Full Disk Encryption and Beyond

Louiza Khati

PSL UNIVERSITÉ PARIS

THÈSE DE DOCTORAT
DE L'UNIVERSITÉ PSL

Préparée à l'École Normale Supérieure

Full Disk Encryption and Beyond

Soutenue par
Louiza Khati
Le 15 juillet 2019

École doctorale n°ED 386
Sciences Mathématiques de Paris centre

Spécialité
Informatique

Composition du jury :

Pierre-Alain Fouque Université de Rennes I	*Rapporteur*
Kenny Paterson ETH Zurich	*Rapporteur*
Elena Andreeva KU Leuven	*Examinateur*
Maria Naya Plasencia Inria Paris	*Examinateur*
David Pointcheval Ecole Normale Supérieure	*Examinateur*
Yannick Seurin ANSSI	*Examinateur*
Damien Vergnaud Sorbonne Université	*Directeur de thèse*

1794 ENS | PSL

Full Disk Encryption and Beyond

Louiza KHATI

Supervisor: Damien VERGNAUD

Résumé

Cette thèse est dédiée à l'analyse de modes opératoires pour la protection des disques durs. Dans un premier temps, l'analyse des modes opératoires permettant de protéger la confidentialité des données est réalisée dans le modèle *Full Disk Encryption*. Ce modèle est très contraignant puisqu'il exclut tout mode qui ne ne conserve pas la longueur (la valeur en clair et chiffrée du secteur doivent avoir la même taille) et seuls des modes déterministes peuvent avoir cette propriété. Néanmoins, il est possible de tirer parti d'une valeur du système nommée le *diversifiant*, qui originellement a un autre but, pour apporter de l'aléa utile pour améliorer la sécurité des modes opératoires. Dans un second temps, nous introduisons deux méthodologies d'analyse dans le modèle *Key-Dependent Message*, où l'adversaire est autorisé à chiffrer des messages qui dépendent de la clé de chiffrement, qui nous ont permis d'analyser la sécurité des schémas Even-Mansour et Key-Alternating Feistel. Enfin, sachant qu'il est impossible de garantir l'authenticité des données dans le modèle FDE, la présence de codes d'authentification étant nécessaire, deux modèles où le stockage de métadonnées est possible sont envisagés : le modèle ADE pour *Authenticated Disk Encryption* et le modèle FADE pour *Fully Authenticated Disk Encryption*. Le premier permet de garantir l'authenticité au niveau du secteur mais est vulnérable aux *attaques par rejeu* et le second garantit l'authenticité du disque en entier et prévient ce type d'attaque. Le stockage n'est pas le seul point à prendre en compte : les vitesses de lecture et d'écriture sont un enjeu de taille pour les constructeurs puisque ces dernières conditionnent fortement les performances d'un disque. C'est la raison pour laquelle, nous avons étudié les codes d'authentification incrémentaux puisque ces derniers ont la propriété d'être mis à jour en un temps proportionnel à la modification réalisée.

Abstract

This thesis is dedicated to the analysis of modes of operation in the context of disk protection usage. Firstly, we give modes of operation secure in the *Full Disk Encryption* (FDE) model where additional data storage are not allowed. In this context, encryption has to be length preserving which implies length-preserving encryption. However, it is possible to use a value already present in the system, called a *diversifier*, to randomize the encryption and to have a better security. Then, we introduce two methods to analyse symmetric primitive in the very constraint *Key-Dependent Message* (KDM) model which is of interest for disk encryption because the encryption key can end up in the disk. It enables to analyse the KDM security of the Even-Mansour and the Key-Alternating Feistel constructions which are the basis of different block-ciphers. Moreover, knowing that data authenticity cannot be ensured in the FDE model because tag storage is not allowed, we relax this constraint which gives us two models: the *Authenticated Disk Encryption* model (ADE) and the *Fully Authenticated Disk Encryption* (FADE). A secure mode in the ADE model ensures data authenticity of a sector but can be vulnerable to *replay attacks*; and a secure mode in the FADE model ensures the authenticity of the entire disk even against replay attacks. Storage is not the only point to take into account, the read and write delays on a sector is a competitive argument for disk manufacturers since disk performances tightly depend on it and adding the computation of codes of authentication does not help. That is why, we tend to analyse incremental Message Authentication Codes: they have the property to be updatable in a time proportional to the corresponding modification.

Contents

Résumé iii

Abstract v

1 Introduction 1
- 1.1 Disk Protection . . . 3
- 1.2 Provable Security . . . 4
- 1.3 Notations and Definitions . . . 4
- 1.4 Our contributions . . . 7
  - 1.4.1 Full Disk Encryption Modes . . . 7
  - 1.4.2 Key-Dependent Message Security . . . 8
  - 1.4.3 Incremental MACs . . . 8

2 Secure storage - Confidentiality and Authentication 11
- 2.1 Data Protection and Data Storage . . . 13
  - 2.1.1 Data Protection . . . 13
  - 2.1.2 From Disk Storage to Data Encryption . . . 15
- 2.2 Full Disk Encryption . . . 18
  - 2.2.1 Challenges . . . 18
  - 2.2.2 FDE and Cryptography . . . 20
- 2.3 Data Authentication . . . 24
  - 2.3.1 Local Authenticity . . . 24
  - 2.3.2 Global Authentication and Incremental Cryptography . . . 27
  - 2.3.3 Merkle Tree . . . 29

3 Full Disk Encryption 35
- 3.1 Disk encryption methods and Security notions . . . 38
  - 3.1.1 Disk Encryption Methods . . . 38
  - 3.1.2 Security Notions for FDE . . . 41
- 3.2 FDE Security with Unique First Block . . . 48
  - 3.2.1 CBC-ESSIV Security . . . 48
  - 3.2.2 IGE-ESSIV Security . . . 52
- 3.3 FDE Security with a Diversifier . . . 54
  - 3.3.1 Solid State Drive . . . 55
  - 3.3.2 Case Studies . . . 58

4 Key-Dependent Message Security 61
- 4.1 KDM Security via Splitting and Forgetting Technique . . . 65
  - 4.1.1 Analysis via Forgetful Oracle Replacement . . . 68
  - 4.1.2 KDM Security of the Ideal Cipher . . . 72

4.2 Security of Even–Mansour Ciphers . . . 76
4.2.1 KDM Attacks on Even–Mansour . . . 76
4.2.2 KDM Security of Even–Mansour Ciphers . . . 78
4.3 KDM Security using the H-coefficient technique . . . 84
4.3.1 H-coefficient and KDM Security . . . 84
4.3.2 KDM Security with a Generic Lemma . . . 85
4.4 Security of Key Alternating Feistel Ciphers . . . 88
4.4.1 KDM Security of Four-Round Key-Alternating Feistel . . . 89
4.4.2 Sliding attack for r-rounds . . . 100
4.5 Even-Mansour KDM security with H-coefficients . . . 102
4.5.1 Security of 1-round Even-Mansour . . . 102

5 Incremental Authentication Schemes 107
5.1 Incremental MACs and Security notions . . . 111
5.1.1 Incremental Authentication Scheme Framework . . . 111
5.1.2 Security Notions for Incremental MACs . . . 115
5.1.3 Relations among Security Notions for Incremental MACs . . . 118
5.1.4 From Single-Document to Multi-Document Security . . . 122
5.2 Incremental MACs with IUF1 Security . . . 127
5.2.1 XMAC Constructions . . . 127
5.2.2 XS Construction . . . 128
5.2.3 MXS Construction . . . 135
5.3 Incremental MACs with IUF2 Security . . . 142
5.3.1 XMAC Constructions . . . 142
5.3.2 MXS Construction . . . 143

