermeasures aims to detect the execution of

the attacks, monitoring on chip the behavior of the user. When the behavior of the user is considered illegitimate, the system is set in a protection mode, preventing an attacker to continue his/her attack.

Detection techniques can be divided in two categories in function of the detection method. The first category, named *static detection*, gathers the solutions based on fixed rules. As soon as a user accessing the test interface does not respect the rules, the user is considered as an attacker. The second category, named *dynamic detection*, is based on machine learning. These techniques require a training phase for the machine learning in order to distinguish normal behavior to illegitimate one. After this training phase, the machine learning can detect by herself the behavior of an attacker.

III.5.a Static detection

Static detection techniques are based on rules defined at design time. These rules define the legitimate behavior of the use of the test interface. If the actions on the test interface are not considered compliant to a legitimate behavior, the user is classified as an attacker trying to exploit the circuit. For instance, the countermeasure, *encryption of the RSN* in [61], implements a static detection with the security checker controlling the number of shift operations performed by the user to open a SIB in the network. Further static detection techniques have been proposed in the literature.

1. *Sequence filters for accessing instruments* [62] is a detection technique in the IJTAG network. Sequence filters are placed on the JTAG TAP controller in order to control the shift sequence used to open the SIBs in the RSN. If the user tries to access a forbidden instrument, the operation is not allowed by the filter. The filters are deactivated by default to allow manufacturing test. After that, they can be activated either by blowing fuses or by user authentication.

   This countermeasure detects an attacker exploiting the IJTAG network when a SIB open the access to unauthorized instruments. If a user authentication is implemented to access to specific instruments in the RSN, the solution thus provides a fine-grained access to instruments in function of the user accreditations. Others threats are not considered in this solution.

   The implementation of this solution implies no test time cost. Testability is not impacted, expect for the test of the sequence filters (not specified by the authors). Since the implementation of one sequence filters protects one shift sequence, the area cost depends on the number of the defined protected shift sequence. Authors evaluate the overhead from 0.2% for 10 protected shift sequences to 10.6% for 100 protected shift sequences.

2. *Representative based anomaly detector* [63] control the sequence of instructions sent to the JTAG IR register. These sequences are chosen at design time as representative of legitimate operations. If the behavior of the user goes sideways for long time with respect to the representative sequences, an attack is then detected. In the implementation, a counter is associated to each representative sequence. When no counter is fed anymore, this means that a non-representative sequence is being performed by the user, i.e. the circuit is subject to an attack.

The threat model is limited to detect an external attacker exploiting the JTAG features such as debugging, uploading a firmware or exploiting the IJTAG network, assuming that the attacker does not know the JTAG instructions for accessing to these features.

Authors have evaluated the area and power consumption cost on FPGA board. Area overhead represents 2,702 LUTs, 1 LUTRAM and 1736 FFs. The power consumed by the detectors at 100 MHz is 342 mW.

The rules detecting the illegitimate behavior are defined at design time. Static detection techniques cannot thus change the access policies without a complete redesign of the detectors. Dynamic detection techniques bring more flexibility on the design of the detectors.

III.5.b Dynamic detection

Dynamic detection is achieved with the implementation of binary classifiers on-chip for machine learning. These classifiers are able to evaluate the sequence of instructions sent by the user by attaching a label to the behavior, either legitimate or illegitimate. The classifiers have to go under a training phase before being operative. Training phase consists in teaching the classifiers to recognize the user behavior by sending instructions sequences belonging to both labels. This way, the classifier sets its internal classification parameters and is then able to successfully classify the sequences autonomously. In [63], authors propose two different classifiers, the random forest and the support vector machine.

1. *Random forest detector* [63] propose a classifier based on decision trees. A tree takes decision regarding features vectors depending on the JTAG operations performed by the user (e.g. number of clock cycles used for shifting the opcode of an instruction, if the opcode corresponds to a valid JTAG instruction, the number of TMS transitions, etc…). Each tree takes as input a feature vector and outputs a binary value that corresponds to its classification. The classification of every trees are sent to a majority voter that establish the result, i.e. if the behavior on the use of the test interface corresponds to an attacker.

   The threat model in this solution consider only the detection of an attacker exploiting the JTAG features.

   The implementation of the random forest detector is evaluated on FPGA board to an area overhead of 399 LUTs, 624 LUTRAMs and 252 FFs, and to a power consumption of 346 mW at 100 MHz.

2. *Support vector machine detector* [63] defines a decision boundary during the training phase. Regarding four successive instructions, the decision boundary classes these instructions as “normal” sequence or “abnormal” sequence. The four instructions are the optimal length of sequence, determining empirically by the authors.

   Support vector machine detects attacker exploiting attacker exploiting JTAG features. Others threats are not considered.

   The costs are evaluated by the authors to an area overhead of 402 LUTs, 385 LUTRAMs and 236 FFs, and to a power consumption of 335 mW at 100 MHz.

While the detector based on the random forest classifier is able to provide a classification based on static features of the instructions in execution, the support vector machine relies on sequences of more instructions. This makes the classification based on support vector machine more efficient against attacks that are unknown at the moment of the training. A common drawback of these two solutions is that each time a new attack comes out, the training process must be performed again. Moreover, machine-learning techniques show more efficiency if coupled with other protections. Indeed, in some situations the detection can fail because the attack is not recognized. Moreover, once the classifier has detected that the user is performing an attack, the system must activate a locking feature or going into a secure mode.

III.6. Evaluation of the existing countermeasures

Table 1 resumes the evaluation of the existing countermeasures in terms of security considering external threats and internal threats, as well as the impact on testability and applicability of the solution. From this analysis, it emerges that each class of countermeasure targets specific protections. A designer has to select the appropriate countermeasure in function of the considered threat model for his/her application and the characteristics and costs of the solution. In order to protect against a large threat model, several countermeasures of different class have to be combined in order to achieve the required security.

Solutions	Security evaluation						Characteristics and costs evaluation									
	External threats				Internal threats		Testability			Applicability						
	Scan attacks	Exploiting JTAG features	Users with a restricted access	Probing attacks	Malicious component	Counterfeit component	Fault coverage	In-field diagnosis and debug	Test time cost	Implementation	Area cost	Power cost	Impact on DfT flow	Test procedure	Non-modifiable core	Secret sharing
Scan chain avoidance																
Blowing fuses	Secure	Not available	Not available	Insecure (probe on fuses)	Secure	Insecure	Same	Impossible	/	Fuses	0	0	No	Standard before blowing	Applicable	No
BIST [36][39]	Secure	Insecure	Not available	Insecure	Secure (against sniffing)	Insecure	Can decrease	Complicated	Possible overhead	PRNG, Responses compactor	+282%[1]	n.s.	Important (insertion BIST)	BIST procedure	Not applicable	No
Secure scan design																
1. <u>On Scan chains</u>																
Mirror-Key register [13]	Secure	Insecure	Not available	Insecure	Insecure	Insecure	Register faults untestable	Preserved	0	Mirror-Register	+229%[1]	n.s.	Moderate (SE control)	Resetting the circuit	Not applicable	No
Obfuscation [40]	Relative to LFSR structure	Insecure	Not available	Insecure	Insecure	Insecure	Same	Preserved	n.s.	LFSR	n.s.	n.s.	Important (netlist file)	Obfuscating test data	Not applicable	LFSR structure
Scan Enable Tree [41]	Insecure	Insecure	Not available	Secure	Insecure	Insecure	Same	Preserved	+1%[2]	SE trees	+36%[1]	Low	Important (netlist file)	Standard	Not applicable	No
Spy cell [42]	Insecure	Insecure	Not available	Secure	Insecure	Insecure	Same	Preserved	+5%[2]	Spy cells	+27%[1]	Low	Moderate (synthesis command)	Standard	Not applicable	No
Scrambling [43]	Insecure	Insecure	Not available	Secure	Insecure	Insecure	Same	Preserved	0	Scrambling chain	+156%[1]	+7%[2]	Moderate (synthesis command)	Scrambling of data	Not applicable	Scrambling order
Test key in scan chain [44]	Secure	Insecure	Not available	Insecure	Insecure	Insecure	Same	Preserved	+5.6%[2]	Key FFs, LFSR	n.s.	n.s.	Moderate (synthesis command)	Unlocking	Not applicable	Test key

[1] Comparison to a JTAG controller of 180 GE
[2] Comparison to original test time for a DES crypto-processor

Secure comparator [45]	Secure	Insecure	Not available	Insecure	Secure (against sniffing)	Insecure	Same	Complicated	6×(#SFF+1) clock cycles	Comparator	n.s.	n.s.	No	Applying expected responses	Applicable	No
2. On RSN (IJTAG network)																
LSIB [46][47]	Insecure	Relative (against exploiting RSN)	Fine-grained access to **instruments**	Insecure	Insecure	Insecure	Same	Preserved	48 clock cycles (per instrument)	Key bits, Trap bits, LSIB	From +0.07%[3] to +16.2%[4]	n.s.	Low (IJTAG network)	Unlocking	Applicable	Key to open LSIB
Authentication protocol [48]	Insecure	Secure (against exploiting RSN)	Fine-grained access to **instruments**	Insecure	Insecure	Insecure	n.s.	Preserved	532+#SIB clock cycles	SHA-3, PRNG	From +2.8%[3] to +4.9%[4]	n.s.	Low (IJTAG network)	Unlocking	Applicable	Challenge response pairs
Secure test access																
1. Password																
On JTAG standard [49]	Secure (but replay attacks)	Secure (all features)	Not available	Insecure	Insecure	Insecure	n.s.	Preserved	32 clock cycles	Key register	+144%[1]	n.s.	Low (JTAG standard)	Unlocking	Applicable	Password
On IEEE 1500 standard [50]	Secure (but replay attacks)	Secure (on protected core)	Fine-grained access to **cores**	Insecure	Insecure	Insecure	Same	Preserved	256 clock cycles	LFSR	+20%[1]	n.s.	Low (IEEE 1500 standard)	Unlocking	Applicable	Password
2. Challenge/Response protocol																
Keyed-Hashes [51]	Secure	Secure (all features)	Fine-grained access to **instructions**	Insecure	Insecure	Insecure	n.s.	Preserved	256 clock cycles + hash computation	TRNG, SHA-256	>500%[1]	n.s.	Low (JTAG standard)	Unlocking	Applicable	Challenge response pairs in SW debug
Symmetric cryptography [53]	Secure	Secure (all features)	Fine-grained access to **instructions**	Insecure	Insecure	Insecure	n.s.	Preserved	n.s.	AES, SHA-1, TRNG	+2,246%[1]	n.s.	Low (JTAG standard)	Unlocking	Applicable	Challenge response pairs in server
Asymmetric cryptography [54]	Secure	Secure (all features)	Fine-grained access to **instructions**	Insecure	Insecure	Insecure	n.s.	Preserved	n.s.	ECC, TRNG	>2,000%[1]	n.s.	Low (JTAG standard)	Unlocking	Applicable	Challenge response pairs in server

[3] Area overhead for 1 protected instrument in t512505 benchmark
[4] Area overhead for 256 protected instruments in t512505 benchmark

Schnorr protocol [55]	Secure	Secure (all features)	Not available	Insecure	Insecure	Secure	n.s.	Preserved	507,966 clock cycles	ECC, PRNG	+1,648%[1]	n.s.	Low (JTAG standard)	Unlocking	Applicable	Challenge response pairs in server
Fine-grained access [56]	Secure	Secure (all features)	Fine-grained access to **instructions** & **memory**	Insecure	Insecure	Insecure	n.s.	Preserved	n.s.	Secure controller + AES, SHA-1, TRNG	>2,300%[1]	n.s.	Low (JTAG standard)	Unlocking	Applicable	Challenge response pairs in server
PUF [57]	Secure	Secure (on protected core)	Fine-grained access to **cores**	Insecure	Insecure	Secure	n.s.	Preserved	160 clock cycles	PUF, PRNG	+60%[1]	n.s.	Low (JTAG standard)	Unlocking	Applicable	Challenge response pairs in server
Secure DfD [58]	Insecure	Secure (on debug trace)	Fine-grained access to **debug data**	Insecure	Insecure	Secure	n.s.	Preserved	300 clock cycles	PRNG, SHA-1, PUF, LUT	+6.3%[5]	+6.2%[5]	Moderate (JTAG and CoreSight)	Unlocking	Applicable	Challenge response pairs in server
3. Encryption of test communication																
Stream cipher on JTAG communication [30]	Insecure (two times pad)	Insecure (two times pad)	Not available	Insecure	Secure (against sniffing and tampering)	Secure	n.s.	Preserved	2464 clock cycles (+ HMAC on each vector)	TRIVIUM, fuses, HMAC	+400%[1]	n.s.	Low (JTAG standard)	Encrypting test data	Applicable	Challenge response pairs
Stream cipher on cores' data [60]	Insecure (two times pad)	Insecure (two times pad)	Fine-grained access to **cores**	Insecure	Secure (against sniffing)	Secure	n.s.	Preserved	1232 clock cycles per protected core	TRIVIUM	+200%[1] per core	n.s.	Low (IEEE 1500 standard)	Encrypting test data	Applicable	Secret key and IV
Stream cipher on IJTAG network [61]	Insecure (two times pad)	Insecure (two times pad)	Fine-grained access to **instruments**	Insecure	Secure (against sniffing in RSN)	Secure	n.s.	Preserved	1152 clock cycles + STUB chains	TRIVIUM, Security checker, STUB chains	>2,800%[1]	n.s.	Low (IJTAG network)	Encrypting test data	Applicable	Secret key and IV
Detection of illegitimate behavior																
1. Static detection																
Sequence filters [62]	Insecure	Detects (against exploiting RSN)	Fine-grained access to **instruments**	Insecure	Insecure	Insecure	n.s.	Preserved	0	Sequence filters	From 0.2% to 10.6%[6]	n.s.	Low (IJTAG network)	Standard	Applicable	No

[5] Compared to ARM9 processor
[6] From 10 to 100 protected shifting sequences

Representative-based anomaly detector [63]	Insecure	Detects (all features)	Not available	Insecure	Insecure	Insecure	n.s.	Preserved	0	Representative sequences	LUT: 2,702 LUTRAM: 1 FF: 1,736	342 mW (@ 100 MHz)	Low (JTAG standard)	Standard	Applicable	No
2. Dynamic detection																
Random forest detector [63]	Insecure	Detects (all features)	Not available	Insecure	Insecure	Insecure	n.s.	Preserved	0	Decision trees	LUT: 399 LUTRAM: 624 FF: 252	346 mW (@ 100 MHz)	Low (JTAG standard)	Standard	Applicable	No
Support vector machine detector [63]	Insecure	Detects (all features)	Not available	Insecure	Insecure	Insecure	n.s.	Preserved	0	Support vector machine	LUT: 402 LUTRAM: 385 FF: 236	335 mW (@ 100 MHz)	Low (JTAG standard)	Standard	Applicable	No

Table 1: Evaluation of the existing countermeasures (red: drawback, green: benefit).

III.7. Conclusion on the state-of-the-art countermeasures

Test infrastructures provide test, diagnosis and debug, which are mandatory in complex ICs to ensure product quality. Nevertheless, these infrastructures are an open door to carry out attacks, thus requiring implementing countermeasures. In this chapter, we have provided a taxonomy of the many countermeasures proposed in the literature. Solutions have classified in four groups: scan chain avoidance, secure scan designs, secure test access and detection of illegitimate behavior. Each class of solutions have pros and cons in terms of security and costs.

The *scan chain avoidance* countermeasure has the main disadvantage to reduce or even to eliminate the testability of the device under test. *Secure scan designs* solutions protects against external attacks, but they lack of protection against internal threats. Additionally, these solutions modify directly the scan structure, thus being difficult to integrate in the DfT flow, and to apply on non-modifiable cores. S*ecure test access* techniques based on complex authentication mechanisms provide also a strong protection against unauthorized accesses to the test infrastructures, without including in their threat model the insertion of malicious component. A main disadvantage of this kind of countermeasures is the cost to implement the complex authentication protocol, based on either password or challenge/response. Additionally, the password or the challenge/response pairs have to be shared with authorized parties, implying a management issue. The last category of countermeasures, *detection of illegitimate behavior*, prevents the misuse of the test interface using algorithms to detect an external attacker. This class of countermeasures cannot protect against internal threats.

Regarding the state-of-the-art countermeasures, no solution provide a complete protection against both external and internal attackers. Nevertheless, the encryption of the test data represent a promising solution for providing the most complete protection, ensuring confidentiality on the test communication. In principle, an unauthorized user and a malicious core are not able to set desired internal states, nor to observe plain test data, since both do not know the secret key encrypting the test data. However, the proposals in the literature does not implement a secure stream cipher to encrypt the test data. An attacker can thus circumvent the encryption of the test communication by exploiting a vulnerability in the stream cipher implementation. In the next chapter, we detail how the existing countermeasures based on stream ciphers are unsecure against scan attacks. Moreover, we present new cost-efficient proposals for the encryption of the test communication: one based on block cipher, another based on stream cipher fulfilling the security requirements.

Chapter IV

Proposed countermeasures based on scan encryption

Summary

IV.1. Encryption with symmetric ciphers

As seen in Chapter III, a countermeasure against attacks exploiting the test interfaces is to encrypt on-chip the data flowing through the scan chains. As depicted in Figure 19, two symmetric ciphers are implemented within the device under test: one for encrypting test data shifting at the scan input, another for decrypting test data shifting at the scan output. Symmetric cryptography is more adapted for encrypting the test communication compared to asymmetric cryptography, because test data are encrypted/decrypted in the device using the same key. Additionally, symmetric ciphers have a cheaper implementation compared to asymmetric ciphers.

Among the symmetric ciphers, two types can be used: either stream ciphers or block ciphers. Both have pros and cons in terms of performance and security. Because of their smaller footprint, stream ciphers are generally preferred to block ciphers in the literature, as seen in Chapter III. Nevertheless, when incompletely implemented, solutions based on stream ciphers are prone to differential attacks and thus have to be completed. This mitigates their interest versus block cipher solutions. In this chapter, we propose two solutions: one exploiting block ciphers, and another exploiting stream ciphers fulfilling security requirement. We refer to Countermeasures based on Block cipher as CBC and Countermeasures based on Stream cipher as CSC.

For the sake of completeness, we provide firstly a brief reminder about the cryptographic primitives, called ciphers, which underlie the encryption techniques. We provide a presentation of block and stream ciphers in order to set the terminology and highlight the key features that are needed to appreciate the vulnerabilities explained thereafter.

In general, a cipher allows the sender to transform an input message m into a ciphered version c using a secret key k. The receiver needs to be able to rebuild m from c upon knowledge of the same k (or derived from k).

A cipher is composed of two functions: E, called encryption function, and D, called decryption function, such as:

- The encryption algorithm takes as input the message m and the secret key k, and outputs a ciphertext c, so that $E(k, m) = c$.
- The decryption algorithm takes as input the ciphertext c and the secret key k, and outputs the plaintext m, so that $D(k, c) = m$.

The encryption of a message followed by the decryption of the correspondent ciphertext must result in the initial message, i.e. $D(k, E(k, m)) = m$.