6 Conclusion and Open Questions 149
6.1 Conclusion . . . 149
6.2 Open Questions and futur work . . . 150

Bibliography 151

List of Illustrations 167
Figures . . . 167
Tables . . . 169

Personal Publications 171

Chapter 1

Introduction

Contents

1.1 Disk Protection **3**
1.2 Provable Security **4**
1.3 Notations and Definitions **4**
1.4 Our contributions **7**
1.4.1 Full Disk Encryption Modes 7
1.4.2 Key-Dependent Message Security 8
1.4.3 Incremental MACs 8

1.1 Disk Protection

Cryptography is a well-known domain aiming at protecting communication and data. A large part of humanity already used cryptography through communication protocols or when accessing securely stored data. Data protection requires secure cryptographic schemes that were, before the introduction of provable security, the result of a ping-pong game between cryptographers, who design and patch schemes when needed, and cryptanalysts who are looking for security flaws in those designs. A scheme that successfully resists years of cryptanalysis does not mean that it is secure, a vulnerability can be found in the future. This thesis aims at analysing symmetric encryption schemes for data at rest with specific constraints.

This is all the more important since the proliferation of mobile electronic devices has made it critical to protect the on-device data. Mobile phones or laptops have the unfortunate propensity to be lost or stolen. In case of such an event, the data stored in the device, which may contain extremely valuable and sensitive information, is extremely vulnerable. Users are indeed storing nowadays copies of personal documents (e.g., social security cards, driver's license, financial documents, personal pictures ...) on these devices that, if breached, can cause significant problems for them. An important security objective is therefore to ensure the confidentiality of on-device data for mobile devices. This mobile technology setting implies strong constraints on the methods to be used for secure disk storage. The encryption algorithm has to be computationally, and space efficient concerning the storage for reading/writing access of the data.

It is usually assumed that an encryption scheme suitable for the application of disk encryption must be length-preserving. This property reduces the scheme candidates by removing cryptographic schemes that require the storage of additional data such as initialization vectors or tags. This so-called *Full Disk Encryption (FDE)* method is implemented in a large panel of devices, from the compact and widespread ones like smartphones but also to the larger ones more fitted to industrial needs such as data center servers, reflecting its importance. FDE, properly used, is known guarantee the confidentiality of every bit in the disk at a low level (e.g., sector level). Obviously, reading and writing operations in a disk should suffer as little as possible from performance degradation that occurs when FDE technology is deployed. This goal requires FDE to have random access to every single sector on the drive and impacts different layers in a device: from the disk memory cells to the operating system.

FDE is now a mainstream technology that provides confidentiality but unfortunately does not provide cryptographic data integrity protection. However, if data confidentiality is necessary, it is perhaps even more critical to descry data alteration. A system using a memory should detect and alert the user when undesired modifications appear on the disk. Up to now, most disk encryption solutions rely on the so-called *poor-man's authentication* to ensure integrity, meaning if an attacker alters some encrypted data on the disk, the corresponding plaintext will be scrambled unpredictably, and the operating system or the application would notice that and gives an alert. However, in practice there exist applications that ignore errors in important data and simply change behavior (e.g., if the security configuration file does not parse, then the application simply ignores it). This poor-man's authentication approach is not cryptographically viable and stronger integrity should be provided. This protection comes with a cost: any authenticity oriented cryptosystem needs at least to store authentication tags and it seems that some latency will be added due to extra read/write accesses and tags' computation. Having said that, it is important to analyse if a cryptographic mechanism

guarantees data authenticity with minimal performance impact and minimal storage.

In this thesis, we present cryptographic constructions specifically designed for disk protection, or at least very suitable for this usage and their security is analysed with security proofs.

1.2 Provable Security

The security analysis of any scheme is not only reduced to ensure that it is not vulnerable to all the known attacks, this is not enough. A new attack can be found every day. This is why provable security was introduced in the seminal paper [GM84] by Goldwasser and Micali. Its principle is that to prove that a cryptographic scheme satisfies some security property, we must show that no algorithm running in polynomial time can be able to break this security property assuming some mathematical assumptions. The security argument is a reduction that uses an algorithm $\mathcal{A}$ against a cryptographic scheme to break the mathematical assumption which means that breaking the security property of the scheme implies attacking the mathematical assumption. To do so, a first step is to specify the security model which is the definition of the security property and the adversarial model. The security property specifies what it means to the cryptographic scheme to be secure, and the adversarial model specifies what actions the adversary is allowed to do and what information he is allowed to know. All cryptographic schemes proved in this thesis have been analysed in the framework of "reductionist security" (with a concrete security approach).

Adversary. Following Church-Turing thesis, an adversary is a probabilistic Turing machine (or algorithm, or program) which can be computationally bounded or unbounded. It is denoted $\mathcal{A}^{\mathcal{O}(.)}$ meaning it has access to an oracle $\mathcal{O}(.)$.

Games. For computationally bounded adversaries, we use the game-based framework of Bellare and Rogaway [BR06]. A Game is a program that is run with an adversary $\mathcal{A}$. The adversary interacts with procedures, usually called oracles, specified in the game Game. A game Game starts with a Initialize procedure, follows with a non-negative number of additional procedures called oracles, and ends with a Finalize procedure. If Finalize is omitted it is understood to be a trivial procedure that simply returns its output. Execution of adversary $\mathcal{A}$ with game Game consists of running $\mathcal{A}$ with oracle access to the game procedures, with the restrictions that $\mathcal{A}$'s first call must be to Initialize (if present), its last call must be to Finalize, and it can call these procedures at most once. The output of the execution is the output of Finalize. By $\Pr[\mathsf{Game}(\mathcal{A})]$ we denote the probability that the execution of game Game with adversary $\mathcal{A}$ results in this output being true. In games, integer variables, set variables and boolean variables are assumed initialized, respectively, to 0, the empty set $\emptyset$, and false.

1.3 Notations and Definitions

We now introduce basic notations and definitions used all along this thesis. Other notations and definitions, relevant only in specific sections, are introduced when needed.
First, we let $\mathbb{N} := \{0, 1, \dots\}$ denote the set of non-negative integers, and $\{0,1\}^*$ denote the set of finite binary strings.

Binary strings. For two binary strings X and Y, $X||Y$ denotes their concatenation and (X, Y) denotes a uniquely decodable encoding of the pair X and Y. For $X \in \{0,1\}^*$, we use $|X|$ to denote the bit length of X and $|X|_\ell$ denotes the number of ℓ-bit block in the binary string X (and in particular $|X|_1 = |X|$). For $\mathsf{n} \in \mathbb{N}$, the set of all binary strings whose length is a multiple of n is denoted $\{0,1\}^{\mathsf{n}*}$. For $s \in \{0,1\}^{\mathsf{n}*}$, we also wrote $s = s[1]||s[2]||\ldots s[\mathsf{n}]$ for $s[i] \in \{0,1\}^{\mathsf{n}}$.

Sets. We denote by $\#S$ the number of elements in the set S. The complement of a finite set S is denoted $\bar{S}$. By $x \twoheadleftarrow S$, we mean sampling x uniformly from set S. By $\Pr[A]$, we mean the probability that the event A happens.

List. For a finite list L such that $L = \{(x_i, y_i), i \in \{0, .., n\}$ where $\#L = n$, we denote by $\mathsf{Dom}(L)$ the domain of L $\mathsf{Dom}(L) = (x_1, .., x_n)$ and by $\mathsf{Rng}(L)$ the range of L $\mathsf{Rng}(L) = (y_1, .., y_n)$. We denote appending element X (resp., a list L') to a list L by $L : X$ (resp., $L : L'$) and $L \leftarrow []$ initializing a list to empty.

Object access. The description of cryptographic scheme CS can be given a tuple

$$\mathsf{CS} := (\mathsf{A}, \mathsf{B}, \mathsf{C})$$

where each element can be a parameter of the scheme or an associated algorithm. The notation $\mathsf{CS.A}$ refers to the parameter or algorithm A of the scheme CS. Sometimes, it can be simply denoted by A if the corresponding scheme CS is clear by context.

Definition 1.3.1. *(Functions)* The set of all functions $\mathcal{D} \to \mathcal{R}$ is denoted $\mathcal{F}^*(\mathcal{D}, \mathcal{R})$ where the domain $\mathcal{D}$ and the range $\mathcal{R}$ are non-empty sets.

Often $\mathcal{D} = \{0,1\}^\ell$ and $\mathcal{R} = \{0,1\}^L$ for $(\ell, L) \in (\mathbb{N} \setminus \{0\})^2$. In the following, a random function F^* refers to a function uniformly sampled at random in the set $\mathcal{F}^*(\mathcal{D}, \mathcal{R})$. We denote the composition of the same function F, i-th times, on a value $x \in \mathcal{D}$ such that $\mathsf{F}^i(x) = \mathsf{F} \circ \cdots \circ \mathsf{F}(x)$.

Definition 1.3.2. *(Permutations)* The set of all permutations is the subset of all bijections in $\mathcal{F}^*(\mathcal{D}, \mathcal{R})$ denoted $\mathcal{P}^*(\mathcal{R})$ such that $\mathcal{D} = \mathcal{R}$.

Often $\mathcal{R} = \{0,1\}^L$ for $L \in \mathbb{N} \setminus \{0\}$. In the following, a random permutation P^* refers to a function uniformly sampled at random in the set $\mathcal{P}^*(\mathcal{R})$.

Definition 1.3.3. *(Function Family)* A *function family* $\mathcal{F}$ is a map $\mathcal{F} : \mathcal{K} \times \mathcal{D} \to \mathcal{R}$ where the *set of keys* $\mathcal{K}$, the *domain* $\mathcal{D}$ and the *range* $\mathcal{R}$ are finite non empty sets. The function $\mathcal{F}$ takes two inputs: a key $K \in \mathcal{K}$ and an argument $x \in \mathcal{D}$ and returns a value $y \in \mathcal{R}$ such that $\mathcal{F}(K, x) = y$. Each key $K \in \mathcal{K}$ defines a map $\mathcal{F}_K : \mathcal{D} \to \mathcal{R}$ such that $\mathcal{F}_K(x) = \mathcal{F}(K, x) = y$. Then $\mathcal{F}_K$ denotes only one instance of the function family $\mathcal{F}$.

A pseudorandom function family (PRF) is a function family which behaves like a random function for a computationally bounded adversary (e.g., input and output behavior). More formally, we have the following definition.

Definition 1.3.4. *(Pseudorandom function family)* A *pseudorandom function family* $\mathsf{F} = (\mathsf{KS}, \mathsf{Dom}, \mathsf{Rng}, \mathsf{eval})$ is a function family $\mathcal{F}$ whose the key space, domain and range are denoted respectively $\mathsf{F.KS}$, $\mathsf{F.Dom}$ and $\mathsf{F.Rng}$ together with an algorithm eval such that for all $K \in \mathsf{F.KS}$ and for all $x \in \mathsf{F.Dom}$, $\mathsf{F.eval}(K, x)$ outputs $\mathcal{F}(K, x)$.

We define the prf-advantage of an adversary $\mathcal{A}$ against the pseudorandom function family F as probability of success in the PRF game $\mathbf{G}^{\mathsf{prf}}_{\mathsf{F},\mathcal{A}}$ defined in the left of Figure 1.1, as $\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{F}}(\mathcal{A}) = 2 \cdot \Pr\left[\mathbf{G}^{\mathsf{prf}}_{\mathsf{F},\mathcal{A}}\right] - 1$.

Usually, we have $\mathcal{K} = \{0,1\}^{\mathsf{k}}$ where k is the *key length*. In the following, for simplicity, we will denote $\mathsf{F}(K, x)$ instead of $\mathsf{F}.\mathsf{eval}(K, x)$.

Game $\mathbf{G}^{\mathsf{prf}}_{\mathsf{F},\mathcal{A}}$	Game $\mathbf{G}^{\mathsf{prp}}_{\mathsf{P},\mathcal{A}}$
procedure INITIALIZE $K \twoheadleftarrow \mathsf{F}.\mathsf{KS}$ $\mathsf{F}_* \twoheadleftarrow \mathcal{F}^*(\mathsf{F}.\mathsf{Dom}, \mathsf{F}.\mathsf{Rng})$ $b \twoheadleftarrow \{0,1\}$	procedure INITIALIZE $K \twoheadleftarrow \mathsf{P}.\mathsf{KS}$ $\mathsf{P}_* \twoheadleftarrow \mathcal{P}^*(\mathsf{P}.\mathsf{Rng})$ $b \twoheadleftarrow \{0,1\}$
procedure $\mathrm{F}(x)$ **if** $(b = 0)$ **then return** $\mathsf{F}_*(x)$ **return** $\mathsf{F}.\mathsf{eval}(K, x)$	procedure $\mathrm{P}(x)$ **if** $(b = 0)$ **then return** $\mathsf{P}_*(x)$ **return** $\mathsf{P}.\mathsf{eval}(K, x)$
procedure FINALIZE(b') **return** $(b' = b)$	procedure $\mathrm{P}^-(y)$ **if** $(b = 0)$ **then return** $\mathsf{P}^-_*(y)$ **return** $\mathsf{P}.\mathsf{inverse}(K, y)$ procedure FINALIZE(b') **return** $(b' = b)$

Figure 1.1: (Right) Game for defining the prf-advantage of pseudorandom family functions F. (Left) Game for defining the prp-advantage of pseudorandom permutation family P.

Definition 1.3.5. *(Permutation Family)* A *permutation family* $\mathcal{P}$ is a map $\mathsf{F} : \mathcal{K} \times \mathcal{D} \to \mathcal{D}$ where the *set of keys* $\mathcal{K}$ and the *domain* $\mathcal{D}$ are finite non empty sets. The function $\mathcal{P}$ takes two inputs: a key $K \in \mathcal{K}$ and an argument $x \in \mathcal{D}$ and returns a value $y \in \mathcal{R}$ such that $\mathcal{P}(K, x) = y$. Each key $K \in \mathcal{K}$ defines a permutation $\mathcal{P}_K : \mathcal{D} \to \mathcal{R}$ such that $\mathcal{P}_K(x) = \mathcal{P}(K, x) = y$. Then $\mathcal{P}_K$ denotes only one instance of the function family $\mathcal{P}$.