Block and stream ciphers provide confidentiality in the test infrastructures. The main difference between stream and block ciphers relies on the size of the data that are processed in each encryption. Stream ciphers encrypt one bit at a time from a bitstream; this results in the encrypted message having a bit-to-bit correspondence with the plaintext message. Differently, block ciphers take as input an n-bit block of the plaintext, which is encrypted into an n-bit block ciphertext; in this case, the properties of the plaintext are dispersed on the whole n bits of the ciphertext.

IV.1.a Block ciphers

Block ciphers are based on mathematical objects called Pseudo Random Permutations (PRP). They are invertible functions that take as input an n-bit value m and a secret key k, and output an n-bit value c. A Pseudo Random Permutations is considered secure if, fixed with a key k, the resulting function is indistinguishable from a random bijective function on n-bit values.

Block ciphers implement a secure Pseudo Random Permutations. They are made of an encryption function that is able to encrypt a plaintext block into a ciphertext block using a secret key; and a decryption function that performs the inverse operation and retrieve the plaintext block from the ciphertext.

The most used block cipher is AES [12]. Other algorithms have been proposed to be more lightweight, i.e. with a lower cost in terms of area and power consumption, such as PRESENT [64] or SKINNY [65].

IV.1.a.i Block ciphers limitations

Block ciphers are based on strong primitives and allow to perform many encryptions with the same secret key. For this reason, an attacker trying to attack a block cipher has the ability to exploit the cipher comparing many plaintext/ciphertext pairs encrypted with the same key.

One weakness arises from the kind of function that is implemented and the bit length of the key. Since the function implemented by the block cipher is publicly known, the target of the attacker is to retrieve the secret key knowing a certain number of plaintext/ciphertext pairs. If the secret key is relatively short, this can be retrieved via exhaustive searching, i.e. trying out all possible keys until a given plaintext gives the corresponding ciphertext as output. If the secret key is n bits long, the complexity of the exhaustive search attack is 2^n.

In the case in which the block cipher function is not properly designed, the exhaustive search attacks can have a complexity that is less than 2^n . A classic example is verified when the function that is implemented by the block cipher is near to be linear. In this case, the whole encryption function can be modeled as a transformation matrix; this provides the execution of *linear attacks*, which allows the retrieval of the key in shorter time with respect to exhaustive search. For this reason, the standard block ciphers are implemented with functions that are the furthest possible from linearity.

Another limitation is inherent to the fact that the same key is used many times. When two equal message blocks m_1 and m_2 are encrypted, the correspondent ciphertext blocks c_1 and c_2 are equal as well. This can give precious information to the attacker, namely to *replay attacks* for instance. In order to overcome this problem, the encryption function is usually randomized, i.e. given a message block m and a key k, the produced ciphertext block c is not deterministically resulted, but it can belong to a bunch of possible ciphertexts. This randomization is injected by making the encryption function dependent from a value, called *nonce*, which is different in every encryption. The nonce-based implementation of block ciphers is defined as $E(n,k,m) = m \oplus F(k,n)$, where n is the nonce, k the secret key, m the message, and F the encryption function of block cipher. This implementation of the block cipher operates as a stream cipher, where the keystream is the encryption result of the nonce with a secret key.

Several block cipher implementations exist to operate as stream cipher. One of them is the counter mode (CTR). In this implementation, the nonce n is composed as $(IV \parallel cntr)$, where IV is a random value and $cntr$ is a counter that counts the number of blocks that are encrypted with the same secret key and IV.

IV.1.a.ii Application for encrypting test data

Test infrastructures present serial interface: the serial ports SI/SO of scan chains, TDI/TDO of JTAG, WSI/WSO of IEEE 1500 wrapper, or the TDI/TDO of IJTAG network. For this reason, no solution in literature has proposed to use block ciphers, preferring instead the use of stream ciphers, operating bitwise. Nevertheless, as described thereafter, stream ciphers are based on cryptographic primitives that are weaker than block ciphers, which implies a potential weakness in the countermeasures.

IV.1.b Stream ciphers

Stream ciphers are based on a theoretical cipher, called One Time Pad. In the One Time Pad, the secret key k must be as long as the message m. The encryption function is defined as $E(k, m) = m \oplus k$, and the decryption function as $D(k, c) = c \oplus k$. If k is perfectly random (i.e. according to the uniform distribution), the One Time Pad has perfect secrecy. This means that the produced ciphertext is indistinguishable from a random sequence (this is due to the properties of the XOR operator). In this case, it is impossible for an attacker that intercepts the ciphertext to derive any information neither on the message nor on the key. However, from a practical point of view, the One Time Pad is not implementable because of the key length.

Stream ciphers are an implementation of the One Time Pad. Instead of XORing a random key k as long as the plaintext, a Pseudo-Random Number Generator (PRNG) generates a pseudo-random sequence of bits called *keystream*. The PRNG takes as input a value k, called the seed of the stream cipher, and outputs the keystream $S(k)$. The encryption and decryption functions are thus defined as $E(k, m) = m \oplus S(k)$ and $D(k, c) = c \oplus S(k)$.

As far as the PRNG produces a keystream that is unpredictable, the resulting stream cipher is considered to be secure.

As seen in the state-of-the-art in Chapter III, the stream cipher TRIVIUM [59] is widely used in the context of scan chain protection. It is based on a Non-Linear Feedback Shift Register used as PRNG. The seed of the TRIVIUM PRNG is made by an 80-bit secret key K, and an 80-bit initialization vector IV, which is publicly known. The generated keystream is denoted as $S(K, IV)$.

IV.1.b.i Stream ciphers limitations

Stream ciphers security relies on the implementation of the PRNG. The stream ciphers are secure as far as the PRNG produces a keystream that is unpredictable. However, they have intrinsic weaknesses that facilitate attacks when they are used incorrectly.

An important requirement for the security of the stream cipher is that the seed k must be used only once. In the opposite case, a simple attack can be performed, which is called *two times pad*. When

the same seed k is used to encrypt two different messages m_1 and m_2, the two bitstreams are equal. Thus: $c_1 = E(k, m_1) = S(k) \oplus m_1$ and $c_2 = E(k, m_2) = S(k) \oplus m_2$. This leads to:

$$c\_1 \oplus c\_2 = (S(k) \oplus m\_1) \oplus (S(k) \oplus m\_2) = m\_1 \oplus m\_2$$

The attacker can exploit the XOR of two messages for a differential attack that consists in obtaining confidential information from the difference between messages.

The two times pad is exploited when the PRNG used to generate the keystream has a short period, i.e. when it produces the same keystream after a fixed amount of time during the same test session. This vulnerability is also exploited when the same PRNG seed is re-used to perform different encryptions. For example, if the TRIVIUM stream cipher is set up with the same K and IV each time that an encryption has started, the same keystream $S(K, IV)$ is used to encrypt different messages.

IV.1.b.ii Vulnerabilities of existing countermeasures using stream ciphers

In Chapter III, we have described the protections based on stream ciphers presented in the literature so far. These countermeasures show some vulnerability, especially towards differential scan attacks. We consider a protection on an AES IP implemented in a SoC. The protection is based on the encryption of the test communication with a stream cipher which secret key is unknown from the attacker. The responses of the AES core are thus encrypted with a stream cipher. The stream cipher produces the keystream $S(K, IV)$ from the key K and the initialization vector IV. Then the data are XORed with the keystream. Let us illustrate how differential scan attacks [13]–[16] on a crypto-core implementing the AES algorithm can be performed even with data encryption. Let R_1 and R_2 be the two responses, shifted out from the round register after one round of computation. For the response R_1, the stream cipher generates a keystream $S(K_1, IV_1)$ from the key K_1 and the initialization vector IV_1. Then, after a reset, the second response R_2 is encrypted with the keystream $S(K_2, IV_2)$ from the key K_2 and the initialization vector IV_2. The differential scan attack, described in Chapter II, consists in computing the Hamming distance between the two encrypted responses. Therefore, the stream cipher encryption prevents the attack unless the key and the IV have always the same value after the reset, i.e. $K_1 = K_2$ and $IV_1 = IV_2$. In this case, the same keystream is generated to encrypt two different responses. We have thus obtained a two times pad, and by XORing the two encrypted responses, $R_1 \oplus R_2$ is obtained. Therefore, an attacker can carry out the differential scan attack, even if the test responses are encrypted.

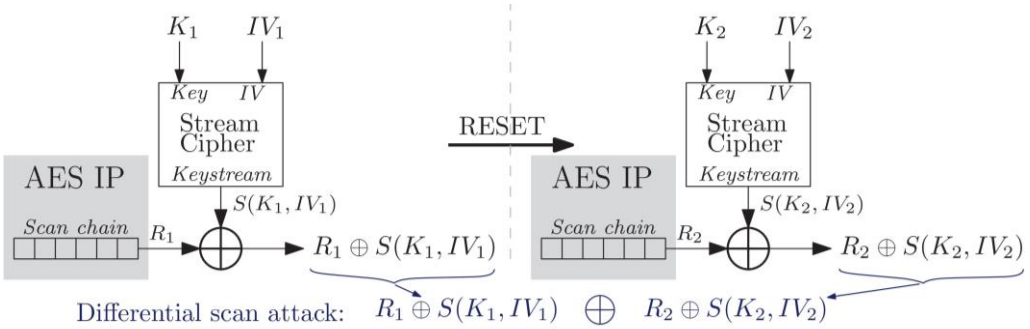


Figure 20: Differential scan attack on test data encrypted with stream cipher.

The CSC presented in [30], [60] and [61] are all exposed to this vulnerability if the IV and the key are not properly refreshed between one encryption and the next one. In [30], the IV is hardwired with fuses, by consequently being the same for each generated keystream. As for the key, the user has to send a challenge to configure the secret key. Even if the attacker does not know the configured secret key, he/she can send the same challenge twice. This way the stream cipher encrypts using the same secret key. Being both the secret key and the IV not changed, the produced keystream is the same. Concerning the solution proposed in [60], the key is sent by the user, and the IV configuration is not specified. In [61], the authors evoke the use of a unique set of keys and IVs for each protected instrument in the IJTAG network. However, the re-use of the same set of keys and IVs to encrypt the test data shifted through a protected instrument leads to the presented vulnerability.

As a conclusion, the same seed must not be used more than once, in order to avoid the generation of the same keystream for several encryptions. In order to protect against differential scan attacks [13]–[16], the stream cipher encryption needs to have different IVs and/or keys for each generated keystream.

IV.2. Principle of proposed scan chain encryption

Regarding the block and stream ciphers overview, we propose two solutions applied for securing the test infrastructures, taking into account the ciphers' features and vulnerabilities. We present a CBC solution, more secure compared to the state-of-the-art consisting in encrypting the test communication, since this countermeasure is based on block cipher. Nevertheless, the encryption by block requires padding the test data, as described in Section IV.3. We also present a secure CSC without the *two times pad* vulnerability in Section IV.4.

For both proposed solution, we assume that the countermeasure is applied on an IC embedding at least one crypto-core, one or multiple secret keys stored in a secure non-volatile memory, and a Secret Key Management Unit (SKMU). We also consider that the circuit implements scan-based DfT where at least some FFs of the scan chains belong to the crypto-core, thus being the target of a possible scan attack. Moreover, we consider that the circuit is used for applications where there is a need for a debugging facility implemented through the access to the scan chain.

The main idea is to decrypt/encrypt the content of the scan chain with either a lightweight block cipher algorithm, or a secured stream cipher algorithm.

The principle, valid for both encryption techniques, is illustrated in Figure 21. Two scan ciphers are implemented at the scan pins. The *Input Scan Cipher* is in charge of processing the on-chip decryption of the test patterns, and the *Output Scan Cipher* is in charge of processing the on-chip encryption of the test responses.

In order not to manage an additional secret key within the circuit to deal with the encryption of the scan chain content, we propose to use the SKMU already existing in the circuit. This way, authorized users encrypt/decrypt test data using a secret key shared with authorized users thanks to the SKMU already implemented. If the SKMU is able to provide a dedicated secret key during the manufacturing test, the scan chain is encrypted with the same secret key, and the external tester is able to encrypt

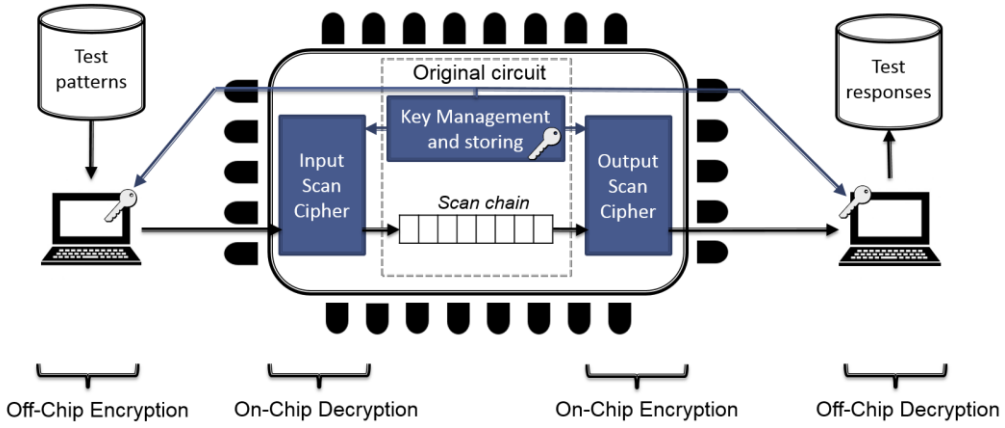


Figure 21: Principle of proposed scan chain encryption.

the scan chain because the secret key is the same used by the ATPG to generate test vectors. When the circuit is in mission mode, the SKMU delivers a different secret key (i.e., the functional one), used by the ciphers implemented within the circuit. Since we assume a crypto-processor in the CUT, the key used to encrypt test data is stored in the same secure memory as the crypto-processor key. The key is also managed with the same operations (key generation, activation and revocation) of the SKMU already implemented in the circuit. Thus the proposed solution introduces no issue with key management, contrary to secure test access using password [49], [50] or challenge/response protocol [51]–[58].

If a user wants to debug the system, the access to the scan chain is possible assuming the debugger knows the functional secret key. The encryption can be performed either at chip scan-in and scan-out, or at each scan-in and scan-out of the protected devices. In the first solution, all data scanned in and out the chip are decrypted/encrypted. In the other solution, the user exploits the reconfigurable scan chains by putting all the devices in bypass mode, except the one that he wants to debug. In any case, if the user knows the secret key applied by the circuit in any of the modes implemented by the SKMU, the same user will be able to encrypt/decrypt the data to/from the scan chains, consequently establishing an automatic authentication method.

IV.3. Proposed countermeasure based on block ciphers

We present a secured test architecture based on lightweight block ciphers for encryption of both scanned-in and scanned-out test data. Lightweight encryption is used for limiting area overhead. As presented in this section, test protocols, diagnostic and debug facilities, as well as design flows are not affected. We detail the application of the CBC solution on single scan chain for both fault models, stuck-at faults and transition-delay faults, as well as the application of the solution on multiple scan chains.

IV.3.a CBC applied on single scan chain

As seen in Chapter I, the main fault model to mimic a manufacturing defect within a circuit is the stuck-at faults model. However, not all faults can be analyzed using this model. A tester would use a transition-delay faults model to detect timing defect in a circuit. The scan chains are used to test the

stuck-at faults in a circuit as well as the transition-delay faults. We show that testability is not affected for both fault models with the scan chain encryption on single scan chain. Additionally, we describe the mode of operations of the CBC, as well as an optimization consisting in introducing low cost observation points.

IV.3.a.i Stuck-at faults testing with CBC

As shown in Figure 22, two block ciphers are implemented: Input Scan Cipher for the decryption performed at scan input, Output Scan Cipher for the encryption performed at scan output. The procedure of stuck-at faults testing, consisting in applying test vectors and collecting test responses in order to compare with expected responses, needs to be adapted to the CBC solution. The proposed test procedure involves the following steps:

1. Generate test patterns for the circuit under test and compute expected 'fault-free' test responses as usual;
2. Off-chip encrypt the test patterns with the chosen lightweight cryptographic algorithm and the secret key related to the current activity;
3. Scan-in an encrypted test pattern, which is first on-the-fly decrypted using the additional Input Scan Cipher, then scanned in the circuit under test;
4. On-the-fly encrypt the test responses using the additional Output Scan Cipher before shifting-out the encrypted circuit response;
5. Off-chip decrypt the encrypted test responses to obtain the actual responses of the circuit and compare with expected ones.

Every N clock cycles, the shift operation must be interrupted in order to allow the scan ciphers to encrypt/decrypt the N-bit data. By assuming that the encryption operation lasts D clock cycles, this solution would require D additional clock cycles for every N bits, and would result in an excessive test time overhead. To reduce this overhead, we propose the use of two registers in each scan cipher, as depicted in Figure 22. These extra registers allow interleaving the shift operation and the encryption process. These two ciphers each have two N-bit round registers (R1 and R2) with the two operating modes, parallel load and serial shift, detailed in Figure 23. Parallel loads are used for storing encryption/decryption results, shift operations are used to load test data into the circuit or observe its internal state. While one of the two registers (e.g., R1) is serially loaded with new data, the other one

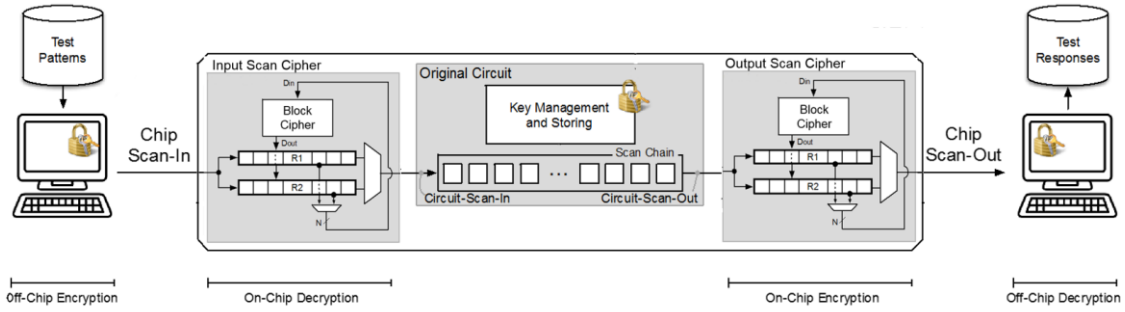


Figure 22: Basic scheme of CBC on a single scan chain configuration.

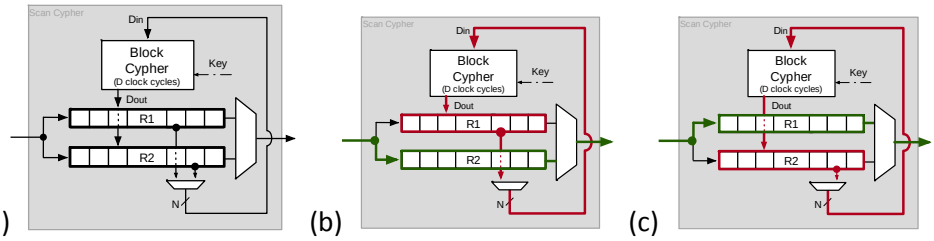


Figure 23: (a) generic scheme of scan cipher; (b) R1 is used for encryption while R2 shifts test data; (c) R2 is used for encryption while R1 shifts test data.

(i.e., R2) is used in the meantime to store decrypted (respectively encrypted) test data shifted-in during the N previous clock cycles.

Thus test time is impacted by 2xN extra clock cycles used at the beginning of the test procedure for loading R1 and R2 registers with first test vector to decrypt. The same additional offset of 2xN extra clock cycles is required at the end of the test procedure in order to read-out the last test response stored in R1 and R2 of the Output Scan Cipher.

IV.3.a.ii Transition-delay faults testing with CBC

As described in Chapter I, LOS and LOC techniques are used for testing the transition-delay faults. In both solutions, a vector V1 is shifted to initialize the scan chain at a frequency usually much slower than the nominal frequency, while the transition to V2 is performed at the nominal frequency.

The LOS test applied with the scan chain encryption consists in encrypting vector V2 by the tester before shifting in the Input Scan Cipher. The encrypted data is then decrypted in one of the Input Scan Cipher registers. Once V2 decrypted, F-1 shifting operations are performed at shifting frequency (to fill the scan chain with the vector V1) and the last shifting operation is performed at nominal frequency (to launch vector V2). In order to adapt the proposed encryption architecture to the LOS scheme, it is necessary to guarantee that the scan cipher is able to run at the circuit nominal frequency, which is higher than the frequency of shift operations. However, if the number of clock cycles required to perform decryption is smaller than the number of FFs in the registers R1/R2, the combinational logic that must be able to run at the nominal (higher) frequency is composed of the scan ciphers registers (R1 and R2) and the output multiplexer. All the encryption algorithms we have used (PRESENT and SKINNY algorithms) work this way. From the design point of view, it is enough to guarantee that the standard gates implementing the multiplexers can correctly work at the nominal frequency.

Regarding the LOC test, since the transition at nominal speed is applied when the scan-enable signal is not asserted, the scan ciphers are not involved at all. Therefore, the LOC scheme can be applied without any interference with the scan ciphers.

In Chapter I, we also present the LOES test, an alternative method to test transition-delay faults. Unlike the LOS or LOC test where F shift operations are needed to test the transition faults, (F+1) shift operations are needed to achieve the LOES test. Due to the extra shift, the scan operations cannot follow the same operations as presented for stuck-at faults testing. This method should thus be avoided when the scan chain encryption is applied. However, in the common scenario, ATPG performs the combination of LOS and LOC tests to achieve a high fault coverage, implying not impact on transition-delay faults test procedure.

IV.3.a.iii Mode of operations

Scan ciphers implemented in the CBC solution follow a specific test procedure to test both stuck-at faults and transition-delay faults. A controller is in charge of enabling the correct sequence of operations based on the value of the scan-enable signal. The controller, described in Figure 24, is composed of an 8-state FSM and a counter modulo N. The counter (ctr) counts the number of clock cycles in test mode (SE=1), it stops to count when the circuit is in functional mode (SE=0). At reset (RST=1), the FSM is in HOLD_WR1 state where the block ciphers are disabled. When the scan-enable is asserted (SE=1), the FSM goes to the ENC_R2 state where the scan ciphers are enabled. The R2 register is decrypted/encrypted, and in parallel, test data are shifted into the R1 register. When the counter reaches $T_{enc} - 1$, corresponding to the number of clock cycles required to perform a decryption/encryption, the decryption/encryption of R2 content is finished. The FSM goes to WAIT_R2 state: the scan ciphers stop to operate, and test data are still shifted-in the R1 register. R1 is entirely filled when the counter reaches N-1. The FSM goes then to the ENC_R1 state and the counter restarts from zero. In the same manner, the ENC_R1 and WAIT_R1 states correspond to the decryption/encryption of R1 and the filling of R2 with scanning-in test data. Whenever the scan-enable is disabled (SE=0), the FSM goes to the corresponding hold state (HOLD_ER2, HOLD_WR2, HOLD_ER1 and HOLD_WR1). In these hold states, the block ciphers are disabled. The block ciphers are enabled as soon as the scan-enable is asserted (SE=1), and the scan operations continue where they left off.

From a security point of view, if the scan-enable is disabled during an encryption (i.e., in the middle of a shift operation), all the registers of the controller and scan ciphers are frozen. It is not possible to access the scan chain content and scan ciphers registers content. Concerning FSM vulnerabilities, no signal connected to the FSM is directly accessible. Therefore, it is not possible to bypass the encryption by altering FSM control signals. A possible attack imagined on the proposed test infrastructure could

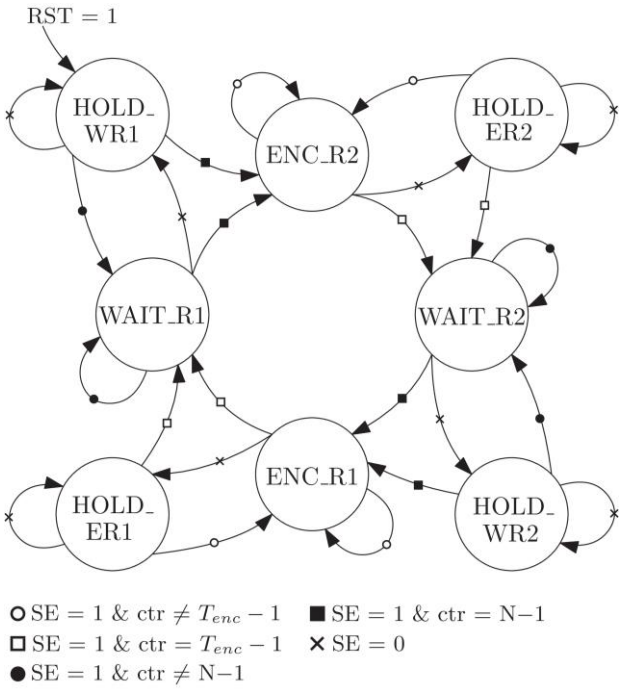


Figure 24: Finite State Machine controlling both scan ciphers in CBC.

be to reset the FSM before the end of R1 filling. This reset operation would lead the FSM to start over shift operations on R1 and unencrypted data would be shifted out. For this reason, registers R1 and R2 are reset whenever the FSM is reset.

The scan chain is filled/flushed segment by segment, each segment consisting of N bits (N being the width of the round register). Managing these operations with scan chains which length is not a multiple of N is feasible. Following the same regular N-shifts scheme the controller can correctly feed the scan chain at the cost of additional clock cycles used to complete the shift operation on the smaller-than-N segment. Figure 25 shows the complete time diagram of shift operations in the case of a circuit having F=S•N+R flip flops, where:

- F is the total number of FFs in the original circuit;
- S is the number of N-bit segments;
- R = F mod N;
- I_j^k is the j[th] N bits segment (with 0≤j≤S) of the k[th] test pattern provided to the circuit;
- O_j^k is the j[th] N bits segment (with 0≤j≤S) of the k[th] test response obtained from the circuit;
- E(x) is the encrypted value of a segment x;
- *decrypt()* and *shift()* are the two operations executed inside a block cipher.

Figure 25 presents the time diagram on a circuit composed of F=3N+R scan FFs when the CBC solution is applied. Each test pattern is divided into three segments $I_{2\ (N\ bits)}^{i}\ I_{1\ (N\ bits)}^{i}\ I_{0\ (R\ bits)}^{i}$, the last segment is padded with N-R extra-bits $P_{(N-R\ bits)}^{i}$. Each test response is also composed of three segments $O_{2\ (R\ bits)}^{i}\ O_{1\ (N\ bits)}^{i}\ O_{0\ (R\ bits)}^{i}$. At the step where the first segment of the second pattern I_0^2 is decrypted, the scan chain is actually set to the first test pattern value: $I_{2\ (N\ bits)}^{1}\ I_{1\ (N\ bits)}^{1}\ I_{0\ (R\ bits)}^{1}$,

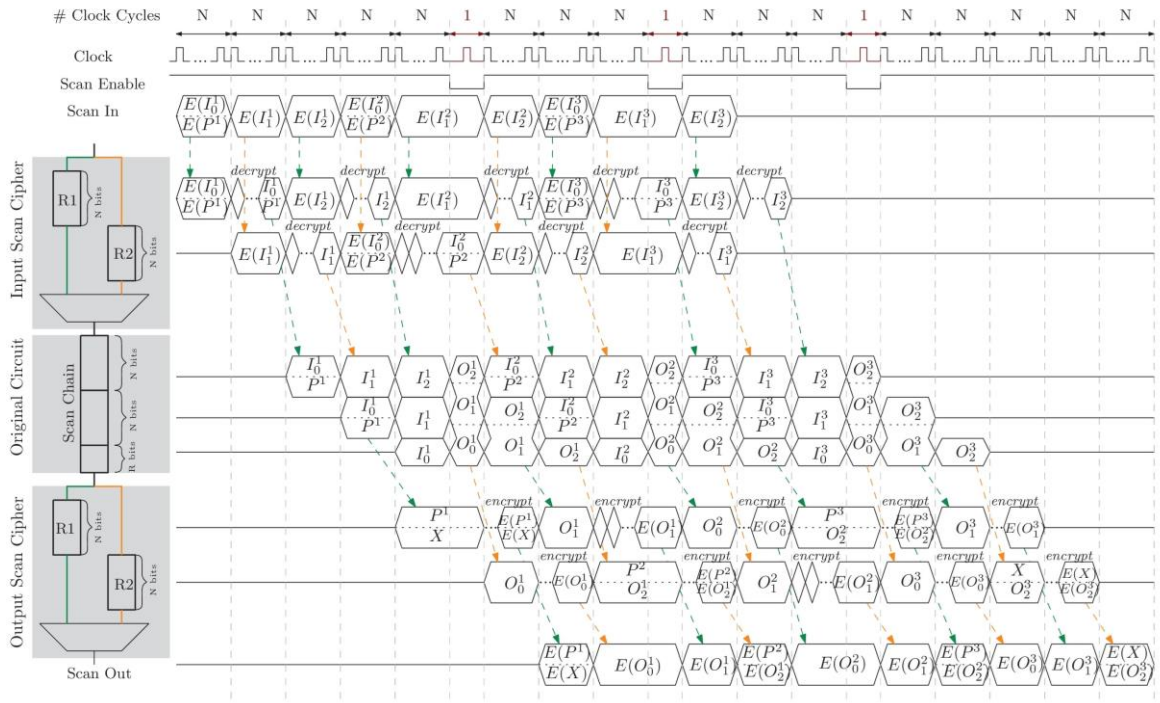


Figure 25: Time diagram of shift operations in the case of CBC applied on single scan chain.

while N-R first bits of the first scanned-in segment $P^1_{(N-R\ bits)}$ have been shifted in the Output Scan Cipher register. After 3N+1 clock cycles, the N-R bits $P^2_{(N-R\ bits)}$ of the second pattern-first segment are encrypted and shifted out as part of the last R-bits test response $O^1_{2\ (R\ bits)}$.

Concerning the test time overhead in the case of R>0, in addition to the 2·2N clock cycles, N-R additional shift operations are needed for each test pattern. More formally, by defining T the number of clock cycles for the original circuit to be tested without the scan attack countermeasure (defined in equation (1)), and K the overall number of test patterns, the number of clock cycles T_f required to test the circuit with the encryption of the scan chain is given in equation (2):

$$T_f = \begin{cases} T + [2 \cdot 2N] & if\ R = 0 \\ T + [2 \cdot 2N + (N-R)(K+1)] & if\ R > 0 \end{cases} \quad (2)$$

Considering a block cipher encrypting/decrypting block size of N=64 bits (such as PRESENT and SKINNY-64), we evaluate the test time cost on 5 circuit examples: a triple-DES core, a pipelined AES core with the 128-bits and 256-bits version, a RSA 1024 bits core and a LEON3 processor. Table 2 resumes the results reporting the result for the original circuit after scan insertion (*Scanned Circuit*) and the overhead (%) induced by CBC. For each circuit, line *#SFF* reports the number F of scan FF, line *#Patterns* reports the number K of patterns needed to test the circuit, line *Test Cov* reports the test coverage of the original circuit. Line *Test Time* reports for each circuit the test time of the original implementation in terms of clock cycles, and the overhead induced by the encryption of both test patterns and test responses. In the case of pipelined AES 256 core, scan length 12736 FFs is a multiple of N=64. Therefore, the test time overhead is only of 2x2N=256 cycles for scan-in initialization and last scan-out, which represents only 0.01% of the original test time. At the opposite, the pipelined AES 128 core has 7873=123×64+1 FFs. Therefore, the number of additional shift operations on each patterns is N-R=63 additional clock cycles. It is the worst case in terms of cost on each pattern. However, even in this case, the test time overhead is limited to 0.81%. For the others circuits, the number of additional clock cycles in each pattern is 24 for Triple-DES, 53 for RSA and 2 for LEON3 processor.

The number of patterns found by ATPG achieves only 70% of stuck-at fault coverage on that CUT because we stopped test pattern generation due to limitation in terms of memory allocation (line *Test Cov* in Table 2).

Circuit	Triple-DES		Pipelined AES128		Pipelined AES256		RSA 1024		LEON3	
	Scanned Circuit	CBC Overhead (%)	Scanned Circuit	CBC (%)	Scanned Circuit	CBC (%)	Scanned Circuit	CBC (%)	Scanned Circuit	CBC (%)
#SFF	8808=137×64+40		7873=123×64+1		12736=199×64		16459=257×64+11		107518=1679×64+62	
#Patterns (K)	77		246		357		2 393		107	
Test Cov.	100%		100%		100%		100%		70%	
Test time (clock cycles)	687101	+0.31	1944877	+0.81	4559845	+0.01	39405239	+0.33	11612051	+0.004

Table 2: Test time cost of CBC for several circuits.

IV.3.a.iv Optimization of CBC solution

The additional clock cycles that are wasted in order to synchronize the CBC scheme using constant segment-length N can be actually exploited for testability improvement without requiring any additional test time.

Indeed, for every test pattern, N-R extra clock-cycles are used to shift N-bits data on a regular manner when the original scan chain involves a number of scan FFs that is not a multiple of N. By adding N-R dummy flip-flops to the original scan chain, we can use these extra FFs as test points, such as presented in Chapter I. Test point insertion does not impact test time, and in particular, observation points can reduce the number K of test patterns for the same fault coverage. Consequently, by adding some FFs to the scan chain and by using them as observation point (for the scan length being a multiple of N), it is possible to compensate for the additional shift operations required by CBC solution.

In order to show that test time cost is reduced with the optimization of the solution, we have conducted experimentations with PRESENT block cipher with block size on 64 bits (N=64) and the encryption/decryption performed in 32 cycles. Experiments are conducted thanks to the synthesis tool Design compiler [66] using a 65-nm library and the ATPG tool TetraMAX [2].

The implementation involves extra costs in terms of area overhead. This cost is relative to the number of scan FFs in the original CUT. For instance, LEON3 processor needs two extra FFs to pad its scan chain. The cost is only 2 FFs over 107518 FFs (<0.002%). In the worst case, 63 FFs are added to AES-128 core, i.e. 0.8% compared to the original scan chain with encryption. For Triple-DES (resp. RSA) circuit, dummy FFs represent an increase of 0.27% (resp. 0.32%) on the total scan chain.

The DfT tool Tetramax was used for selection of observation points in the circuit logic. We constrained the tool to use only the N-R extra FFs for testability improvement. After selection, observation points drives extra XOR trees ending on the proposed extra FFs, thus allowing their observation at test time. The XOR-trees configuration depends on the number of signals to observe. The number of test points per tree is user-defined and is ranging from 1 (only one observation point feeds the extra FF), to 8 (8 observation points drives an 8-input XOR tree feeding the extra FFs).

To choose the best implementation, the eight cases are studied on each circuit. Table 3 presents the results for AES-128 core. As explained before, 63 extra scan FFs can be added to this circuit scan chain without affecting its test time. We iteratively experimented observability improvement with 1-to-8 observation points per XOR tree. For instance, with only one observation point per extra FF, the ATPG tool reduces the test sequence from 246 to 242 patterns. The best implementation corresponds to 6 observation points per extra FF, saving 11 patterns. In other words, the extra test time due to the required synchronization with PRESENT encryption/decryption, and leading us to add extra shifts on

Pipelined AES-128 ($F = 7873 = 123 \times 64 + 1$)										
	Scan	CBC (%)	Optimized version overhead (+63 FF) + CBC (%)							
#observation points per FF			1	2	3	4	5	***6***	7	8
#Patterns (K)	246	246	242	245	237	238	239	***235***	236	236
Area	367 926	+2.92	+3.10	+3.17	+3.23	+3.30	+3.36	***+3.43***	+3.51	+3.58

Table 3: Impact on number of observation points per flip-flops for test time optimization on Pipelined AES-128 circuit.

every pattern, is compensated by the reduction of the total number of patterns to achieve the same fault coverage. The extra cost for test point insertion is +3.43% compared to original scanned design, with +2.92% for scan encryption without testability optimization, i.e. without 63 extra FFs and 6-inputs XOR trees.

Experimental results on test time optimization and costs are presented in Table 4. For each circuit, the number of scanned FFs, the number K of patterns, the test time (in clock cycles) and the area (µm²) are given for four versions: the original circuit with scan chain (row *Circuit*), the circuit with the scan chain encryption (row *Circuit+CBC*), optimized version with added scan FFs connected to observability points (row *Optimized*) and optimized circuit with the scan chain encryption (row *Optimized+CBC*). Test time results for *Circuit+CBC* are compared to original test time (*Circuit*) while results for *Optimized+CBC* are compared to test time with test point insertion (*Optimized*). Area of original scanned circuit is the comparison reference for both *CBC* versions. When two implementations lead to the same pattern optimization, we choose the implementation with the smaller impact in area.

Concerning Triple-DES, four observe points per FF are used on the 24 added scan FFs. Test time increases by 0.038% compared to test time for optimized circuit. The area cost of CBC increases from 5.74% for the non-optimized method to 5.87% for the optimized one. For LEON3 processor, 107

Circuit		#SFF	#Patterns	Test time	Area
Triple-DES	Circuit	8808	77	687101	187 494
	Circuit+CBC	8808	77	+0.31%	+5.74%
	Optimized	8808+24	74	662730	
	Optimized+CBC	8808+24	74	**+0.038%**	+5.87%
Pipelined AES 128	Circuit	7873	246	1944877	367 926
	Circuit+CBC	7873	246	+0.81%	+2.92%
	Optimized	7873+63	235	1873131	
	Optimized+CBC	7873+63	235	**+0.013%**	+3.43%
RSA 1024	Circuit	16459	2393	39405239	468 415
	Circuit+CBC	16459	2393	+0.33%	+2.30%
	Optimized	16459+53	2393	39532121	
	Optimized+CBC	16459+53	2393	+0.001%	+2.51%
LEON3*	Circuit	107518	107	11612051	1902095
	Circuit+CBC	107518	107	+0.004%	+0.57%
	Optimized	107518+2	102	11074662	
	Optimized+CBC	107518+2	102	**+0.002%**	+0.57%

Table 4: Cost to use optimized CBC regarding several circuits. *: for LEON3, test time and number of patterns are evaluated to obtain a test coverage of 70%.

patterns achieve a test coverage of 70%. With 4 observe points on the 2 added scan FFs, this number of patterns decreases by 5 patterns. Test time increases by 0.002% for an area cost of 0.57%. The area cost is almost the same than the non-optimized CBC due to the large size of the CUT.

Unfortunately, insertion of test points does not allow reducing the number of test patterns on RSA. The AES-256 is not reported here since it is already optimized (its scan chain length is a multiple of 64). Extra test time for CBC is only 0.01% increase over non-encrypted scan test (see Table 2).