A pseudorandom permutation family (PRP) is a function family which behaves like a random permutation for a computational bounded adversary (e.g., input and output behavior). More formally, we have the following definition.

Definition 1.3.6. *(Pseudorandom function family)* A *pseudorandom permutation family* $\mathsf{P} = (\mathsf{KS}, \mathsf{Dom}, \mathsf{eval}, \mathsf{inverse})$ is a permutation family $\mathcal{P}$ whose key space and domain are denoted respectively $\mathsf{P}.\mathsf{KS}$ and $\mathsf{P}.\mathsf{Dom}$ and $\mathsf{F}.\mathsf{Rng}$ together with two algorithms eval and $\mathsf{inverse}$ such that for all $K \in \mathsf{P}.\mathsf{KS}$ and for all $x \in \mathsf{P}.\mathsf{Dom}$, $\mathsf{P}.\mathsf{eval}(K, x)$ outputs $\mathcal{P}(K, x)$ and $\mathsf{P}.\mathsf{eval}(K, \mathcal{P}(K, x))$ outputs x.
We define the prp-advantage of an adversary $\mathcal{A}$ against a pseudorandom permutation family

P is defined in terms of the probability of success in the PRP game $\mathbf{G}^{\mathsf{prp}}_{\mathsf{P},\mathcal{A}}$ as defined in the right of Figure 1.1, as $\mathbf{Adv}^{\mathsf{prp}}_{\mathsf{P}}(\mathcal{A}) = 2 \cdot \Pr\left[\mathbf{G}^{\mathsf{prp}}_{\mathsf{P},\mathcal{A}}\right] - 1$.

In the following, for simplicity, we will denote $\mathsf{P}(K, x)$ instead of $\mathsf{P}.\mathsf{eval}(K, x)$ and $\mathsf{P}^{-1}(K, x)$ instead of $\mathsf{P}.\mathsf{inverse}(K, x)$.

Definition 1.3.7. *(Block-Ciphers BC)* Given two non-empty subsets $\mathcal{K}$ and $\mathcal{M}$ of $\{0,1\}^*$, called the key space and the message space respectively, we let $\mathrm{Block}(\mathcal{K}, \mathcal{M})$ denote the set of all functions $\mathsf{E} : \mathcal{K} \times \mathcal{M} \longrightarrow \mathcal{M}$ such that for each $K \in \mathcal{K}$ the map $\mathsf{E}(K, \cdot)$ is (1) a permutation on $\mathcal{M}$ and (2) length preserving in the sense that for all $M \in \mathcal{M}$ we have that $|\mathsf{E}(K, M)| = |M|$. Such an E uniquely defines its inverse $\mathsf{D} : \mathcal{K} \times \mathcal{M} \longrightarrow \mathcal{M}$. A block-cipher for key space $\mathcal{K}$ and message space $\mathcal{M}$ is a triple of efficient algorithms $\mathrm{BC} := (\mathsf{K}, \mathsf{E}, \mathsf{D})$ such that $\mathsf{E} \in \mathrm{Block}(\mathcal{K}, \mathcal{M})$ and its inverse is D.

A block-cipher BC is therefore a pseudorandom permutation family with key space $\mathcal{K}$, domain $\mathcal{M}$ and algorithm $\mathsf{eval} = \mathsf{E}$ and $\mathsf{inverse} = \mathsf{D}$. In more detail, K is the randomized key-generation algorithm which returns a key $K \in \mathcal{K}$. Typically $\mathcal{K} = \{0,1\}^{\mathsf{k}}$ for some $\mathsf{k} \in \mathbb{N}$ called the key length, and K endows it with the uniform distribution. Algorithm E is the deterministic enciphering algorithm with signature $\mathsf{E} : \mathcal{K} \times \mathcal{M} \longrightarrow \mathcal{M}$. Typically $\mathcal{M} = \{0,1\}^{\mathsf{n}}$ for some $n \in \mathbb{N}$ called the block length. (3) D is the deterministic deciphering algorithm with signature $\mathsf{D} : \mathcal{K} \times \mathcal{M} \longrightarrow \mathcal{M}$. Thus a block-cipher is correct in the sense that for all $K \in \mathcal{K}$ and all $M \in \mathcal{M}$ we have that $\mathsf{D}(K, \mathsf{E}(K, M)) = M$. It is also length preserving. (Note that length preservation follows from correctness if $\mathcal{M} = \{0,1\}^n$). A permutation on $\mathcal{M}$ is simply a block-cipher with key space $\mathcal{K} = \{\varepsilon\}$. We denote a permutation with P and its inverse with P^-. A permutation can be trivially obtained from block-cipher (by fixing the key). For a block-cipher $\mathrm{BC} := (\mathsf{E}, \mathsf{D})$, notation $\mathcal{A}^{\mathrm{BC}}$ denotes oracle access to both E and D for $\mathcal{A}$. We abbreviate $\mathrm{Block}(\{0,1\}^{\mathsf{k}}, \{0,1\}^{\mathsf{n}})$ by $\mathrm{Block}(\mathsf{k}, \mathsf{n})$ and $\mathrm{Block}(\{\varepsilon\}, \{0,1\}^n)$ by $\mathrm{Perm}(n)$.

Definition 1.3.8. *(Ideal-cipher Model ICM)* The ideal cipher for key space $\mathcal{K}$ and message space $\mathcal{M}$ is the uniform distribution over $\mathrm{Block}(\mathcal{K}, \mathcal{M})$. The Ideal-Cipher Model (ICM) with key space $\mathcal{K}$ and message space $\mathcal{M}$ is a model of computation where all parties, honest or otherwise, have an oracle access to a uniformly chosen random element in $\mathrm{Block}(\mathcal{K}, \mathcal{M})$ together with its inverse. The ideal-cipher model when restricted to $\mathcal{K} = \{\varepsilon\}$ gives rise to the random-permutation model (RPM).

1.4 Our contributions

1.4.1 Full Disk Encryption Modes

In chapter 3, we revisit the problem of Full Disk Encryption (FDE), which refers to the encryption of each sector of a disk volume. In the context of FDE, it is assumed that there is no space to store additional data, such as an IV (Initialization Vector) or a MAC (Message Authentication Code) value. We formally define the security notions in this model against Chosen Plaintext Attacks(CPA) and Chosen Ciphertext Attacks (CCA). Then, we classify various FDE modes of operation according to their security in this setting, in the presence of various restrictions on the queries of the adversary. We will find that our approach leads to new insights for both theory and practice. Moreover, we introduce the notion of a *diversifier*, which does not require additional storage but allows the plaintext of a particular sector to be

encrypted to different ciphertexts. We show how a 2-bit diversifier can be implemented in the EagleTree simulator for Solid State Drives (SSDs), while decreasing the total number of Input/Output Operations Per Second (IOPS) by only 4 %. These results were the subject of a publication at CT-RSA 2017 entitled "Full Disk Encryption: Bridging Theory and Practice" with Nicky Mouha and Damien Vergnaud.

1.4.2 Key-Dependent Message Security

Chapter 4 is dedicated to Key-Dependent Message security. The Key-Dependent Message (KDM) security model enables to analyse the security of a scheme even when the messages depend on the secret key. This approach is relevant for full disk encryption as the whole disk is encrypted, the encryption key or a function of the encryption key can end up in the disk. Two contributions analyse the KDM security of two constructions widely used as the basis for block-cipher design with different security proofs techniques: the iterated Even-Mansour (EM) ciphers also known as the Key-Alternating Ciphers is analysed with a particular framework and the Key Alternating Cipher (KAF) is analysed with the H-Coefficient technique.