IV.3.b Extension to multiple scan chains design

CBC scheme can be extended to the case of multiple scan chain designs. The adaptation to multiple scan chains is however not trivial and requires the implementation of a dedicated architectural solution. The solution we propose is based on the decryption/encryption of whole slices of scan chains, while at the same time the previous slices are shifted-in or -out the scan chains.

While scan operations are performed at low-than-nominal frequencies to avoid power related problems due to the toggling of the bits in the scan chains, the encrypt/decrypt operations can be performed at higher frequencies. We exploit this feature in order to encrypt/decrypt the data of the test slices at higher frequency, while shifting the test data in the chains at lower frequency. Thus the solution operates with two clocks: the Slow-CLK for scan shift operations, and the Fast-CLK for the scan ciphers.

For the CBC implementation, we assume that the classical DfT architecture is already implemented in the circuit before the insertion of the two scan ciphers. The solution can thus be applied without modification of the multiple scan chains structure or the test compression scheme potentially used to limit the number of test interfaces (e.g. [6], [7] presented in Chapter I). Therefore, CBC has a very limited impact on the modification of the design and test flows.

Let us denote N the block size of the data encrypted by the block cipher, i.e., the number of bits of the registers R1 and R2. For instance, N can be equal to 64 for PRESENT and SKINNY-64, while it is 128 for AES. Let us assume having L parallel scan chains in the circuit, with L smaller than N (the case L>N implies the use of multiple scan ciphers, and it will be discussed at the end of this section). We suppose each scan chain being composed of Z bits. In order to adapt the proposed solution in the case of multiple scan chains, the scan ciphers registers R1 and R2 are logically divided into segments of L FFs. Instead of loading the registers serially, each segment of L bits is loaded parallel to the content of one scan slice. Once the register is filled (i.e., after N/L Slow-CLK cycles), its content is encrypted/decrypted at Fast-CLK, while the other register (which content has previously been encrypted/decrypted) is used to feed the scan chains while it is, at the same time, loaded in parallel (one segment per Slow-CLK cycle).

Let us denote k the number of segments (each of them with L bits) within the registers R1 and R2. By expressing N as $N=k \bullet L+R$ (where R=N mod L, is the number of extra bits in the case N cannot be divided by L), the registers R1 and R2 are divided in k segments of L bits plus R additional bits (with R<L). For instance, in a circuit with 6 parallel scan chains and N=64, each register can receive the test data of 10 slices, by having R=4 spare bits that will not be used as test data. Nevertheless, the

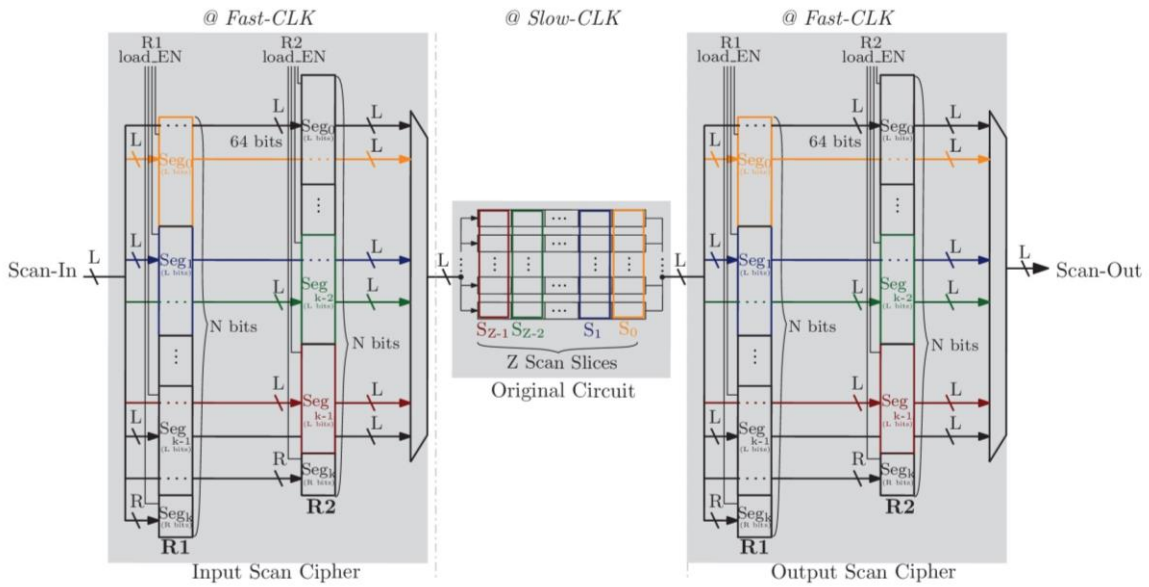


Figure 26: Proposed CBC solution applied on L multiple scan chains.

encryption/decryption algorithm requires the whole N bit values to be defined to correctly generate the expected test vectors; therefore even the last segment of R bits must be filled.

Figure 26 presents the global scheme of the proposed solution. The scan ciphers registers are decomposed of k L-bit segments and a last smaller R-bit segment. Each L-bit segment of Input Scan Cipher registers (respectively, Output Scan Cipher registers) contains the test data on one scan slice (resp., the test responses on one scan slice) after decryption (resp., before encryption). The control logic of the scan ciphers ensures that the L-bits test data are loaded into the correct segment of R1 or R2, using the corresponding *load_EN* signals. As shown in Figure 26 with a circuit composed of Z scan slices S_i, all L-bit segments are associated to one scan slice. For example, the register R1 contains the test data associated to the slice S_0 on its first segment Seg_0, and the test data associated to the slice S_1 on its second segment Seg_1. In the same manner, the register R2 contains the test data associated to the last scan slice S_{Z-1} on its segment Seg_{k-1}. The R-bit segment Seg_k of both registers contains the extra-bits in order to pad the scan slices data for decryption/encryption purpose.

For a scan cipher with N=64 (when the PRESENT or SKINNY-64 block ciphers are used for example), Figure 28 presents the partial time diagram of the Input Scan Cipher operations on a circuit with L multiple scan chains composed of 4 slices S_i and with the decomposition of scan cipher's registers on 2xL+R=64, where:

- $p_3^j p_2^j p_1^j p_0^j$ is the j[th] test pattern where each p_i^j corresponds to the data of the scan slice S_i,
- $r_3^j r_2^j r_1^j r_0^j$ is the j[th] test response where each r_i^j corresponds to the response of the slice S_i,
- $v_{i,i+1\,(R\,bits)}^j$ corresponds to the extra-bits used to pad the scan slices $p_{i\,(L\,bits)}^j$ and $p_{i+1\,(L\,bits)}^j$, in order to correctly perform the encryption,
- $E_1\left(p_{i,i+1}^j\right)_{(L\,bits)} \left| E_2\left(p_{i,i+1}^j\right)_{(L\,bits)} \right| E_3\left(p_{i,i+1}^j\right)_{(R\,bits)}$ is the result of the encryption of $p_{i\,(L\,bits)}^j |p_{i+1\,(L\,bits)}^j |v_{i,i+1\,(R\,bits)}^j$ where '|' is the operation of concatenation, and $E_n\,(x)$ is the n[th] segment composing the encrypted message.

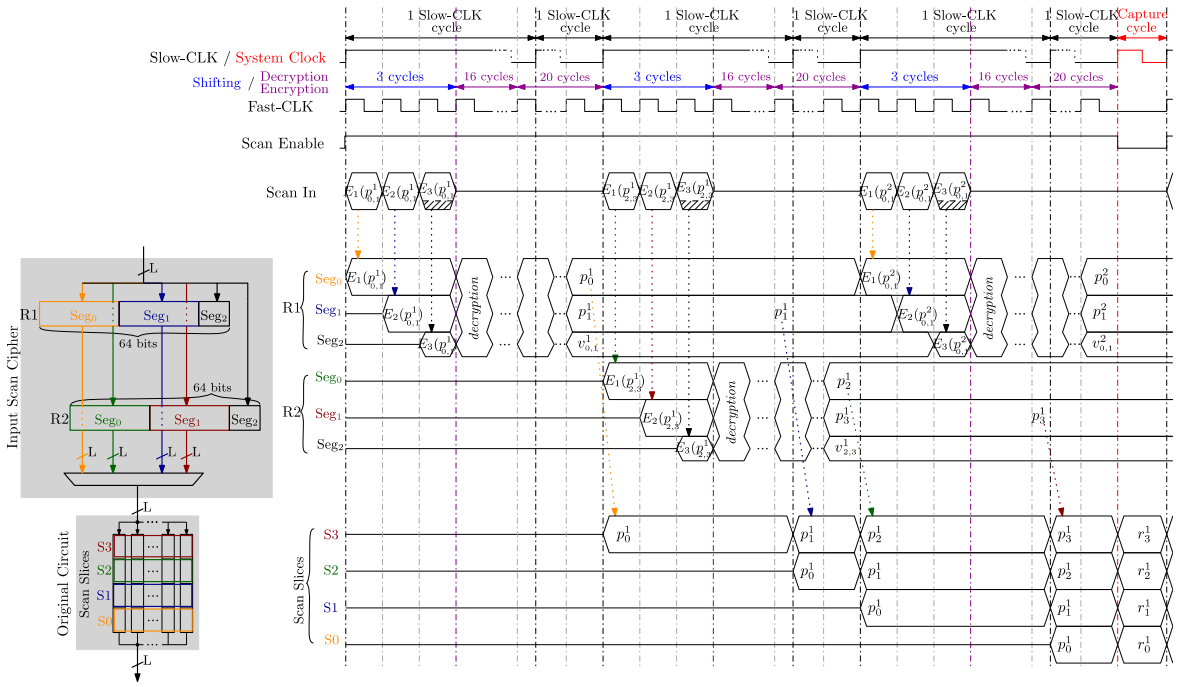


Figure 27: Time diagram of shift operations for the Input Scan Cipher in the case of CBC applied on L multiple scan chains.

At the step where the first scan slice data p_0^1 is shifted from the register R1 in the L parallel scan chains at Slow-CLK, each segment E_n ($p_{2,3}^1$), composing the encrypted message with the p_2^1 and p_3^1 scan slices data, is shifted one after the other in the second register R2 using 3 Fast-CLK cycles. Once register R2 is entirely filled with the 3 segments, the decryption of its content begins. At the next Slow-CLK cycle, the second scan slice data p_1^1 is shifted in the scan chains, and concurrently the decryption of the R2 content continues using Fast-CLK. At the end of the decryption, the p_2^1 and p_3^1 scan slices data are present in the first two segments of R2 and ready to be shifted in the scan chains at Slow-CLK. During 2 Slow-CLK cycles, the p_2^1 and p_3^1 scan slices data are shifted in the scan chains during the decryption of the next scan slices data on the other register at Fast-CLK. Once all scan slices are filled, the test responses r_3^1 r_2^1 r_1^1 r_0^1 are encrypted following the same scan operations.

Figure 27 presents the timing operations when the whole decryption/encryption operations require 36 rounds (as for SKINNY-64-128 block cipher), implemented in 36 Fast-CLK cycles. During the 2 Slow-CLK periods where the test data of 2 scan slices are shifted in the scan chain, the next 2 test slices have to be entirely decrypted using (i) 3 Fast-CLK to shift the 3 segments in the scan cipher register and (ii) 36 Fast-CLK cycles to the SKINNY decryption. Totally, the Fast-CLK period needs to be higher than or equal to 36+3=39 times 2 Slow-CLK periods.

More formally, considering a circuit composed of L multiple scan chains (L≤N), the scan ciphers registers are divided into $\left\lceil \frac{N}{L} \right\rceil$ segments. Among all segments composing the scan ciphers registers, there are $k = \left\lfloor \frac{N}{L} \right\rfloor$ L-bits segments. If $\#Rounds$ denotes the number of rounds needed to a whole decryption/encryption, $T_{Slow-CLK}$ the Slow-CLK period, $T_{Fast-CLK}$ the Fast-CLK period, the relation given in equation (3) must hold:

$$\left(\#Rounds + \left\lceil \frac{N}{L} \right\rceil\right).T_{Fast-CLK} \leq \left\lfloor \frac{N}{L} \right\rfloor.T_{Slow-CLK} \qquad (3)$$

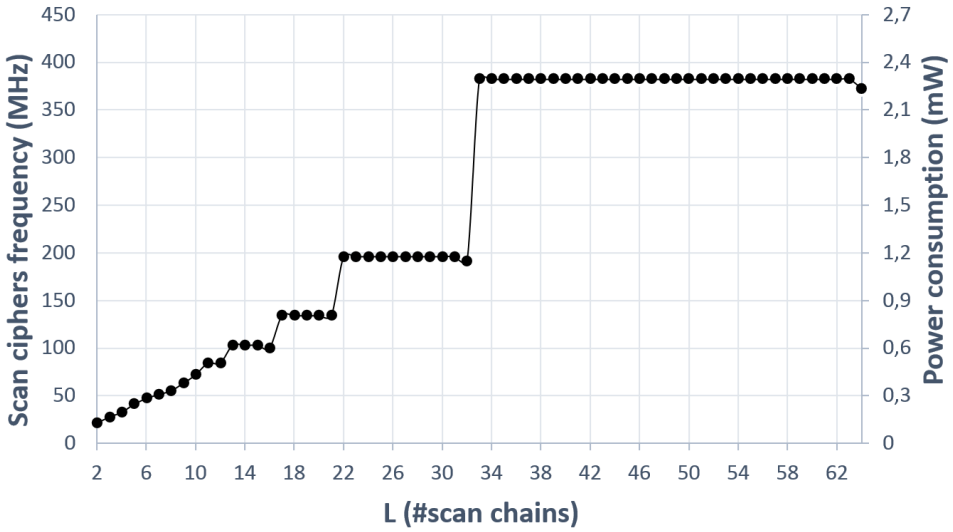


Figure 28: Scan ciphers frequency and power consumption according to the number L of scan chains (in the case of SKINNY block cipher).

In order to evaluate the minimal required scan ciphers' frequency ($1/T_{Fast-CLK}$), we use the SKINNY-64-128 block cipher (N=64, $\#Rounds$=36) as an example. From our synthesis experiments, the SKINNY implementation on 65 nm library runs at 400 MHz. Typically, test data are shifted at 10 MHz in order to limit switching activity in the circuit and thus to limit its power consumption. In the previous example described at the Figure 27, there are $\left\lceil \frac{64}{L} \right\rceil = 3$ segments with $k = \left\lfloor \frac{64}{L} \right\rfloor = 2$ L-bits segments and 1 smaller segment. The scan ciphers have therefore to operate at least at [(36+3)/2]×10=195 MHz, regarding the equation (3). More generally, Figure 28 presents the minimal required scan ciphers frequency ($1/T_{Fast-CLK}$) versus the number L of scan chains. At the minimum, the scan ciphers have to operate at least at 22 MHz in case of L=2 scan chains. At the maximum when L is between 33 and 63 scan chains, the scan ciphers frequency must reach 380 MHz.

The operations have been described so far should the number of scan chains be less than or equal to N. If the number L of scan chains is greater than N, extra scan ciphers must be implemented to deal with more scan chains. In other words, a couple of ciphers is required for securing scan-in and scan-out operations on every set of N scan chains.

The scan operations are presented in the case of a simple implementation of multiple scan chains, but it can be applied as well when a test compression method is used. Figure 29 represents the application of the CBC solution on a test compression technique. The countermeasure can be applied regardless of the test decompressor and compactor implemented, such as 2D Elastic Compression [6] or Embedded Deterministic Test [7]. The ATE generates the compressed stimuli used to test the device. These compressed stimuli are encrypted in the ATE before they are scanned into the device. The decryption performed at scan input decrypts several segments of stimuli, in the same manner as presented above. The decrypted test stimuli are then applied to the test decompressor. The encryption performed at scan output encrypts several segments of compacted responses before scanned-out. The ATE decrypts the compacted test responses in order to compare with expected ones.

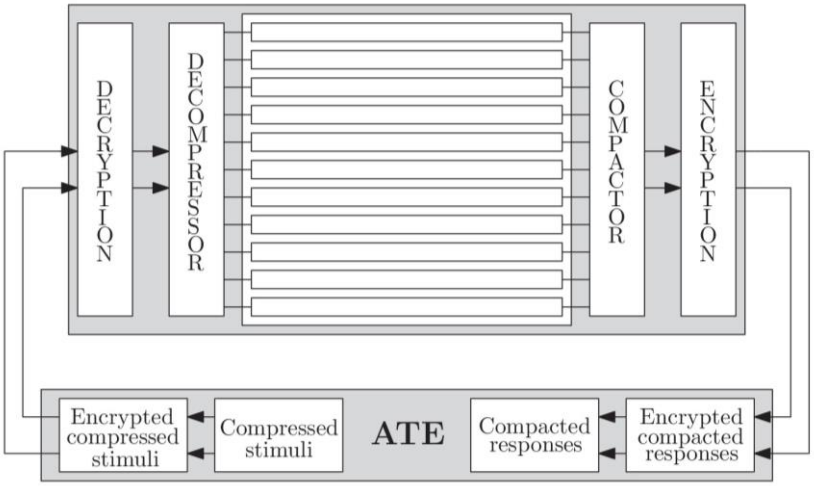


Figure 29: CBC applied when a test compression method is used.

IV.4. Proposed countermeasure based on stream cipher

Additionally to the proposed CBC, we propose a secure CSC solution compared to existing solutions. The novelty of the approach resides in the fact that the IV of the stream cipher is not a constant value, but it is randomly generated by a TRNG at every circuit reset. The use of a TRNG guarantees to never re-use the same IV for the keystream generation. By initializing the stream ciphers with a different IV, the same keystream is not generated twice to encrypt different data. The proposed solution does not present thus the two times pad limitation, preventing an attacker to carry out differential scan attacks.

Figure 30 describes the CSC consisting in adding stream ciphers at the input and the output of the scan chain. An attacker unaware of the secret key used for encryption of the test data is not able to set the circuit to a desired state, nor to plainly read the circuit state. Only users with the knowledge of the secret key can access the scan content for debugging purpose. As for the CBC, the secret key of the stream ciphers is stored and managed by the SKMU of the protected circuit.

Concerning the IV, the random value generated by the TRNG has to be known by the external tester/debugger in order to allow the correct communication to user and devices owning the correct

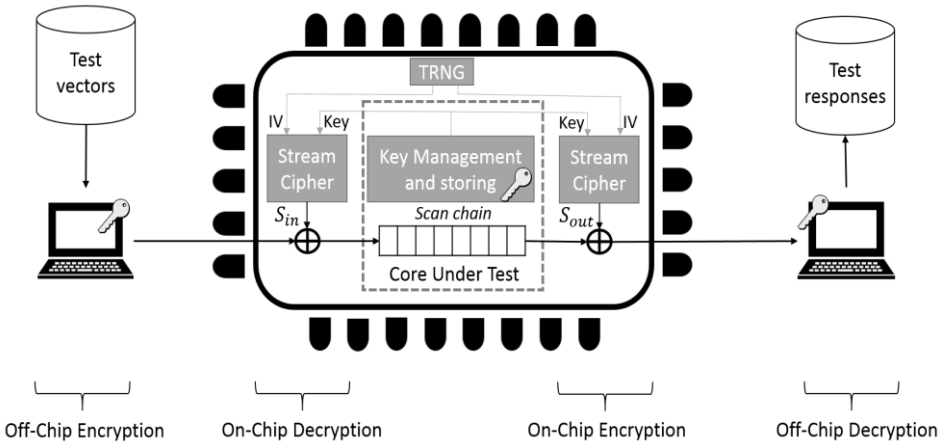


Figure 30: Principle of the proposed CSC using a random IV.

secret key. We have proposed a first solution sharing the IV from the scan chain. However, this proposal present an issue concerning the integration of the solution in a SoC. We propose then the CSC integrated in a JTAG infrastructure in order to share the IV using a dedicated instruction, fixing the integration issue. It is important to note that the security of the stream cipher is not jeopardized by making public the IV, since its secret key is still kept secret.