The analysis of the KDM security of the iterated Even-Mansour ciphers is a joint work with Pooya Farshim and Damien Vergnaud; it was published with the title "Security of Even–Mansour Ciphers under Key-Dependent Messages" at IACR Transactions on Symmetric Cryptology in 2018. The iterated Even-Mansour ciphers form the basis of many block-cipher designs. Several results have established their security in the CPA/CCA models, under related-key attacks, and in the indifferentiability framework. We also formalize the folklore result that the ideal cipher is KDM secure. We then show that EM ciphers meet varying levels of KDM security depending on the number of rounds and permutations used. One-round EM achieves some form of KDM security, but this excludes security against offsets of keys. With two rounds we obtain KDM security against offsets, and using different round permutations we achieve KDM security against all permutation-independent claw-free functions. We also present some KDM-attacks on some EM ciphers. As a contribution of independent interest, we present a modular framework that can facilitate the security treatment of symmetric constructions in models that allow for correlated inputs.

The analysis of the KDM security of the Key-Alternating Feistel (KAF) constructions is a joint work with Pooya Farshim, Yannick Seurin and Damien Vergnaud. We use a different technique to analyse the KDM security of KAF schemes and this time, the analysis is based on the H-coefficient technique. We give a general approach to analyse KDM security of block-ciphers and we apply it to the 4-round KAF. This security result gives properties on the KDM set of functions that the adversary can ask to encrypt to have a secure scheme. Moreover, we describe a KDM-attack against 4-round KAF if those properties are not respected (and additional attacks on other KAF schemes).

1.4.3 Incremental MACs

An interesting feature for disk protection is integrity but using regular authentication schemes has a severe impact on performances that is why in chapter 5, we analysed some constructions to minimize as much as possible the slowdown of performances: the incremental MACs. Introduced in cryptography by Bellare, Goldreich and Goldwasser in 1994, *incrementality* is an attractive feature that enables to update a cryptographic output efficiently like a ciphertext, a signature or an authentication tag after modifying the corresponding input.

This property is very valuable in large scale systems where gigabytes of data are continuously processed (e.g., in cloud storage). Adding cryptographic operations on such systems can decrease dramatically their performances and incrementality is an interesting solution to have security at a reduced cost.

On the one hand, we analyse the security of the original Xor-scheme construction proposed by Bellare, Goldreich and Goldwasser in [BGG95] and based on a chained structure as defined in [BGG94]. We provide an attack that breaks the Xor-scheme *basic security* claimed by the authors[1]. It succeeds with probability 1 using only one MAC query. It takes advantage of the chaining structure of this scheme and some XOR function properties. This attack is very simple and it is surprising that it remained unnoticed until now (especially since the paper [BGG95] appeared in a major computer science conference and was extensively cited since 1994). We analyse our attack and the original Xor-Scheme to find where its security breaks down. We show that the main flaw is that the Xor-Scheme does not explicitly take into account the document length and we noticed that adding the number of data blocks to the construction prevents this kind of attacks. We analyse different ways to patch the scheme by introducing the document block-length in the construction and found that the scheme can still be weak for some options. We propose a modified version of the Xor-Scheme and prove its basic security. Our security proof for the patched Xor-Scheme uses tools from the unchained XOR-scheme security proof [BGR95a]. This result was published in a paper co-authored with Damien Vergnaud and entitled "Analysis and Improvement of an Authentication Scheme in Incremental Cryptography" which was published at SAC 2018.

On the other hand, we revisit incremental MACs security notions by giving clear security games for basic security where an adversary can call for update operations only on untampered documents and tamper-proof security game where the adversary can call for update operations on modified documents. We compare the relative strengths of our new security notions and we provide generic constructions. Moreover, we analyse existing incremental constructions and we provide a new scheme using our security proofs which is the first one achieving some kind of tamper-proof security (as formalized by our new model). This part of the chapter is a joint work with Vivek Arte, Mihir Bellare and Damien Vergnaud.

[1] In [BGG95, Theorem 3.1], Bellare, Goldreich and Goldwasser stated a security result for their scheme but no proofs are provided in their paper.

Chapter 2

Secure storage - Confidentiality and Authentication

Contents

2.1 Data Protection and Data Storage 13
2.1.1 Data Protection 13
2.1.2 From Disk Storage to Data Encryption 15
2.2 Full Disk Encryption 18
2.2.1 Challenges 18
2.2.2 FDE and Cryptography 20
2.3 Data Authentication 24
2.3.1 Local Authenticity 24
2.3.2 Global Authentication and Incremental Cryptography 27
2.3.3 Merkle Tree 29

This chapter aims at giving in a first place a state-of-the-art of cryptographic constructions used in FDE model and the reason for their design induced by storage constraints and performance goals. Then we formalize new models where, in addition to confidentiality, two levels of data authentication at low-level are possible. In section 2.1, data at rest threats are analysed to present adversary models and explain which adversary model FDE covers and which one it does not. Disk storage technologies are described in order to extract disk storage constraints. In section 2.2, the challenge of FDE is to take into account these constraints to maintain the device performances. In this context, performance impacts different aspects of a device and developers have to consider specific implementation choices for different components and layers. This leads to numerous types of FDE products. Section 2.3 explores the possibility to add data authentication. Two strategies of data authentication are detailed: local and global authentications. The first one protects authentication at a block level but does not prevent replay attacks where an adversary tries to restore a former version of the disk content. The second one protects the entire disk even against replay attacks by using specific authentication schemes that have the property to be incremental. An incremental authentication scheme enables to generate a tag on an input once and then for any modification; the tag is updated instead of being re-computed from scratch. The implications of these authentication schemes in terms of storage and update time are analysed and discussed.

2.1 Data Protection and Data Storage

2.1.1 Data Protection

Nowadays, the need for user data protection has become a critical matter, even more, emphasized by democratized cheap outsourced storage and the constant rise of mobile devices. For various reasons, a user (the device owner) can let his mobile device unattended, e.g. when attending a meeting or doing sport. A device may be sent to repair; sold to someone else, lost or stolen. It is also common that a device has to be given to legal authorities during checks in airports or when entering a police station. These examples are ordinary situations where a third party can have access to the user data stored in the disk even if the device is switched off. Unfortunately, the user is not always aware of potential threats when this third party is an adversary.

Devices. In the following, **a device** refers to a system that runs autonomously with *a persistent memory to protect*. FDE concerns data at rest stored in the disk only. Data is encrypted before being stored in the disk and decrypted after being loaded from the disk which means that the disk should always store data encrypted[1]. Data is said to be encrypted and decrypted "on the fly".
From a data at rest perspective, devices can be classified by the data size to protect, their power on and off cycle frequency, the number of users and so on. A server stores terabytes of data, it can be used by multiple users as it is a shared resource. It is commonly used in professional environments where critical material is stored in a shared appliance that is owned by the company or just rented for commodity (cloud). A desktop computer or a laptop is used by less users than a server and has a smaller disk but is powered off more often than a server which has to be running almost all the time. A smartphone or a digital tablet

[1]The strategy used for the initial encryption step depends on the FDE tool.

is usually used by a single user. As they become more and more powerful, they can almost be used as laptops and store sensitive data like contacts, business data, personal data or passwords. These devices are designed to be small and mobile to be taken everywhere: in the street, public transportation, at work and so on which render them easy to lose or to steal. We can mention smartwatches, biomedical devices and IoT devices which can store data at rest. Another category is USB thumb drives and disks which are dedicated to storing data at rest and are powered off very often when unplugged from a computer. The power on/off cycles frequency of a device is an important factor: FDE protects data against unauthorized access only when a machine is powered off. In other system states (running and sleep modes) FDE is not sufficient and additional protections are required.