IV.4.a First proposal to share the IV from the scan chain

The first idea for CSC is to share the IV shifting through the scan chain. We present how it works, as well as the limitation in the integration of the solution in a SoC.

As for the state-of-the-art, we propose to encrypt the scan chain with the TRIVIUM stream cipher, generating a keystream from an 80-bit secret key and an 80-bit IV. First, this stream cipher presents a low-cost implementation. Secondly, an alternative implementation allows increasing the keystream throughput (see Figure 31). Instead of implementing one stream cipher at scan-in and another at scan-out, the implementation of one TRIVIUM is sufficient to generate two different keystreams for a marginal additional cost of 3 AND gates and 11 XOR gates [59]. One TRIVIUM stream cipher generates thus the keystream S_{in} for the decryption process of the test vectors at scan input and the keystream S_{out} for the encryption process of the test responses at scan output.

The global architecture of the first proposed CSC is represented in Figure 31. The scan chain encryption is composed of a TRNG, a shift register containing the IV, the stream cipher and the control unit. The control unit manages the initialization process, corresponding to the generation of the IV before beginning the encryption phase.

After a circuit reset, the control unit starts the initialization as soon as the circuit switches from normal mode to test mode. During the entire initialization process, the scan chain is kept inaccessible since the stream cipher does not generate the keystreams. Both the scan input SI and the scan output SO are connected to the TRNG, thanks to the implemented multiplexers. This way, an attacker is not able to send desired data, since the circuit scan chain is connected to the random bitstream generated by the TRNG. An attacker is also not able to observe the internal states of the circuit. He observes the

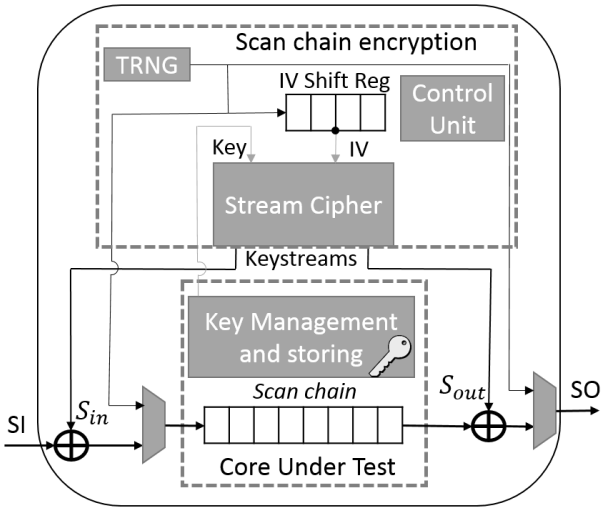


Figure 31: Architecture of the first proposed CSC using a random IV.

bitstream generated by the TRNG, including the IV value. The IV is firstly stored in the shift register, before initializing the stream cipher. Once the stream cipher setup is finished, the multiplexers are switched to the SI and SO. The stream cipher then generates the two keystreams. The keystream S_{in} is XORed bitwise to the data (test vectors) at scan input and the keystream S_{out} is XORed bitwise to the test responses of the circuit. The control unit manages the stream cipher encryption during test mode. If the circuit switches to normal mode, the control unit stops the scan encryption. As soon as the test mode is asserted again, the stream cipher resumes the keystreams generation.

The encryption with a stream cipher does not cause issues with test procedure for both fault models: stuck-at faults and transition delay faults. Indeed, test data has not to be padded contrary to CBC solution, since stream cipher encrypts/decrypts data bitwise.

The main problem with the architecture of the proposed CSC in Figure 31 is the integration of the solution in a SoC. Indeed, the protected device is connected to the others devices in daisy-chain configuration. Therefore, the random IV generated by the TRNG is shifted through all the devices belonging to the test daisy-chain, implying possible issues for the other devices. For instance, if a device in the daisy-chain is being tested at the same time of the CSC initialization process, the random bitstream generated by the TRNG is thus shifted to the scan chain of the device being tested, perturbing the test.

IV.4.b Proposal of CSC integrated in JTAG infrastructure

In order to share the random IV to external user, we propose a better solution than the previous one, consisting in adding a specific TDR to the JTAG instruction set. This special register contains the IV value that the user can read executing the custom instruction called *GETIV* (more details are given thereafter). With this method, the generated random IV is shared with the external world, making the value publicly known. Even if an attacker can read the IV, the security is not compromised since the key of the stream cipher remains secret.

For illustrating the principle of the solution, we consider a SoC embedding a crypto-processor, hence susceptible to be the target of differential scan attacks. An attacker can shift in and out the scan content of the circuit using the *INTEST* instruction, provided by the JTAG standard. The proposed CSC is effective in protecting against a malicious use of the *INTEST* instruction. We suppose that, when the INTEST instruction is executed, the TDR that is connected between the TDI and TDO signals is the internal scan chain. In the proposed solution, the content of data, shifted through the device, after the execution of the *INTEST* instruction, is encrypted with the stream cipher. As we have already mentioned, the generation of the IV is performed randomly. This means that every single device will generate a different random value, thus reducing the efficiency of the manufacturing test. Indeed, after the production of the ICs, many dies are usually tested in parallel, directly probing the silicon wafer, using the same test patterns. If the proposed scheme is also activated in this phase, parallel testing is not possible anymore since each circuit requires the test patterns encrypted with a different keystream. We explain firstly how to use the proposed CSC to easily perform parallel wafer testing, while we explain secondly how the CSC is used in mission mode.

IV.4.b.i Wafer testing

The first test of newly fabricated integrated circuits is performed when the dies are still part of the silicon wafer. In order to gain test time during manufacturing test, several dies are tested in parallel by applying the same patterns at the same time.

As explained above, implementing the proposed encryption technique makes the test patterns to be applied become unique for every single circuit, because they must be encrypted resorting to a random number that differs from one circuit to the other. Therefore, the proposed CSC cannot be used for parallel testing of multiple circuits.

To thwart this disadvantage, we propose to disable the use of the TRNG during the manufacturing process and to use a predefined hardwired IV for all circuits. In this way, all keystreams are identical and all test patterns can be encrypted in the same way. To bypass the TRNG when the circuit is still on the wafer, we propose to use in-wafer sensors, able to identify whether the die is still part of the wafer. These sensors are either based on One Time Programmable memories, or on the so-called Saw Bow. The latter is based on an electrical connection made by strong pull-up and weak pull-down elements, which are physically interconnected by a metal line across the sawing lines of the wafer. The strong pull-up resistance sets a logic value on the line when the sawing line is intact. When the dies are sawed, the weak pull-down resistance sets the opposite value on the line [67].

From a security point of view, disabling the TRNG at manufacturing stage does not represent a security issue. As seen in Chapter II with the threats throughout the supply chain, no threat is considered at the fabrication stage. Therefore, a secure device does not need protection yet against attacks using the test interface.

IV.4.b.ii Mission mode

When in mission mode, the circuit can be the target of an attack. The scan chain is accessible to external users via the JTAG interface using the *INTEST* instruction. The principle of the solution is to encrypt the scan chain content using the stream cipher, as illustrated in Figure 32.

As for the first proposal, a single TRIVIUM stream cipher generates two keystreams: S_{in} for decrypting the test data shifted into the scan chain, and S_{out} for encrypting the test data shifted out of the scan chain. The utilization of the proposed CSC also consists in an initialization phase and a successive encryption phase. However, both phases differ a little bit compared to the first proposal, since the proposed CSC is integrated within the JTAG infrastructure.

During the initialization phase, the TRNG generates the IV, and sends it to the circuit implementing the stream cipher to perform its setup. During this phase, the TAP controller locks the use of the *INTEST* instruction. If an external user requires this instruction, the TAP controller remains set on bypass mode. When the initialization phase is completed, the tester can ask for access to the protected instruction. The external user has to execute a specific instruction, called *GETIV*. When executed, this instruction connects a special register, containing the generated IV value, to the TDI/TDO signals. This way, the tester can shift out of the device the IV that has been produced by the TRNG during the initialization phase. If the *GETIV* instruction is executed before the initialization phase is completed, a sequence of all ‘0s’ is returned as response.

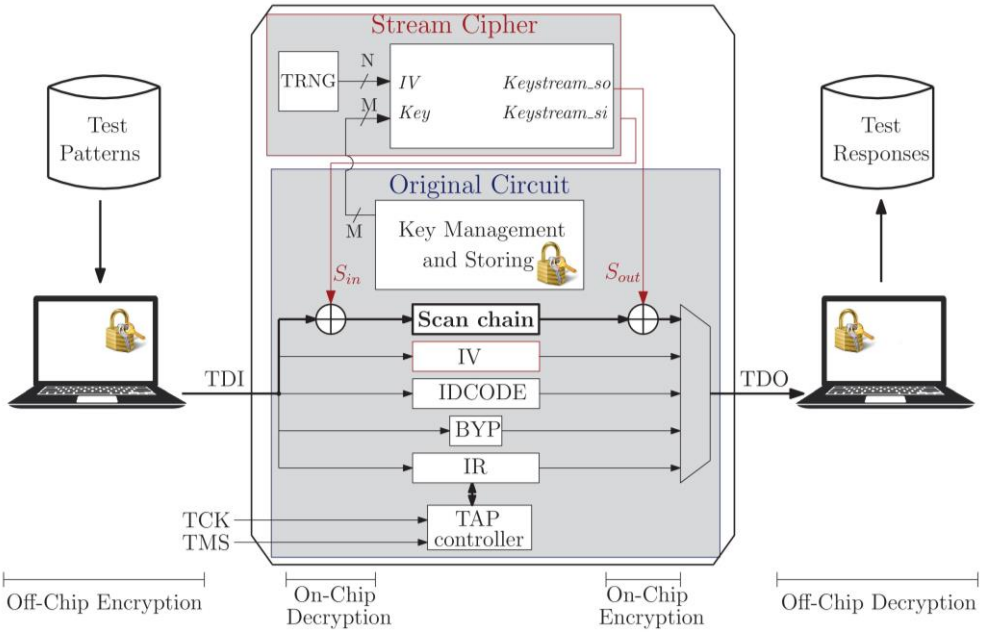


Figure 32: Basic scheme of CSC on single scan chain integrated in JTAG infrastructure.

During the encryption phase, the tester knows both the secret key (if the tester is authorized) and the IV obtained via the *GETIV* instruction. From this moment, it is possible to encrypt off-chip the test patterns using the IV recovered from the device. At this point, the test patterns shifted through the TDI for the *INTEST* instruction are decrypted on-chip before being introduced into the corresponding TDR. During the scanning-out operation, the test responses are encrypted on-chip. The tester collects encrypted responses from the TDO interface that can be decrypted off-chip, using the same IV and secret key used for the off-chip encryption.

IV.4.c Implementation of CSC integrated in JTAG infrastructure

We present in this Section the implementation of the solution with details on the modifications made on the JTAG test wrapper, and on the control unit managing the initialization phase. Moreover, we show that the solution is not limited to classical testing purposes with the protection of the *INTEST* instruction and the scan chains, but the countermeasure can be extended to a whole set of protected instructions whose involved data can benefit from encryption.

IV.4.c.i General architecture

The proposed CSC implies some modifications on the original JTAG wrapper, as illustrated in Figure 33. The JTAG TAP controller is modified to add the new instruction *GETIV* and the associated TDR containing the IV value.

The designer has to define a set of protected instructions. Only the data shifted through the TDRs associated to these instructions are encrypted by the stream cipher. Since the decrypting keystream S_{in} and the encrypting keystream S_{out} are not correlated, it is necessary that the ciphering is not active when the device is in bypass mode. Indeed, the data D shifted through TDI and TDO signals, when the countermeasure is activated, results into $D \oplus S_{in} \oplus S_{out}$, with $S_{in} \neq S_{out}$. When the protected device is integrated into a JTAG daisy-chain, with the BYP and IR registers not encrypted, the downstream

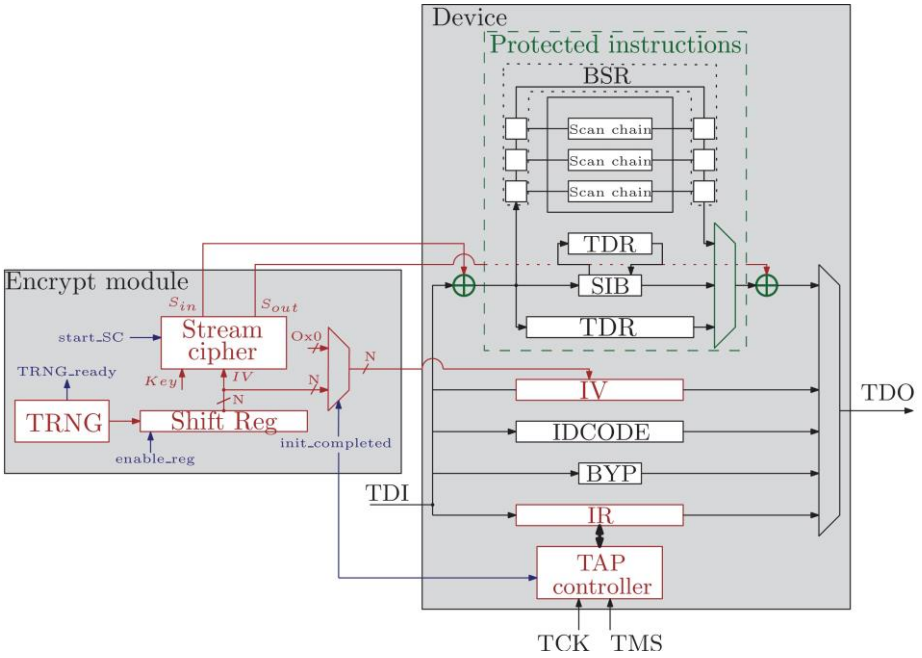


Figure 33: Detailed architecture of the CSC solution.

devices in the daisy-chain receives the correct data without having them perturbed. A device supporting the stream cipher encryption can thus be successfully integrated into a JTAG daisy-chain, while granting protection of the confidential data contained into its sensible TDRs.

For example, in Figure 33 the protected instructions can include (1) the *EXTEST* and *INTEST* instructions with the encryption of the BSR and the internal scan chains, (2) the IJTAG instruction accessing the RSN including critical instruments, and (3) any instruction accessing a TDR containing confidential data, such as firmware updates of the device.

During the initialization phase, these protected instructions are not accessible since they are replaced by the *BYPASS* instruction. The TDR associated to the *GETIV* instruction contains only '0s' during this phase. The signal *init_completed*, managed by the control unit, is in charge of enabling the access to the encrypted TDRs and the *GETIV* execution. This signal is not accessible from the external, but the delay introduced by the initialization process can be known from an external tester thanks to the datasheet of the protected circuit. This delay, varying in function of the TRNG and the stream cipher implementation, indicates the waiting time until the circuit enters the encryption phase.

During the encryption phase, the protected instructions are accessible. The stream cipher encrypts the content of data addressed to the TDRs associated to the protected instructions. An authorized user knows the secret key. The random IV is fetched thanks to the *GETIV* instruction. The encryption is employed in the examples illustrated in Figure 33: (1) all data passing through the BSR are encrypted preventing the observation and the control of the scan content; (2) the RSN is also encrypted; (3) the firmware is decrypted before being saved into the memory.

When the encryption is performed on an IJTAG network, an attacker without the knowledge of the RSN configuration, faces troubles to open or close the SIBs, due to the keystream XORed with the test data at the RSN input. Even if the attacker is able to configure the RSN in a chosen configuration, he/she is not able to send and read data related to confidential instruments, since the data shifted through the RSN go through the input decryption and the output encryption.

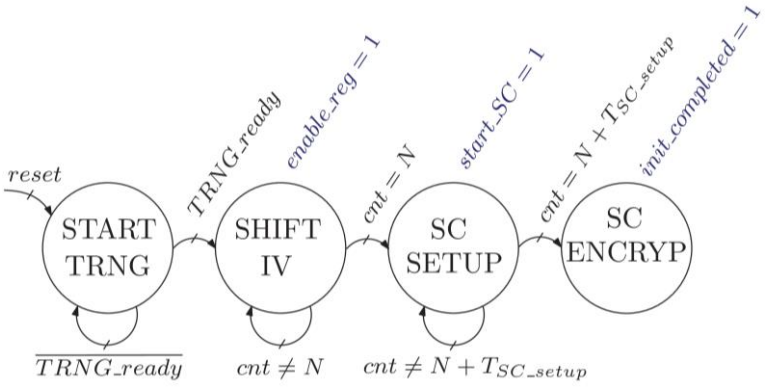


Figure 34: Finite State Machine controlling the initialization procedure of the CSC solution.

When the test interface is used to configure a memory (e.g. for firmware updates), the encryption ensures that the content cannot be readable without knowing the secret key. In the same manner, a FPGA configuration process is protected from sniffing [26]. Moreover, an attacker cannot update the firmware with a corrupted version due to the decryption performed on the data sent through the test interface.

IV.4.c.ii Control Unit

The procedure to initialize the stream cipher is controlled by a FSM, whose state transition graph is given in Figure 34. The FSM is composed of 4 states (START_TRNG, SHIFT_IV, SC_SETUP, and SC_ENCRYPT) and its outputs three control signals ($enable\_reg$, $start\_SC$ and $init\_completed$). All of them are initialized to '0'.

At reset, the TRNG starts the initialization, while in the START_TRNG state. TRNGs have usually an initial set-up time during which the generated numbers are not random enough. Therefore, they require some time to reach sufficient entropy. During this period, the generated value cannot be used. As soon as the TRNG reaches a good entropy ($TRNG\_ready = '1'$), the IV generation begins.

During the SHIFT_IV state, the shift register (Shift Reg in Figure 33) receives the random bitstream generated by the TRNG ($enable\_reg = '1'$). A counter cnt is launched at the same time. When cnt reaches the value N, the TRNG stops generating the random bitstream. N is equal to the number of bits of the IV. When the N bits of the random IV are generated, the TRNG is no longer used and it becomes available to other applications, if needed. Otherwise, it can be turned off.

Once the counter have reached the value N, the control unit goes to the SC_SETUP state and starts the stream cipher initialization ($start\_SC = '1'$). The FSM stays in this state during the time $T_{SC_{setup}}$, needed for the stream cipher setup.

Once the counter reaches $N + T_{SC_{setup}}$, the initialization process is completed ($init\_completed = '1'$). The stream cipher encrypts the data passing through the TDI and TDO terminals of the protected TDRs. The keystreams are generated only in the case in which the TAP controller is in the *Shift-DR* state and a protected TDR is selected by the instruction under execution. In the other cases, the encryption is not needed and the stream cipher is deactivated and it generates no keystream.