Threat Models. All these devices store some amount of data that need to be protected in confidentiality, and different solutions exist depending on the model (architecture and adversary). In the following, we gather all these scenarios in three main attack categories according to how the adversary accesses the disk.

1. **Single passive access to the disk.** When a device is stolen or lost, the user wants to be assured of his data confidentiality: anyone except the user should learn no information about the disk data. When the disk is encrypted, the adversary has access to encrypted data, and he aims at discovering some information about the corresponding plaintexts. As examples, he can cryptanalyse the cryptographic algorithms, or he can compare some parts of the disk to others.

2. **Multiple passive access to the disk.** It encloses -1- but here the adversary is more powerful as he can make copies of the disk at different times. The adversary can analyse these copies in order to track modifications on the disk. Later, we will see that even with a strong FDE encryption mode, an adversary is able to recognize if the same value was stored in the same place at different moments and he knows exactly which parts have been changed between two copies.

3. **Active access to the disk.** This scenario encloses -1- and the adversary can tamper with the disk data. These attacks are also known as "active attacks" [Gjø05a]. As an active attack, we can think about silent data corruption [BAA+08] even if it is not an intentional attack as it can be due to random hardware failures. This issue is usually solved by checksums and CRC32 [SSTC02] is the most used one.

Another case is an adversary that succeeds to perform malicious modifications in the disk without the user realizing it. He can also tamper with the disk, and they can give it back to the user. After that, the user works with his disk as usual without knowing that his data are corrupted. An example is a copy-and-paste attack where a disk is shared between a user 1 and a user 2: the malicious user 1 copies data from the user 2 part of the disk and pastes it in its own part. As in many FDE tools, a unique key is used for the entire disk (more details in section 2.2.1); user 1 could be able to decrypt user 2's data.
In addition to data confidentiality, the user wants to be able to check the **authenticity** of the data he is using. A specific tampering attack is what we can call a "Downgrade attack" or a "replay attack": the adversary snapshots the entire disk at time t and waits for a specific moment to restore it. The desired property to thwart this is a so-called **temporal data authentication**: at any moment, he wants to be sure that the data he is manipulating now is the data stored during the previous legitimate manipulation.

For each category, an adversary can perform **online attacks** which means that it has physical access to the device to perform his attacks. As a consequence, he can attack the disk content but also the other components including hardware components. Otherwise, if it performs **offline attacks**, we suppose that he has no access to the entire device but only to the raw disk memory content, i.e. storage memory cells bits.

Attacks out of scope. Many other attacks [MF15] can be performed when the disk is **in use**. An adversary can try to freeze and extract cryptographic secrets from the Random Access Memory (RAM) when the disk is in use to decrypt the disk afterwards. This attack is known as a cold boot attack [GM14, MTF12, CK10, Sim11]. The encryption key can also be recovered through side channels by observing data manipulations on the disk and exploiting physical leakage such as power consumption or electromagnetic radiations [UM16]. A malware can also infect a device in order to spy on the legitimate user password and other secrets; this is the Evil Maid attack[2] as exhibited in [Rut09, Kle09, Ter10, GM14].

Attack families -1- and -2- threaten data confidentiality whereas scenario -3- is also a threat to their authenticity. In the next section, disk architectures are detailed to highlight specific disk features that have to be taken into account to choose a disk protection mechanism.

2.1.2 From Disk Storage to Data Encryption

In this section, a simplified description of a system is provided in order to understand what is full disk encryption and the differences between FDE products.

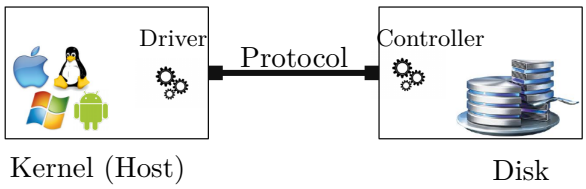

Figure 2.1: Simplified view of components involved in disk encryption.

First, let us define the taxonomy of the subjects that are of interest when studying FDE. At the highest level, we consider systems running operating systems (OS) and hosting applications. The core and privileged component of the OS is its kernel. In a nutshell, when a system is powered on, the first piece of code executed is called the BIOS, which runs in chain specific components in order to execute the kernel. After the boot, the kernel handles the memory and peripherals. It manages data stored in the disk through different virtualization layers as shown in Figure 2.2. Simply put, from an end-user perspective, data is abstracted in the form of files. All the files are aggregated in a filesystem, and filesystems can be stored in volumes or partitions (a volume can be split among different partitions). The kernel handles physical disks through partitions with specific drivers in charge of low-level reading and writing operations. As shown in Figure 2.1, a dedicated protocol conveys read and write commands at the physical layer level. Such commands are limited to a fixed size data unit called a sector. The low-level driver is consequently in charge of sector-based read and writes operations. The Parallel ATA (PATA) and Serial ATA (SATA[Del09]) standards are examples of such protocols, and are commonly used as classical disk interfaces.

[2]From "Evil Maid goes after TrueCrypt"

The physical disk contains a controller and non-volatile memory units to store data sectors. The controller is usually a dedicated micro-controller with embedded firmware that deals with the commands received from the kernel driver and manages the internal non-volatile memory depending on the disk technology (see Figure 2.5) as described in what follows.

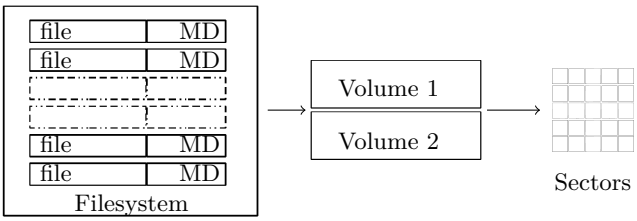

Figure 2.2: Simplified logical levels: from files to sectors. MD stands for Meta-data.

2.1.2.1 Hard Disk Drive

The older and well-known technology is the Hard Disk Drive (HDD) that is essentially composed of platters. It can be read and written by a mechanical arm handled by the disk controller. Each drive has one or more disk platters, each with two magnetic surfaces. The platters are coated on both surfaces with magnetic particles, allowing to store and retrieve data in the form of zeros and ones by polarizing the relevant area. Information on the disk surface is located on a set of concentric rings called tracks. The tracks on each platter are in turn divided into a number of sectors as shown in Figure 2.3.

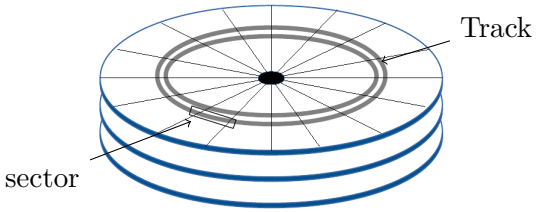

Figure 2.3: HDD tracks and view of a sector.