During the SC_ENCRYPT state, when the tester wants to write into a protected TDR, it executes firstly the *GETIV* instruction to read the IV in order to encrypt the data using the shared secret. The

tester executes then the protected instruction to access the corresponding TDR. The tester places the IC into *Shift-DR* state where the stream cipher generates the keystream. The tester shifts in the encrypted data, which are decrypted before being sent to the TDR. The tester places then the IC into *Exit-DR* state, in which the stream cipher stops the keystream generation. At the end of the operations, the TDR contains the data in plaintext.

The initialization process cannot be interrupted. The control unit ensures the setup completion before any possible operations on the protected TDRs. If a circuit reset occurs, the control unit is reinitialized and the TRNG generates a new IV for the stream cipher.

The control unit enables the stream cipher as soon as an operation on a protected TDR is required. This prevents any clear bitstream to be inserted in input or observed from the output. Even during the initialization process, the controller ensures that no test data can be shifted in and out the protected TDR by forcing the executed instruction to *BYPASS*.

IV.4.c.iii Overheads compared to original JTAG test wrapper

Proposed CSC architecture has a cost in terms of area and test time compared to the original test JTAG wrapper. To evaluate the area overhead, we have considered a simple JTAG wrapper implementing a TAP controller, the IR, the BYP, and the IDCode registers. The CSC solution implies modification on the JTAG wrapper to include the *GETIV* instruction and its associated register. Moreover, some modules are added in addition to the modified JTAG wrapper: the TRIVIUM stream cipher, the shift register containing the random IV and the control unit. Table 5 reports the area cost of the proposed CSC compared to the original JTAG wrapper, representing an area overhead of 500%. The solution is thus dedicated to large devices, such as SoC designs.

Modules	Original JTAG (GEs)	Proposed solution (GEs)
JTAG wrapper	625	1 147
TRIVIUM	/	2 048
IV Shift Register	/	300
Control Unit	/	252
Total	**625**	**3 747**

Table 5: Area cost of the proposed CSC compared to the original JTAG wrapper.

Concerning the test time cost, the proposed solution introduces only an overhead due to the initialization process. This process takes some time to initialize the TRNG, $N = 80$ clock cycles to shift the random IV into the shift register, and $T_{SC\_setup} = 1152$ clock cycles for the TRIVIUM setup. After this initialization process, the tester has to recover the IV executing the *GETIV* instruction before starting the encrypted test communication with the device. This corresponds to 80 clock cycles to shift out the content of the IV register. Basically, in addition to the time required to generate a random number, the solution implies a test time overhead of 1312 clock cycles at the beginning of a test procedure. This test time overhead has to be compared with the whole test sequence of the CUT. Experimentations have been conducted on several circuits in Chapter V.

IV.4.c.iv Extension to multiple scan chains

The CSC can be directly adapted to multiple scan chains, regardless of the test compression technique. In the case of test decompressor and compressor implemented at scan pins, the ATE generates compressed stimuli used to test the circuit under test. The ATE encrypts, by stream ciphering, these generated stimuli following the same test procedure as described before. The encrypted compressed stimuli are scanned in the circuit and decrypted with the keystreams generated for the scan-inputs. The decrypted test stimuli are then applied to the test decompressor. The test responses are compressed before being encrypted on-chip with the keystreams generated for the scan-outputs. Finally, the ATE decrypts the compacted test responses in order to compare them with the expected ones.

The only limitation for the extension to multiple scan chains is the number of keystreams that the stream cipher is able to generate. Considering a circuit where multiple scan chains are accessible through L scan-inputs and L scan-outputs, as illustrated in Figure 35, the stream cipher has to produce 2·L-bits keystreams.

The number of possible keystreams depends on the used stream cipher. For instance, TRIVIUM can compute up to 64 keystream bits in one clock cycle. Therefore, 32 parallel test data can be decrypted at scan-inputs and 32 parallel test data encrypted at scan-outputs. Considering block ciphers in CTR mode, this number is fixed by the ratio between the encrypted block size and the number of rounds for the crypto-algorithm (i.e. the number of clock cycles needed to encrypt a block). For PRESENT, the encryption of 64 bits is done in 31 rounds. As a consequence, PRESENT is able to generate two bits of the keystream in one clock cycle. Thus, the solution with PRESENT CTR can only be applied on a single scan chain. Considering the AES-128 CTR, 128 bits are encrypted in 10 rounds. The stream cipher is therefore able to generate 12 keystreams in one clock cycle, encrypting up to 6 scan chains.

Apart from the limit on the number of keystreams, other perspectives can be considered to increase the number of scan chains to be encrypted. One of them is to run the stream cipher at higher frequency

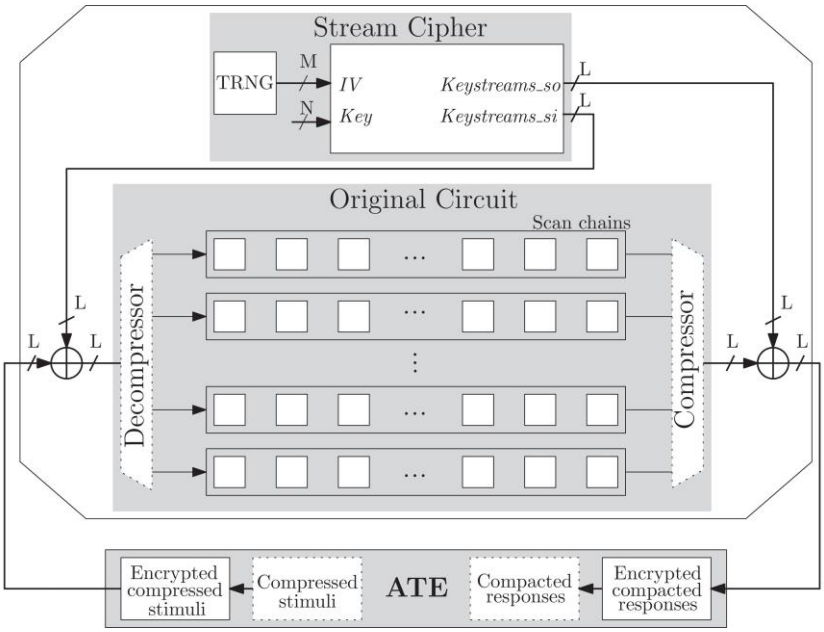


Figure 35: CSC solution applied on L multiple scan chains regardless of test compression.

in order to increase its throughput. Due to the low combinational complexity of the stream cipher, the frequency increase is a potential solution. Another solution is to implement several stream ciphers even though a bigger area overhead needs to be tolerated. For both perspectives, the cost in power consumption increases also in the same way.

IV.5. Conclusion on the new proposed countermeasures

The access to a scan network can be exploited by an attacker to carry out scan attacks, to exploit the JTAG features (illegal debugging, updating corrupted firmware, and exploiting IJTAG network), and to insert malicious cores within the test chain. To prevent these threats, a solution consists in encrypting the scan network. Several solutions have been proposed based on stream cipher. However, they present the weakness of using several times the same keystream to encrypt the test data. In this chapter, we have studied a solution based on stream cipher encryption without this vulnerability, and another based on lightweight block ciphers. The scan content is encrypted with a secret key developed for the current activity, and shared with the authorized users using the key management already present in the circuit. Both solutions can be applied on single and multiple scan chains configuration.

These countermeasures have been published in the journal Transactions on Computer-Aided Design of Integrated Circuits and Systems [69]. In addition, we have presented these works in several European and international conferences [70]–[72], and in several workshops [76]–[81]. In the next chapter, we compare the proposed scan encryption solutions to the state-of-the-art countermeasures. Additionally, we draw a comparison between both countermeasures.

Chapter V

Features and comparison of the two proposed countermeasures

Summary

V.1. Global advantages of scan encryption compared to the state-of-the-art

In this chapter, we evaluate the two solutions presented in Chapter IV. Figure 36 resumes the architecture of both scan encryption techniques, CBC and CSC. Firstly, we perform a security evaluation of scan encryption countermeasures regarding the considered threat model with both external and internal attackers. Secondly, we evaluate the characteristics and costs of the proposed solutions. Thanks to these evaluations, we compare CBC and CSC solutions with the existing countermeasures preventing against attacks through the test infrastructures, presented in Chapter III.

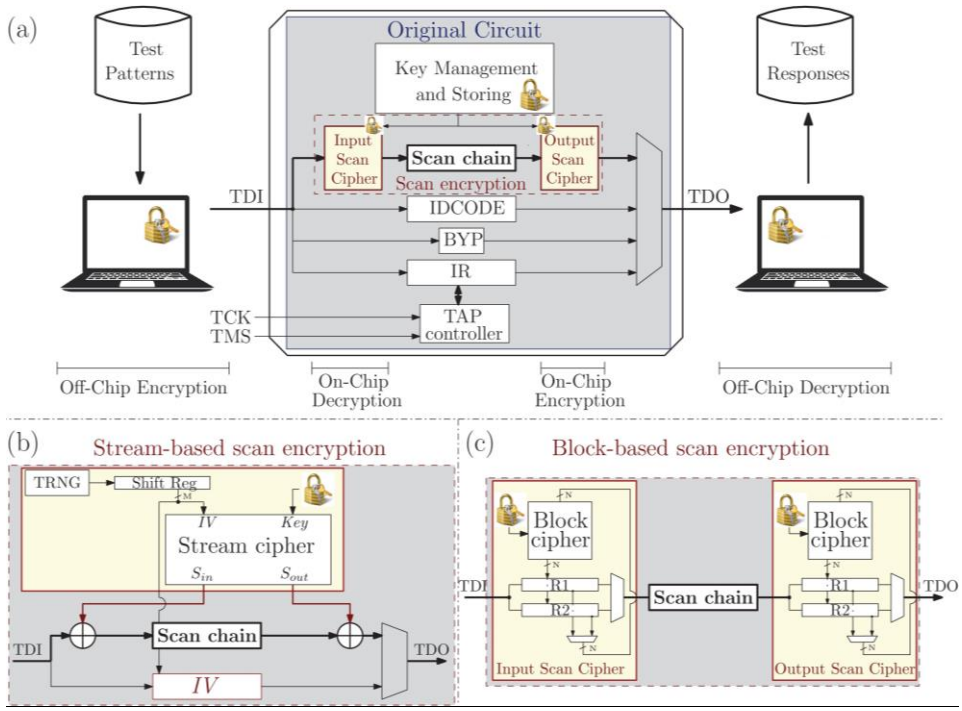


Figure 36: (a) Global architecture of the scan encryption countermeasures. (b) CSC implementation. (c) CBC implementation.

V.I.a Security evaluation

The scan encryption countermeasures are evaluated regarding the threat model described in Chapter II. Both solutions, CBC and CSC, are considered.

An external attacker cannot carry out scan attacks, exploit the JTAG features (debugging, uploading firmware, accessing IJTAG network), since the scan network is encrypted. The decryption performed on the scanning-in test data prevents the setup of desired values in the scan chain without knowing the secret key. The encryption performed on the scanning-out test responses prevents the observation of the internal states of the circuit under test without firstly performing the decryption. The controller of the scan encryption enables the stream cipher as soon as a scan operation is required to prevent any clear bit stream from being inserted or observed.

The proposed CSC does not present the same vulnerability as the previous countermeasures in [30], [60] and [61] concerning the differential scan attacks. The seed used to initialize the stream cipher is randomly generated, consequently encrypting with a different keystream at each reset of the circuit. Therefore, differential scan attacks [13]–[16] are not possible. Considering the CBC, it does not present the possible vulnerabilities of the state-of-the-art CSC, since the countermeasure is based on block ciphers, consequently being a more secure solution.

Both proposals also prevent the exploitation of the scan chain that would help to reverse-engineer an IP. In FPGA, the configuration bit stream is a critical information (IP confidentiality) that can also be protected thanks to encryption [26]. In the same manner with the proposed countermeasure, the on-chip decryption of that bit stream before use prevents IP definition analysis. In addition, encryption at scan output prevents scanning out critical data related to the IP design.

Moreover, several possible implementations of the proposed scan encryption mechanisms allows to tune the granularity of the target components. As described for the CSC solution in Chapter IV, specific TDRs can be encrypted, protecting the associated instruments. The proposed solutions can also be implemented either to secure the scan chain of a core in SoC or to secure specific instruments in IJTAG network, guaranteeing possible fine-grained access to cores and instruments.

The proposed schemes allows to distinguish between different groups of users. A first set of instructions can be encrypted with a key associated to one group, and another set of instructions can be encrypted with another key associated to the second group.

Concerning internal attacks, proposed scan encryption countermeasures ensures the confidentiality of the test communication, protecting against a malicious core possibly sniffing test data on shared test paths. However, the integrity of the communication is not ensured, allowing a malicious component in a test daisy-chain to tamper test data (patterns or responses) related to another component. Encryption however prevents chosen-plaintext and known-plaintext attacks as far as the secret key is not known from the attacker, which severely limits the attack possibilities.

Additionally, the secret key embedded in the device and used for encrypting the test communication allows to ensure the device authentication in order to prevent overproduction. If an authorized user receives test responses encrypted with a “wrong” key, he/she knows that the circuit is a counterfeit.

In terms of security, CBC and CSC both protect against the considered threat model, except against probing attacks. However, such attacks require important resources from an attacker. In the case where the secure device needs to be protected against invasive attacks, a dedicated countermeasure against probing attacks can be implemented and combined with the proposed scan encryption.

Table 6 resumes the security evaluation of the proposed scan encryption techniques and compares with the state-of-the-art. Compared to existing countermeasures, proposed CBC and CSC ensures the largest protection, against external threats as well as against internal threats.

V.I.b Characteristics and costs evaluation

In addition to the security ensured by the proposed CBC and CSC, these solutions present several advantages compared to the state-of-the-art countermeasures.

From the testability point of view, the solutions preserve in-field diagnosis and debug contrary to the countermeasures consisting to avoid scan chain, such as disconnecting the test accesses or using BIST engines [36]. Indeed, an authorized user knowing the secret key can debug a system using the scan chain, by setting the system in a desired state and by reading plain test responses. Regarding the fault coverage, the scan encryption countermeasures do not impact test efficiency achieved on the original circuit since deterministic and structural test patterns remains the same (i.e. the original test sequence is applied on the target module after on-chip decryption). Additionally, test responses allow to achieve the same fault coverage and diagnostic levels. There is no extra cost (test patterns, test time) related to the extra scan ciphers embedded with the device under test because, as shown in Section V.3, these extra logic is freely tested during the test procedure of the original circuit. As described in Chapter IV, scan encryption countermeasures have a test time overhead, composing of only the initialization procedure for CSC, and eventually padding of the data for CBC. In both solutions, the test time overhead represents a marginal cost, since even for CBC we propose an optimization of the solution consisting in inserting observation points.

Concerning area and power consumption overheads, the proposed solutions implementing stream cipher or lightweight block ciphers present a cheaper cost compared to the secure test access based on challenge/response protocol [51]–[58] implementing TRNG, hash engines, and asymmetric or symmetric cryptography requiring a lot of resources. Even for CSC, we propose to re-use a TRNG already available on the chip in order to limit the area and power consumption cost. These costs are detailed in Section V.2.

Another advantage compared to the secure test access countermeasures [49]–[58] is the secure key management. A sharing method needs to be implemented for the passwords or challenge/response pairs, while we propose to re-use the SKMU already implemented for the crypto-processors in the chip.

Compared to secure scan designs countermeasures [40]–[45], the proposed CBC and CSC do not modify the scan structure, implying two main advantages. Firstly, CBC and CSC are applicable on non-modifiable cores (black-block IPs) by just adding the scan ciphers and the control unit. Secondly, the proposed countermeasures have a low impact on the DfT flow by adding the scan ciphers on the netlist file with scan design, and by adding the TDR containing the IV in the JTAG wrapper, as well as the *GETIV* instruction in the JTAG instruction set for CSC.

Overall, the proposed solutions present several advantages compared to the existing countermeasures in literature, resumed in Table 6.

In the following, we give a comparison between the two proposed countermeasures, CBC and CSC, with an evaluation of the implementation costs on single as well as multiple scan chains, testability evaluation, and integration in a SoC design.

Solutions	Security evaluation						Characteristics and costs evaluation									
	External threats				Internal threats		Testability			Applicability						
	Scan attacks	Exploiting JTAG features	Users with a restricted access	Probing attacks	Malicious component	Counterfeit component	Fault coverage	In-field diagnosis and debug	Test time cost	Implementation	Area cost	Power cost	Impact on DfT flow	Test procedure	Non-modifiable core	Secret sharing
Scan chain avoidance Blowing fuses	Secure	Not available	Not available	Insecure (probe on fuses)	Secure	Insecure	Same	Impossible	/	Fuses	0	0	No	Standard before blowing	Applicable	No
Scan chain avoidance BIST [36][39]	Secure	Insecure	Not available	Insecure	Secure (against sniffing)	Insecure	Can decrease	Complicated	Possible overhead	PRNG, Responses compactor	Moderate	n.s.	Important (insertion BIST)	BIST procedure	Not applicable	No
Secure scan design [13], [40], [44], [45]	Secure	Insecure	Not available	Insecure	Insecure	Insecure	Same	Preserved	Low	Modification of the scan chains	Low	n.s.	Moderate	Few changes	Not applicable	Depends
Secure scan design [41], [42], [43]	Insecure	Insecure	Not available	Secure	Insecure	Insecure	Same	Preserved	Low	Modification of the scan chains	Low	Low	Important	Standard	Not applicable	No
Secure scan design (IJTAG) [46], [47], [48]	Insecure	Secure (against exploiting RSN)	Fine-grained access to **instruments**	Insecure	Insecure	Insecure	n.s.	Preserved	Low	Modification of the RSN	Low	n.s.	Low	Unlocking	Applicable	Challenge response pairs or key (LSIB)
Secure test access Password [49], [50]	Secure (but replay attacks)	Secure	Fine-grained access to **cores** [50]	Insecure	Insecure	Insecure	n.s.	Preserved	Low	Password storage and control	Moderate	n.s.	Low	Unlocking	Applicable	Password
Secure test access Challenge/response [51], [53], [54], [55], [56], [57], [58]	Secure	Secure	Fine-grained access	Insecure	Insecure	Insecure	n.s.	Preserved	Important	TRNG, hash, asymmetric or symmetric ciphers or PUF	Important	n.s.	Low	Unlocking	Applicable	Challenge response pairs (server)
Secure test access Previous CSC [30], [60], [61]	Insecure (two times pad)	Insecure (two times pad)	Fine-grained access	Insecure	Secure (integrity in [30])	Secure	n.s.	Preserved	Low	TRIVIUM	Moderate	n.s.	Low	Encrypting test data	Applicable	Secret key and IV
***Secure test access* Proposed CBC and CSC**	**Secure**	**Secure**	**Fine-grained access**	**Insecure**	**Secure (against sniffing)**	**Secure**	**Same**	**Preserved**	**Low**	**TRIVIUM or lightweight block ciphers**	**Moderate**	**Low**	**Low**	**Encrypting test data**	**Applicable**	**Re-use SKMU**

Table 6: Comparison of the proposed scan chain encryptions to the state-of-the-art (red: drawback, green: benefit).

V.2. Implementation costs

First, we compare the solutions on several benchmarks, namely a triple-DES, a pipelined 128 bits AES, a pipelined 256 bits AES, a 1024 bits RSA and a processor LEON3. In the first experiments, all the circuits are equipped with a single scan chain.

These benchmarks as well as the ciphers have been synthesized using a 65-nm library. We have implemented the CSC with TRIVIUM [59], PRESENT [64] in CTR mode, and AES [12] in CTR mode. For CBC, we have used PRESENT and SKINNY [65].

Concerning the CSC solution, we do not consider the cost to implement a TRNG. As seen in Chapter IV, in the case where a TRNG is already implemented for the functional mode of the original circuit, the proposed countermeasure can exploit this TRNG during the test mode, implying no overhead cost for the random number generation. When a TRNG has to be implemented, the related cost is evaluated as 15,000 GE from the Synopsys DesignWare IP library [68] representing about 31 200 µm² using the 65-nm library adopted for the experiments. This TRNG is composed of a whitening circuit with a noise source used to seed a random number stream, classifying the random generator as a Non-deterministic Random Bit Generator.

V.2.a On single scan chain

Area overheads are reported in Table 7. It must be noted that the CBC requires the implementation of two ciphers, one for decrypting test patterns, and the other for encrypting test responses. Conversely, the CSC only requires the implementation of one cipher that delivers two keystreams. The area overhead is to be compared with the original CUT. As it can be seen, both solutions are expensive in terms of area, making these solutions suitable to large designs only. Furthermore, in our experiments, we have not considered the key management using the one adopted in the benchmark under test. Thus, this solution is particularly adequate when at least one crypto-core is implemented in the circuit under test.

Concerning test time, we consider for each core the test sequence for stuck-at faults obtained thanks to the TetraMAX ATPG tool. Table 7 reports the test time overhead related to the scan encryption countermeasures. The test time overhead for CSC is due to the initialization phase: the TRNG initialization, the shift of the random IV, and the stream cipher setup. Without considering the TRNG initialization, we obtain respectively 138, 95 and 1232 extra clock cycles for AES-128 CTR, PRESENT-128 CTR, and TRIVIUM. In any case, they represent a marginal cost of 0.022% in average over all the experimentations (see *Overhead (%)* in CSC section, Table 7).

As presented in Chapter IV for CBC, every pattern has to be padded in such a way that its total length is a multiple of the block size of the cipher, i.e. 64 bits in the case of PRESENT and SKINNY. This induces a test time overhead for each pattern, representing a cost of 0.29% in average over all the experimentations (see *Overhead (%)* in CBC section, Table 7). This overhead can be reduced thanks to the presented optimization (i.e. insertion of observation points), except for pipelined AES-256 core which has a scan chain length multiple of the block size N=64 and thus does not require any pattern

padding. With the optimization of the CBC solution, test time cost is reduced to 0.0135% in average over all the experimentations (see *Overhead (%)* in CBC section for *Optimization* row, Table 7).

It is clear that CSC outperforms CBC due to the data volume-dependent overhead of the last one. This comparison on test time overhead works out in favor of the CSC solution. When the optimization of CBC is applied on the original circuit, the test time overhead is equivalent to the one achieved with the CSC solution.

Original circuit		**Triple-DES**		**Pipelined AES-128**		**Pipelined AES-256**		**RSA 1024**		**LEON3**	
		Area (µm²)	Test time* (clock cycles)	Area (µm²)	Test time* (clock cycles)	Area (µm²)	Test time* (clock cycles)	Area (µm²)	Test time* (clock cycles)	Area (µm²)	Test time* (clock cycles)
		187,494	687,101	367,926	1,944,877	669,193	4,559,845	468,415	39,405,239	1,902,095	11,612,051
Scan Encryption	Area (µm²)	*Overhead (%)*		*Overhead (%)*		*Overhead (%)*		*Overhead (%)*		*Overhead (%)*	
CSC (without TRNG implementation)											
AES-128 CTR	48,118.20	+25.66	+0.020	+13.08	+0.007	+7.19	+0.003	+10.27	+0.0004	+2.53	+0.001
PRESENT-128 CTR	6,833.84	+3.64	+0.014	+1.86	+0.005	+1.02	+0.002	+1.46	+0.0002	+0.36	+0.0008
TRIVIUM	5,408.52	+2.88	+0.18	+1.47	+0.06	+0.81	+0.03	+1.15	+0.003	+0.28	+0.01
CBC											
PRESENT-128	10,658.96	+5.74	+0.31	+2.92	+0.81	+1.61	+0.006	+2.30	+0.33	+0.57	+0.004
Optimization (observation points)		+5.87	+0.038**	+3.43	+0.013**			+2.51	+0.001**	+0.57	+0.002**
SKINNY-64-128	9,282.52	+4.95	+0.31	+2.52	+0.81	+1.39	+0.006	+1.98	+0.33	+0.49	+0.004
Optimization (observation points)		+5.08	+0.038**	+3.03	+0.013**			+2.09	+0.001**	+0.49	+0.002**

Table 7: Area and test time cost of scan encryption with stream cipher and block cipher, applied on several circuits.

****: test time considered for a fault coverage of 100% on the original circuit, except for the LEON3 processor where the fault coverage reaches 70%.***

*****: test time overhead for the block-based solutions compared to the test time of the optimized circuit when observation points are inserted.***

V.2.b On multiple scan chains

We present in Table 8 the experimental results of both CSC and CBC proposed solutions applied on devices with multiple scan chains. We consider a frequency of 10 MHz for the scan shift operations. In the following experiments, we use an estimation of the power consumed by the proposed solutions considering the ciphers implemented (AES-128 CTR and TRIVIUM for CSC, and PRESENT and SKINNY for CBC). The power consumption estimation is obtained after the synthesis of the scan encryption countermeasures with Design Compiler tool [66] at the frequency defined in Table 8.

The stream cipher decrypts/encrypts test data at the same frequency than shift operations (10 MHz) regardless of the number of scan chains (columns CSC, Table 8), implying a marginal cost in power consumption (about 82 µW for AES-128 CTR and 35 µW for TRIVIUM).

With block ciphers encryption (columns CBC, Table 8), the cost in power consumption increases with the number of scan chains, as explained in Chapter IV. For the encryption of 4 scan chains, the power consumption is already 2.5 times higher with block ciphers encryption compared to AES-128 CTR encryption (221.3 µW vs 82.46 µW), and 6 times higher compared to TRIVIUM (221.3 µW vs 34.16 µW). The difference is even greater for the case of 32 scan chains: the TRIVIUM performs the scan encryption at 10 MHz, consuming 36.37 µW, while PRESENT and SKINNY operates at 330 MHz (2,434.7 µW), and 370 MHz (2,232.9 µW) respectively, consuming 33 times higher than TRIVIUM. However, the generation of several keystreams from TRIVIUM to encrypt several scan chains has an area cost. To encrypt 32 parallel scan chains, the area cost of CSC with TRIVIUM represents 9,999.60 µm², equivalent to the cost (9,838.92 µm²) of CBC with SKINNY.

Scan encryption	**CSC (without TRNG implementation)**						**CBC**					
Ciphers	AES-128 CTR			TRIVIUM			PRESENT-128			SKINNY-64-128		
# scan chains	Area (µm²)	Power (µW)	Freq. (MHz)	Area (µm²)	Power (µW)	Freq. (MHz)	Area (µm²)	Power (µW)	Freq. (MHz)	Area (µm²)	Power (µW)	Freq. (MHz)
1	48,118.20	81.68	10	5,408.52	34.01	10	10,658.96	73.80	10	9,282.52	60.35	10
2	48,128.60	81.70	10	5,553.60	34.02	10	11,877.84	147.6	20	10,501.4	128.2	21.25
4	48,243.00	82,46	10	5,851.04	34.16	10	11,652.16	221.3	30	10,275.72	196.1	32.5
8			10	6,453.20	34.55	10	11,532.56	368.9	50	10,156.12	331.9	55
16			10	7,615.92	35.14	10	11,520.08	664.0	90	10,143.64	603.5	100
32			10	9,999.60	36.37	10	11,215.36	1,254.2	170	9,838.92	1,146.7	190
64							11,183.12	2,434.7	330	9,806.68	2,232.9	370

Table 8: Scan encryption with stream cipher and block cipher applied on multiple scan chains design.

V.3. Testability evaluation

The scan encryption permits to test the original circuit without reducing the test coverage since the original test patterns are still applied on the CUT. However, extra resources for scan encryption must also be tested. First of all, scan chains on extra circuitry would expose the cipher to scan attack and are thus avoided. We propose to functionally test extra ciphers using the test patterns dedicated to the CUT and processed by the ciphers during encryption or decryption.

The test of block ciphers, such as PRESENT, SKINNY and AES, is facilitated by the diffusion properties of the crypto-algorithms, as described in [37] and [38]. The stream ciphers based on shift registers, such as TRIVIUM, are also easily testable since all the states of the stream ciphers are shifted out the circuit as keystream.

Random data and possible errors are indeed easily propagated through typical operations involved in such encryption algorithms. Experiments performed on PRESENT, SKINNY, TRIVIUM and AES show that 100% fault coverage of stuck-at faults can be achieved with 1000 random patterns of N=64 bits each.

Therefore, both ciphers easily propagate the possible errors to the outputs when an encryption is performed. To validate the assumption, we have evaluated the test coverage achieved on every studied cipher (PRESENT, SKINNY, TRIVIUM and AES) by applying on each the test sequence of the

original CUTs. At scan-input, test patterns are processed by the input scan cipher, and the test responses are processed by the output scan cipher.

We have performed experiments with the test sequence of the Pipelined AES-256, Triple-DES, Pipelined AES-128, RSA 1024 and LEON3 processor cores. Table 9 presents the results for the CBC solution with both PRESENT and SKINNY block ciphers. For each circuits, we report the number of scan FFs (row *#SFF*), the number of patterns (row *#Patterns*), the test coverage of the original circuit, the number of encryption performed by the block cipher (row *#Encryption (64-bit block size*), and the test coverage of the scan ciphers. In all cases, the important number of encryption ensures that the fault coverage for stuck-at faults in the CBC architecture is 100%. In other words, the ciphers are tested for free. The same results are obtained on the CSC architecture.

Original circuit	***Triple-DES***	***Pipelined AES 128***	***Pipelined AES 256***	***RSA 1024***	***LEON 3***
#SFF	*8 808*	*7 873*	*12 736*	*16 459*	*107 518*
#Patterns	*77*	*246*	*357*	*2 393*	*107*
Original circuit Test coverage	*100%*	*100%*	*100%*	*100%*	*70%*
#Encryption (64-bit block size)	*138×77=10,626*	*124×246=30,504*	*199×357=71,043*	*258×2393=617,394*	*1680×107=179,760*
Scan ciphers Test Coverage	***100%***	***100%***	***100%***	***100%***	***100%***

Table 9: Scan ciphers tested for free.

V.4. Integration of the solutions in a SoC design

The integration on the scan encryption countermeasures consists in adding ciphers at the input and the output of the scan chain(s). In the case of CSC, as shown in Section IV.4.c.i, it also implies some modifications on the JTAG test wrapper, but not on the core itself. Therefore, the solutions can be applied without modification on the protected core.

The test wrappers of the cores composing a SoC are often connected in a test daisy-chain, as described in Chapter II. The serial input/output of the test interface provides access to the scan chains for all cores implemented in the SoC. When the scan encryption is implemented on one core, all the core included in the test daisy-chain propagates encrypted data, providing protection against malicious core. The stream cipher encryption operates bitwise on the data, implying no issue on the integration of the CSC in a test daisy-chain. Contrarily, the block cipher encrypts blocks of data, implying to pad the test data to a multiple of the block size. The extra data used for padding is thus propagated through other cores in SoC daisy-chain, resulting in possible issues during the test operations. Therefore, the designer has to be aware of the potential problems when the CBC is implemented. A way to avoid the

padding issue is to have a scan chain length equals to a multiple of the cipher block size. However, the insertion of extra FFs implies to modify the original circuit.

V.5. Conclusion on the advantages of the proposed countermeasures

In Chapter IV, we have proposed two new countermeasures consisting in encrypting the test communication: one based on block cipher, another based on stream cipher. Compared to the state-of-the-art countermeasures, these solutions have the advantages to provide a large protection against external threats (unauthorized access to the test interface) as well as internal threats (insertion of malicious device in a daisy-chain). In addition to the security provided by the proposed scan encryption countermeasures, the implementation of these solutions is plug-and-play, cost efficient and preserves the testability of the device under test. As presented in this chapter, the test of the extra scan encryption devices does not involve any test time overhead as they are freely tested during CUT testing.

In this chapter, we have also given a comparison between both proposed solutions. Experimental results have shown that the scan encryption with stream cipher has a lower cost in terms of area, test time and power consumption, compared to the scan encryption with block cipher, if a TRNG is already implemented in the original system. In the opposite case, the block cipher encryption is a good alternative allowing a possible optimization in order to reduce the test time and facilitate the integration in a SoC design.

The comparison between both encryption techniques has been presented in a conference [81] and has been submitted to the journal Transactions on Very Large Scale Integration Systems (TVLSI). In the next chapter, we give a summary of the comparison to conclude this thesis, and draw some perspectives.

Chapter VI

Conclusion

Summary

VI.1 Contributions

Cryptographic ICs are widely used for ensuring confidentiality in the system. It is the case for mobile devices, protecting the cryptographic operations using the TEE, isolating at the software level from the Rich OS Android. However, this protection is not sufficient for protecting the secure circuit against hardware attacks, such as the exploitation of test infrastructures.

The test of cryptographic circuits is mandatory to ensure the quality of the product. It is especially important since physical defects can compromise the security. IC designers needs thus to implement scan design, the most DfT technique, as well as test wrappers (JTAG, IEEE 1500 and IJTAG) bringing additional features than testing, such as debugging, uploading firmware, configuring FPGA and accessing instruments. Nevertheless, these test infrastructures give points to attack the secure system. In this thesis, we have presented contributions with new countermeasures consisting in encryption the scan network.

Overall, the scan encryption ensures protection against scan attacks [13]–[18], against an attacker exploiting the JTAG features (stealing IP design [26], updating a corrupted firmware [27], exploiting debugging facilities [28][29], and exploiting IJTAG network), and against internal threats such as the insertion of malicious core [30], and counterfeit components [31], due to test communication encryption. An attacker is not able to set the circuit in a desired state, nor to observe internal states of the circuit without knowing the secret key. Moreover, the presented CSC does not show the two times pad vulnerability, making it more secure than the state-of-the-art CSC [30], [60] and [61].

The establishment of the encrypted test communication between an authorized user and the secure circuit is possible with the shared knowledge of the secret key. Consequently, the user with the possession of the secret key is automatically authenticated by the secure circuit. Only authorized users can access the scan network for testing.

The proposed countermeasures also present advantages compared to existing scan attacks countermeasures. One of them is that it is still possible to perform in-field diagnosis and debug only for authorized users knowing the key, as opposed to the simple countermeasure consisting in disconnecting the test accesses, such as the countermeasure implemented in the reference board of TEEVA project [35]. Another advantage is the key management re-using the one already implemented in the circuit. Previous countermeasures based on a secure protocol using cryptographic primitives, such as in [51]–[58], need to have a dedicated key management.

The scan encryption does not impact the test coverage of the original circuit, contrary to the BIST solutions [36] which can reduce the test coverage in some cases. Moreover, the ciphers are tested functionally with the patterns of the original circuit, at the same time as the original circuit is tested.

VI.2 Summary of the comparison

In this thesis, we have also compared the two proposed solutions CBC and CSC. Table 10 resumes the comparison. In order to bring the best comparison possible, we give the pros and cons of the two

solutions for the implementation to achieve the best performance possible. Regarding the experimental results in Chapter V, the CBC is performed by SKINNY, while the CSC is performed by TRIVIUM. The CBC is evaluated for two cases. The solution being optimized or not with the insertion of test points in the original circuit in order to reduce the test time overhead. The CSC is also evaluated for two cases. The TRNG being already implemented or not.

An advantage of the CSC is that the integration in test daisy-chain implies no issue compared to the CBC. The optimization of the CBC permits to avoid the integration issue, but it is then not applicable on a non-modifiable core.

When a TRNG is already implemented in the device, another advantage of the CSC compared to the CBC is the area and test time costs for single scan chains. However, in the case where a TRNG is not available in the device for the scan encryption, the CBC can be preferred to the CSC due to the important area cost of a TRNG.

Concerning the implementation on multiple scan chains, the CSC has a lower cost in terms of area and power. However, the limit on the number of chains processed by one cipher is lower, from 2 to 32 scan chains, compared to the limit for the CBC which is from 2 to 64 scan chains.

Therefore, the choice of the CSC or CBC depends on the original circuit where the protection is implemented: if a TRNG is already set up, if the scan chain length is a multiple of the encrypted block

Scan encryption	**CBC (SKINNY)**		**CSC (TRIVUM)**	
	Test points insertion for test time optimization:		TRNG already implemented:	
	Not optimized	Optimized	Yes	No
Security				
Scan attacks	Protected		Protected (two times pad not possible)	
Exploiting JTAG features	Protected		Protected (two times pad not possible)	
Malicious core	Protected		Protected	
Counterfeit components	Protected		Protected	
Global features				
User authentication	Yes		Yes	
In-field diagnosis & debug	Yes		Yes	
Key management	Re-use the key management already implemented		Re-use the key management already implemented	
Integration				
Integration in test daisy chain	Possible issue with the padding of test data	No issue	No issue	
Appl. on non-modifiable core	Yes	No	Yes	
Test coverage				
Original circuit	No impact		No impact	
Secure test infrastructure	Functional test with test patterns of the original circuit		Functional test with test patterns of the original circuit	
Costs for single scan chain				
Area	9,282.52 µm²	+ Insertion of test points (< 500 µm²)	5,408.52 µm²	+ ~ 31,200 µm² for TRNG
Test time	Depends on the scan length (multiple or not of the block size)	Marginal cost (256 clocks cycles)	Clock cycles required for the initialization phase (1 232 clock cycles)	+ time to initialize the TRNG
Cost for multiple scan chains				
Area	~ 10,000 µm²		From 5,553.60 µm² to 9,999.60 µm²	+ ~ 31,200 µm² for TRNG
Power	From 128.2 µW to 2,232.9 µW		~ 35 µW	+ power consumption of the TRNG
Limit on the number of chains processed by one cipher	From 2 to 64 scan chains		From 2 to 32 scan chains	

Table 10: Comparison between scan encryption with stream cipher and block cipher.

size, if the original circuit is modifiable in order to insert observation points; and you also have to consider the number of scan chains that need to be encrypted.

VI.3 Perspectives

Considering the threats using the test interfaces defined in Chapter II, the only additional protection, which could be interesting to add to our proposed countermeasures, is to ensure integrity on the test messages in addition to the confidentiality. The integrity associate a tag to the message in order to distinguish valid messages from invalid ones. With this additional protection, the test messages coming from an attacker could be detected since they do not include the appropriate tags. In others words, a malicious component could not tamper test data, and an external attacker could not send “random” encrypted data to erase a memory region, such as a firmware. In literature, only the authors in [30] propose to add a HMAC on the test messages in order to ensure the integrity. However, HMAC implementation requires important hardware resources and add an important overhead on the test time, since the HMAC is computed for every received test messages. Instead of using a HMAC, a possible continuation would be to use *authenticated encryption*, ensuring both confidentiality and integrity of the test communication. The implementation and the costs of such solutions need to be determined.