As already stated, the controller communicates with a driver through a specific interface using a protocol that enables write and read commands depending on sector numbers also called the Logical Block Addresses (LBA).

2.1.2.2 Solid State Drive

The physical composition of a Solid State Drive (SSD) is totally different. SSDs are flash memory devices that are gradually replacing the HDDs due to improved performances.

Similarly to magnetic drives, they are logically sector-addressable devices. The flash memory cells of an SSD (see Figure 2.4) are hierarchically organized as a set of flash chips called packages, which are further divided in dies, planes, blocks[3] and pages. Every page consists of one or more sectors and is the smallest unit that can be written. The smallest unit that can be erased is a block. Invalidated blocks must be erased before writing. The Flash Translation Layer (FTL) stores the mapping between LBAs and Physical Block Addresses (PBAs). This FTL ensures an even distribution of writes to every sector number (*wear leveling*) and, it invalidates blocks so that they can later be recycled (*garbage collection*). The FTL is necessary due to the physical constraints of flash storage: any physical address can only be written a limited number of times, and rewriting individual sectors is not possible: invalidated sectors can only be recovered in multiples of the erase block size.

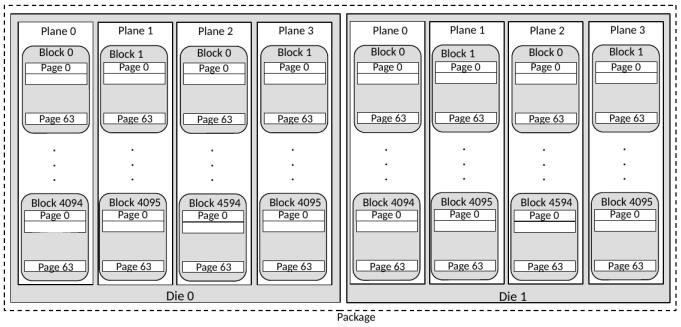

Figure 2.4: Hierarchically organized SSD memory

The FTL implements an abstraction layer to be compatible with systems using HDD: the driver reads and writes sectors using LBA. Disk interfaces such as PATA or SATA have different physical link properties, speed and communication protocols but they all enable access to sector granularity. The driver and other components from the OS manage data and optimize read and write access.

2.1.2.3 Physical and Logical sectors

The physical sector is the sector used internally in the disk, and the logical sector is the one presented to the host by the disk controller. The kernel asks to write at least one logical sector and, this operation is **atomic** from its point of view: the host cannot ask to write less than one sector. Historically, the HDD had 512-bytes length sectors that were presented to the host so, at the time, the logical and the physical distinction did not exist. To read and write disk data efficiently, the OS used a unit called a page[4] that is a virtual unit composed usually by eight logical sectors. A page is the smallest unit the OS can read and write and, its size is a multiple of logical sector size. OS storage page size is commonly chosen to match the virtual memory page size: this allows file system transfers between disk and RAM, caching and read/write operations to be optimized. Nowadays, disk manufacturers are able to produce disks with longer sectors of 4096 bytes on par with the OS page size to increase

[3]These blocks should not be confused with the blocks of the block cipher, nor with the "block" (actually "sector") in the term Logical Block Address (LBA).

[4]It is different from SSD pages.

disk performance at the lowest native level. Due to this heterogeneous situation regarding logical and physical sectors sizes, the Advanced Format standard has been created to specify various compatibility modes.

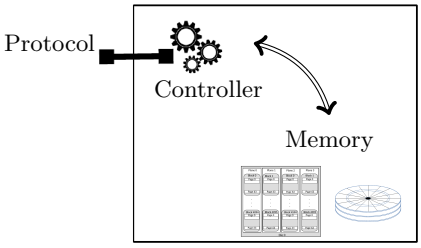

Figure 2.5: Disk where memory are on platters for HDD and are flash cells for SSD.

2.2 Full Disk Encryption

2.2.1 Challenges

Encryption can be implemented at different levels: from file to sector-based encryption (see Figure 2.2). File encryption aims at encrypting independently each file with different keys and usually without encrypting the corresponding meta-data. With filesystem encryption, the files and the meta-data (including the internal hierarchy) are encrypted as implemented in ecryptfs [Tec] for instance. Volume encryption, partition encryption and disk encryption use a block-oriented encryption known as Full Disk Encryption (FDE). Block refers to a sector which is the disk data unit. It is also known as Low-level full disk encryption or whole disk encryption[Ter10]. An advantage of FDE is that it ensures unconditional encryption of all the data stored in the disk, instead of selectively encrypting only the data to be kept secret. This is an advantage over file level encryption where extra data like the number of files or the file size cannot be encrypted. Hence, FDE has an interesting feature to ensure confidentiality for all the data while being transparent for the user. These advantages are even more important for organizations and companies where data protection must be enforced in contexts where human errors are unavoidable.

FDE cost. Disk memory access is time-consuming, and it is a critical issue for manufacturers and academic research. Software and hardware manufacturers made huge efforts to optimize memory accesses and succeeded to achieve tremendous advances in memory latency. Memory subsystems are tuned as much as possible with cache memories and fast SRAM memory chips to speed up computer capabilities. Adding an encryption layer could affect **performance** drastically. FDE frameworks developers try to integrate encryption that takes into account this optimization work. A consequence is that FDE is **length-preserving**: all the sectors are encrypted **independently** without additional data; otherwise sector (un)alignment management becomes a problem. Moreover, most of FDE implementations aim at being compatible with a maximum number of systems. A large panel of FDE frameworks tries to defeat different adversaries by implementing encryption at different layers and by managing the key differently.

Key management. To perform decryption, the user has to provide a secret; it is usually a password or a pin code. The secret will unlock the disk encryption key, and this key is then used to decrypt the whole disk. More complicated key managements are also implemented in some devices, but in this paper, we consider the classical FDE encryption schemes where the whole disk is encrypted with a unique key. This choice is due to shared disk usage, password modification and backup. The disk encryption key is encrypted with another key derived from a user password. This provides the ability to encrypt the key encryption key for different users with independent passwords. It is also useful in cases where a user wants to change his password without re-encrypting the whole disk (which would prove very inefficient), and allows to set up a backup password. Popular and well known key derivation functions are scrypt[PJ16] and PBKDF2[Jos11] because of their resistance to brute force attacks. These functions take as parameters the user secret, a random salt and a number of iterations. The random salt prevents rainbow attacks [KKJ+13] where the adversary has pre-computed hash tables for different passwords. A large number of iterations slows down slightly the legitimate user that computes it once, but it aims at making the computation time of the brute force attacker unpractical because it has to evaluate the key derivation function on a very large number times. The minimum salt size recommended for PBKDF2 by the standard PKCS#5 [Jos11] is 64 bits. In 2017, NIST recommended at least 1 000 iterations[Pau17] and specifies that more iterations are better as long as it is acceptable for the device to compute one evaluation of the derivation function (the appropriate number is 10 000 000 [Pau17]). Nowadays, attackers can use mainstream GPUs, FPGAs or dedicated hardware like ASICs to drastically speed up and parallelize the key derivation evaluation to recover passwords [VG18]. This is the reason why the memory-hard functions [ACP+17] like scrypt have been designed. The idea is that if the computational time is divided by some factor, it should cost an increase in space by the same factor. Another independent key (specific to a device) can be added as a parameter in the key derivation function. When this key is hardware bound and protected by a hardened circuit (like eFuses [RFC+07] in recent processors and SoCs), offline attacks become impossible, and the adversary is forced to perform online attacks with the devices as an oracle: brute force exhaustive search becomes unpractical even with dedicated hardware components. This is mostly the case for mobile phones that embed Secure Elements and/or hardened hardware: a fused AES-256 key called UID for iPhones [5] [App18] and a key used by the hardware-backed KeyMaster module for Android devices[LB17].