BIBLIOGRAPHY

[1] ARM. (2009). ARM Security Technology. Building a Secure System using TrustZone Technology. *ARM White Paper*.

[2] Synopsys, TetraMax. [https://www.synopsys.com/implementation-and-signoff/rtl-synthesis-test/test-automation/tetramax-atpg.html]

[3] Savir, J. (1992). Skewed-Load Transition Test: Part I, Calculus. Proceedings - International Test Conference, 1992–Janua, 705–713.

[4] Lin, X., Press, R., Rajski, J., Reuter, P., Rinderknecht, T., Swanson, B., & Tamarapalli, N. (2003). High-frequency, at-speed scan testing. IEEE Design and Test of Computers, 20(5), 17–25.

[5] Borda, P., & Prajapati, P. (2014). Loc, Los and Loes At-Speed Testing Methodologies for Automatic Test Pattern Generation Using Transition Delay Fault Model. IJRET: International Journal of Research in Engineering and Technology, 3(3), 273–277.

[6] Rajski, J., Tyszer, J., Kassab, M., & Mukherjee, N. (2004). Embedded Deterministic Test. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 23(5), 776–792.

[7] Cadence. Cadence Modus DFT Software Solution. [www.cadence.com]

[8] Committee, I. S. (1990). IEEE Standard Test Access Port and Boundary-Scan Architecture. IEEE Std (Vol. 2001).

[9] IEEE Standard Testability Method for Embedded Core-based Integrated Circuits. (2012). IEEE Std 1500-2005.

[10] The IEEE Standards Association. (2014). IEEE Standard for Access and Control of Instrumentation Embedded within a Semiconductor Device.

[11] ARM Ltd. (2009). CoreSight Components.

[12] Daemen, J., & Rijmen, V. (2002). The Design of Rijndael. New York. https://doi.org/10.1007/978-3-662-04722-4

[13] Yang, B., Wu, K., & Karri, R. (2006). Secure Scan: A design-for-test architecture for crypto chips. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 25(10), 2271 2276. https://doi.org/10.1109/TCAD.2005.862745

[14] Yang, B., Wu, K., & Karri, R. (2004). Scan based side channel attack on dedicated hardware implementations of Data Encryption Standard. International Test Conference, 339 344. https://doi.org/10.1109/TEST.2004.1386969

[15] Da Rolt, J., Di Natale, G., Flottes, M. L., & Rouzeyre, B. (2011). New security threats against chips containing scan chain structures. 2011 IEEE International Symposium on Hardware-Oriented Security and Trust, HOST 2011, 105 110. https://doi.org/10.1109/HST.2011.5955005

[16] Ali, S. S., Saeed, S. M., Sinanoglu, O., & Karri, R. (2015). New scan-based attack using only the test mode and an input corruption countermeasure. IFIP Advances in Information and Communication Technology, 461(1), 48 68. https://doi.org/10.1007/978-3-319-23799-2_3

[17] Da Rolt, J., Das, A., Di Natale, G., Flottes, M. L., Rouzeyre, B., & Verbauwhede, I. (2012). A scan-based attack on elliptic curve cryptosystems in presence of industrial design-for-testability structures. Proceedings - IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, 43 48. https://doi.org/10.1109/DFT.2012.6378197

[18] Liu, Y., Wu, K., & Karri, R. (2011). Scan-based attacks on linear feedback shift register based stream ciphers. ACM Transactions on Design Automation of Electronic Systems, 16(2), 1 15. https://doi.org/10.1145/1929943.1929952

[19] NIST, N. I. of S. and T. (1999). Data Encryption Standard (DES). Federal Information Processing Standards Publication (FIPS PUB 46-3), 25(10), 1–22.

[20] Miller, V. (1986). Use of Elliptic Curves in Cryptography. Advances in Cryptology CRYPTO'85, LNCS 218, 417 426. https://doi.org/10.1007/3-540-39799-X

[21] C. Berbain, O. Billet, A. Canteaut, N. Courtois, B. Debraize, H. Gilbert, L. Goubin, A. Gouget, L. Granboulan, C. Lauradoux, M. Minier, T. Pornin, and H. Sibert. DECIM. http://www.ecrypt.eu.org/stream/decimp3.html

[22] C.J.A. Jansen, T. Helleseth, and A. Kholosha. Cascade jump controlled sequence generator and Pomaranch stream cipher. http://www.ecrypt.eu.org/stream/pomaranchp3.html

[23] I. Erguler and E. Anarim. A Modified Stream Generator for the GSM Encryption Algorithms A5/1 and A5/2. EUSIPCO 2005

[24] S. Thomas, D. Anthony, T. Berson, and G. Gong. The W7 Stream Cipher Algorithm. Internet Draft, April 2002

[25] A. Clark, E. Dawson, J. Fuller, J. Golic, H-J Lee, W. Millan, S-J. Moon, and L. Simpson. The LILI-II Keystream Generator. Proceedings of the 7th Australian Conference on Information Security and Privacy, volume 2384 of Lecture Notes In Computer Science, Springer-Verlag, 2002, pp. 25 39

[26] Altera. (2009). White Paper Protecting the FPGA Design From Common Threats. Memory, (June), 1–5.

[27] Maestra. Comprehensive Guide to Satellite Testing V5. 2002. http://www.maestra.ca

[28] G. Hotz. Blog: Finding JTAG on the iPhone. 2007.

[29] Free60. Xbox360 Hardware: JTAG/SMC Hack. 2009. http://free60.org/wiki/SMC_Hack

[30] Rosenfeld, K., & Karri, R. (2010). Attacks and defenses for JTAG. IEEE Design and Test of Computers, 27(1), 36 47. https://doi.org/10.1109/MDT.2010.9

[31] Top 5 Most Counterfeited Parts Represent a $169 Billion Potential Challenge for GlobalSemiconductor Market, https://technology.ihs.com/405654/top-5-most-counterfeited-parts-represent-a-169-billion-potential-challenge-for-global-semiconductor-market

[32] N. Keltner. Blog: Here Be Dragons: Vulnerabilities in TrustZone. 2014. http://atredispartners.blogspot.fr/2014/08/here-be-dragons-vulnerabilities-in.html

[33] D. Rosenberg. Blog: Unlocking the Motorola Bootloader. 2013. http://blog.azimuthsecurity.com/2013/04/unlocking-motorola-bootloader.html

[34] D. Shen. Blog: Exploiting Trustzone on Android. 2015. https://www.blackhat.com/docs/us-15/materials/us-15-Shen-Attacking-Your-Trusted-Core-Exploiting-Trustzone-On-Android-wp.pdf

[35] HiKey. (2015). HiKey (LeMaker version) Hardware User Manual, (December), 1–22.

[36] Wohl, P., Waicukauski, J. A., & Patel, S. (2004). Scalable selector architecture for X-tolerant deterministic BIST. Proceedings - Design Automation Conference, 934 939. https://doi.org/10.1109/DAC.2004.239784

[37] G. Di Natale, M. Doulcier, M. L. Flottes, B. Rouzeyre, "Self-Test Techniques for Crypto-Devices", IEEE Transaction on VLSI Systems, pp. 1-5, February 2010, Volume:18, Issue: 2, DOI: 10.1109/TVLSI.2008.2010045

[38] A. Schubert and W. Anheier, "On random pattern testability of cryptographic VLSI cores," J. Electron. Test.: Theory Appl., vol. 16, no. 3, pp. 185–192, Jun. 2000.

[39] Doulcier, M., Flottes, M. L., & Rouzeyre, B. (2008). AES-based BIST: Self-test, test pattern generation and signature analysis. Proceedings - 4th IEEE International Symposium on Electronic Design, Test and Applications, DELTA 2008, 314 321. https://doi.org/10.1109/DELTA.2008.86

[40] FUJIWARA, H., & FUJIWARA, K. (2015). Strongly Secure Scan Design Using Generalized Feed Forward Shift Registers. IEICE Transactions on Information and Systems, E98.D(10), 1852 1855. https://doi.org/10.1587/transinf.2015EDL8100

[41] Hély, D., Bancel, F., Flottes, M. L., & Rouzeyre, B. (2005). Test control for secure scan designs. Proceedings of the 10th IEEE European Test Symposium, ETS 2005, 2005, 190 195. https://doi.org/10.1109/ETS.2005.36

[42] Hely, D., Bancel, F., Flottes, M.-L., & Rouzeyre, B. (2006). A secure Scan Design Methodology. Proceedings of the Design Automation & Test in Europe Conference, 1 2. https://doi.org/10.1109/DATE.2006.244019

[43] Hély, D., Flottes, M.-L., Bancel, F., Rouzeyre, B., Bérard, N., & Renovell, M. (2004). Scan design and Secure Chip. Proceedings of the 4th IEEE International On-Line Testing Symposium (IOLTS'04).

[44] Lee, J., Tehranipoor, M., & Plusquellic, J. (2006). A low-cost solution for protecting IPs against scan-based side-channel attacks. Proceedings of the IEEE VLSI Test Symposium, 2006, 94 99. https://doi.org/10.1109/VTS.2006.7

[45] Da Rolt, J., Natale, G. Di, Flottes, M., & Rouzeyre, B. (2014). Thwarting Scan-Based Attacks on Secure-ICs With On-Chip Comparison. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 22(4), 947 951.

[46] Dworak, J., Crouch, A., Potter, J., Zygmontowicz, A., & Thornton, M. (2013). Don't forget to lock your SIB: Hiding instruments using P16871. Proceedings - International Test Conference, 1 10. https://doi.org/10.1109/TEST.2013.6651903

[47] Zygmontowicz, A., Dworak, J., Crouch, A., & Potter, J. (2014). Making it harder to unlock an LSIB: Honeytraps and misdirection in a P1687 network. In Design, Automation & Test in

Europe Conference & Exhibition (DATE), 2014 (pp. 1–6). New Jersey: IEEE Conference Publications. https://doi.org/10.7873/DATE2014.208

[48] Baranowski, R., Kochte, M. A., & Wunderlich, H. (2015). Fine-Grained Access Management in Reconfigurable Scan Networks. IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, 34(6), 937 946. https://doi.org/10.1109/TCAD.2015.2391266

[49] Novak, F., & Biasizzo, A. (2006). Security extension for IEEE Std 1149.1. Journal of Electronic Testing: Theory and Applications (JETTA), 22(3), 301 303. https://doi.org/10.1007/s10836-006-7720-x

[50] Chiu, G. M., & Li, J. C. M. (2012). A secure test wrapper design against internal and boundary scan attacks for embedded cores. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 20(1), 126 134. https://doi.org/10.1109/TVLSI.2010.2089071

[51] Clark, C. J. (2010). Anti-tamper JTAG TAP design enables DRM to JTAG registers and P1687 on-chip instruments. Proceedings of the 2010 IEEE International Symposium on Hardware-Oriented Security and Trust, HOST 2010, 19 24. https://doi.org/10.1109/HST.2010.5513119

[52] Da Rolt, J., Das, A., Di Natale, G., Flottes, M.-L., Rouzeyre, B., & Verbauwhede, I. (2014). Test Versus Security: Past and Present. IEEE Transactions on Emerging Topics in Computing, 2(1), 50 62. https://doi.org/10.1109/TETC.2014.2304492

[53] Park, K., Yoo, S. G., Kim, T., & Kim, J. (2010). JTAG security system based on credentials. Journal of Electronic Testing: Theory and Applications (JETTA), 26(5), 549 557. https://doi.org/10.1007/s10836-010-5170-y

[54] Buskey, R. F., & Frosik, B. B. (2006). Protected JTAG. Proceedings of the International Conference on Parallel Processing Workshops, 405 412. https://doi.org/10.1109/ICPPW.2006.65

[55] Das, A., Da Rolt, J., Ghosh, S., Seys, S., Dupuis, S., Di Natale, G., … Verbauwhede, I. (2013). Secure JTAG implementation using schnorr protocol. Journal of Electronic Testing: Theory and Applications (JETTA), 29(2), 193 209. https://doi.org/10.1007/s10836-013-5369-9

[56] Pierce, L., & Tragoudas, S. (2013). Enhanced secure architecture for joint action test group systems. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 21(7), 1342 1345. https://doi.org/10.1109/TVLSI.2012.2208209

[57] Das, A., & Kocabas, U. (2012). PUF-based secure test wrapper design for cryptographic SoC testing. Design, Automation and Test in Europe (DATE), 866 869. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6176618

[58] Backer, J., Hely, D., & Karri, R. (2015). Secure design-for-debug for Systems-on-Chip. Proceedings - International Test Conference, 2015 Novem, 1 8. https://doi.org/10.1109/TEST.2015.7342418

[59] De Canniere, C., & Preneel, B. (2005). TRIVIUM Specifications. ECRYPT Stream Cipher Project, Report, 30, 2005.

[60] Rosenfeld, K., & Karri, R. (2011). Security-aware SoC test access mechanisms. Proceedings of the IEEE VLSI Test Symposium, 100 104. https://doi.org/10.1109/VTS.2011.5783765

[61] Kan, S., Dworak, J., & Dunham, J. G. (2017). Echeloned IJTAG data protection. Proceedings of the 2016 IEEE Asian Hardware Oriented Security and Trust Symposium, AsianHOST 2016. https://doi.org/10.1109/AsianHOST.2016.7835558

[62] Baranowski, R., Kochte, M. A., & Wunderlich, H. J. (2014). Access Port Protection for Reconfigurable Scan Networks. Journal of Electronic Testing: Theory and Applications (JETTA), 30(6), 711–723. https://doi.org/10.1007/s10836-014-5484-2

[63] Ren, X., Torres, F. P., Blanton, R. D., & Tavares, V. G. (2018). IC Protection Against JTAG-based Attacks. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 0070(c), 1–14. https://doi.org/10.1109/TCAD.2018.2802866

[64] A. Bogdanov, L.R. Knudsen, G. Leander, C. Paar, A. Poschmann, M.J.B. Robshaw, Y. Seurin, and C. Vikkelsoe, P. Paillier and I. Verbauwhede. PRESENT: An Ultra-Lightweight Block Cipher. CHES 2007, LNCS 4727, pp. 450–466, Springer-Verlag Berlin Heidelberg 2007

[65] Beierle, C., Jean, J., Kölbl, S., Leander, G., Moradi, A., Peyrin, T., … Sim, S. M. (2016). The SKINNY Family of Block Ciphers and its Low-Latency Variant MANTIS. IACR-CRYPTO-2016

[66] Synopsys, Design Compiler. [https://www.synopsys.com/ implementation-and-signoff/rtl-synthesis-test/dc-ultra.html]

[67] Di Natale, G., Flottes, M.-L., Rouzeyre, B., & Pugliesi-Conti, P.-H. (2017). Manufacturing Testing and Security Countermeasures. In N. Sklavos, R. Chaves, G. Di Natale, & F. Regazzoni (Eds.), Hardware Security and Trust: Design and Deployment of Integrated Circuits in a Threatened Environment (pp. 127–148). Cham: Springer International Publishing.

[68] Synopsys. (2015). DesignWare True Random Number Generator Core.

PUBLICATIONS

JOURNAL

[69] M. Da Silva, M. L. Flottes, G. Di Natale, and B. Rouzeyre, "Preventing Scan Attacks on Secure Circuits Through Scan Chain Encryption," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD), vol. 0070, no. c, pp. 1–13, 2018. https://doi.org/10.1109/TCAD.2018.2818722

CONFERENCES WITH PROCEEDINGS

[70] M. Da Silva, M. Flottes, G. Di Natale, B. Rouzeyre, P. Prinetto, and M. Restifo, "Scan chain encryption for the test, diagnosis and debug of secure circuits," 22nd IEEE European Test Symposium (ETS), 2017, pp. 1–6. https://doi.org/10.1109/ETS.2017.7968248

[71] M. Da Silva, M.-L. Flottes, G. Di Natale, and B. Rouzeyre, "Experimentations on scan chain encryption with PRESENT," 2nd International Verification and Security Workshop (IVSW) 2017, pp. 45–50, 2017. https://doi.org/10.1109/IVSW.2017.8031543

[72] M. Da Silva, E. Valea, M.-L. Flottes, S. Dupuis, G. Di Natale, and B. Rouzeyre, "A new secure stream cipher for scan chain encryption," 3nd International Verification and Security Workshop (IVSW) 2018.

[73] E. Valea, M. Da Silva, G. Di Natale, M.-L. Flottes, S. Dupuis, and B. Rouzeyre, "SECCS : SECure Context Saving for IoT Devices," IEEE International Conference On Design & Technology of Integrated Systems In Nanoscale Era (DTIS), 2–3. https://doi.org/10.1109/DTIS.2018.8368561

[74] M. Da Silva, E. Valea, M.-L. Flottes, S. Dupuis, G. Di Natale, and B. Rouzeyre, "Encryption of test data: which cipher is better?," 14th conference on PhD Research In Microelectronics and Electronics (PRIME) 2018. https://doi.org/10.1109/PRIME.2018.8430366

OTHER PUBLICATIONS

[75] M. Da Silva, M.-L. Flottes, G. Di Natale, and B. Rouzeyre, "Sécurisation des structures de test : étude comparative," Colloque National du Groupement de Recherche (GDR) SoC2 (GDR SoC2) 2017.

[76] M. Da Silva, M.-L. Flottes, G. Di Natale, and B. Rouzeyre, "Scan Chain Encryption," Journée des Doctorants de l'école doctorale I2S (DOCTISS) 2017.

[77] M. Da Silva, M.-L. Flottes, G. Di Natale, and B. Rouzeyre, "Scan Chain Encryption for the Test, Diagnosis and Debug of Secure Circuits," South-Europe Test Seminar (SETS) 2017.

[78] M. Da Silva, M.-L. Flottes, G. Di Natale, and B. Rouzeyre, “Scan Chain Encryption in Test Standards,” Workshop on SecURity, REliAbiLity, test, prIvacy, Safety and Trust of Future Devices (SURREALIST) 2018.

[79] M. Da Silva, M.-L. Flottes, G. Di Natale, and B. Rouzeyre, “Scan Chain Encryption, a countermeasure against scan attacks,” Workshop on Practical Hardware Innovations in Security Implementation and Characterization (PHISIC) 2018.

[80] M. Da Silva, E. Valea, M.-L. Flottes, G. Di Natale, and B. Rouzeyre, “Stream cipher-based scan encryption in test standards,” Colloque National du Groupement de Recherche (GDR) SoC2 (GDR SoC2) 2018.

[81] M. Da Silva, M.-L. Flottes, G. Di Natale, and B. Rouzeyre, “Does stream cipher-based scan chains encryption prevent scan attacks?,” Workshop on Trustworthy Manufacturing ad Utilization of Secure Devices (TRUDEVICE) 2018.

[82] M. Da Silva, M.-L. Flottes, G. Di Natale, and B. Rouzeyre, “Sécurité des moyens de test des SoC”, Journée des GDR SoC2 et GDR Sécurité Informatique du CNRS : Sécurité des SoC complexes hétérogènes – de la TEE au matériel 2018.

[83] E. Valea, M. Da Silva, G. Di Natale, M.-L. Flottes, S. Dupuis, and B. Rouzeyre, “SECCS : SECure Context Saving for IoT Devices,” Colloque National du Groupement de Recherche (GDR) SoC2 (GDR SoC2) 2018.