The location of the key management, the kernel driver or the disk controller, is a way to define two families of FDE products: the first one called Self-Encrypted Disks (SEDs) corresponds to stand-alone systems performing encryption/decryption inside the disk by the controller and the other corresponds to implementations performing encryption/decryption in software on the host side.

Standard Encrypted Disks . A standard disk (either HDD or SSD) usually aims only at storing data without encryption. Software FDE solutions like dm-crypt[dm-19a] (Linux), Bitlocker [Fer06] (Windows), FileVault[CGM13] (MacOS), Truecrypt/Veracrypt[6] (multi-platforms)[ver19] perform encryption through a computer and store the encrypted data in the standard disk. Even though such implementations can take advantage of hardware accelerated instructions such as AES-NI, we refer to them as software based solutions as

[5] iOS implements file based encryption and not full disk encryption

[6] Truecrypt project was ended in 2014. Veracrypt is a fork.

opposed to dedicated hardware solutions where most of the FDE logic is performed in dedicated controllers. In software oriented solutions, the standard disk does not manipulate raw user secrets neither the raw cryptographic key but the host does. An adversary can take advantage of this by targeting the RAM yielding in the attacks denoted "out of scope" previously in this section. In this case, the adversary has access to the standard disk and aims at breaking confidentiality.

Self-Encrypted Disks. In SEDs, encryption is performed by some dedicated hardware component in the disk itself. The SED is supposed to be unlocked only by the legitimate user. The user authentication is performed directly on a SED interface like a pinpad or through a trusted computer. Then the storage device performs the encryption/decryption on the device and the encryption key should never leave the device [MLF12].

Software/Hardware Implementation. Software implementations are easy to develop and maintain contrary to hardware implementations. But hardware implementations have the advantage to be dedicated to encrypt data making them more efficient, reliable[BE02] and potentially more resistant to attack vectors such as side channel leakage. Such advantages explain that SEDs are usually found at a higher price compared to software based solutions. Different SEDs were analysed in [MvG18] and the authors exploited flaws in the firmware of the micro-controllers handling the hardware encryption. This study disproves the common belief that hardware FDE solutions are more secure than software based one. Building a secure FDE solution is however not straightforward, it is a complex mechanism that that involves a versatile skill set. The resistance of the cryptographic mechanism, the key derivation function, the usage of hardware keys are not sufficient: the encryption and the firmware implementations are also part of this mechanism and none of them should be neglected. Having the control of the entire design and production chain, which is the case for SEDs and some smartphones, can help but comes at some cost regarding the implementation neatness and consistency.

Performance. The performance of a FDE solution can encompass the execution time and for some specific usages power consumption.
The execution time depends essentially on the cryptographic algorithm, the way the tool is implemented, the number of the read and write operations. The cryptographic algorithm efficiency includes among others its structure (number of basic operations), its parallelization capabilities for encryption and decryption, the key size. An implementation can be more or less optimized to speed up the execution time. And a dedicated hardware implementation should have a smaller execution time than its software version. Obviously, read and write operations on the disk should be minimized.
It is desirable to limit the power consumption of the system for cost efficiency reasons. For some of them like smartphones, it is crucial because the usability of the smartphone depends on its battery: using too much power on cryptographic computations and extra read/write operations on the memory for data protection is not an interesting deal. It is therefore suitable to have an FDE mechanism that limits energy consumption [FPR12].

2.2.2 FDE and Cryptography

As argued above, writing and reading a sector have to be as quick as possible which implies that encryption/decryption delays have the same requirement. That is why, each sector is

encrypted **independently** and encryption relies on block ciphers for their time **performance**. A secure block cipher reveals no information about the plaintext knowing the ciphertext as long as the key is randomly sampled and kept secret. Today, the most used block cipher is AES (128 bits block length). The sector length is a multiple of the block cipher length and a secure encryption mode has to be used because using sequentially[7] the block cipher on the sector content will lead to insecure encryption. A full disk encryption mode uses the same key [Gjø05a, KMV17] to encrypt the whole disk and it can be the case that the key is composed by sub-keys or used to derive sub-keys.

A widespread FDE mode is CBC-ESSIV [Fru05] for "Cipher Block Chaining-Encrypted Salt-Sector IV". This mode is a CBC mode where the Initialization Vector (IV) is derived from the sector number. In CBC mode, the IV is required for decryption and it is stored as a first block in the ciphertext which is not possible for length preserving encryption. That is why, in CBC-ESSIV, for each sector, the IV is the encryption of the sector number s under a different key k' and the plaintext block are encrypted using the key k. The keys k and k' have to be different otherwise this mode is vulnerable to sliding attacks [Rog04b]. This mode is parallelizable for decryption only and it is secure for an adversary belonging to the threat models -1- and -2-. This mode is less and less used and it should not be used because of its vulnerability to malleability attacks. *A malleability attack* consists in applying a transformation on a ciphertext block i and knowing the impact on the plaintext. To be able to perform a malleability attack, the adversary must have an active access to the disk (theat model -3-). For CBC, flipping the j-th bit on the ciphertext block i will lead to decrypt the plaintext block $i+1$ correctly but with the j-th bit also flipped as shown in Figure 2.6. The i-th plaintext, which is 128 bits for AES, is randomized. This attack has severe impacts: a plaintext bit can be flipped at any position which gives the power to the adversary to change the plaintext block the way he wants. Practical attacks are demonstrated by Lell in [Lel13] on dm-crypt. In the bitlocker solution, the elephant diffuser component[Fer06], which is a keyed diffuser, is applied to the entire sector plaintext to mix all plaintext bits and then CBC-ESSIV is used for encryption. The diffuser makes the malleability attack on CBC unpractical.

Nowadays, the standard for storage devices [IEE08] specifies XTS mode for "XEX Tweakable block cipher with ciphertext Stealing" which is based on XEX for "Xor-Encrypt-Xor" designed by Rogaway [Rog04a]. *A tweakable block cipher* is a block cipher that takes not only the classical inputs, a key and a plaintext block, but also a tweak and outputs the plaintext block. A tweak is a public value and in the case of FDE, it is the sector number then there is no need to store this value. This trick enables to have a length preserving mode to encrypt each sector. A modification on a sector can be seen by a passive adversary (threat model -2-): all unchanged plaintext blocks at the same position will have the same ciphertext blocks and the modified ones will be different. Then it provides a **"spatial" security** in the senses that each encrypted sector blocks look random for the adversary as long as there is no repetition of plaintext block at the same position (security proof [Rog04a]). If the adversary is active and modifies sector content (ciphertext), he is limited to tamper with one block cipher only that corresponds to 128 bits in case of AES usage. For example, it can flip one bit in the sector, after decryption the corresponding plaintext will look random (see Figure 2.7) but all the other plaintext blocks will be decrypted normally. This property makes a tampering attack harder but does not prevent it. Moreover, XTS has the advantage to be parallelizable

[7] This mode is called ECB.

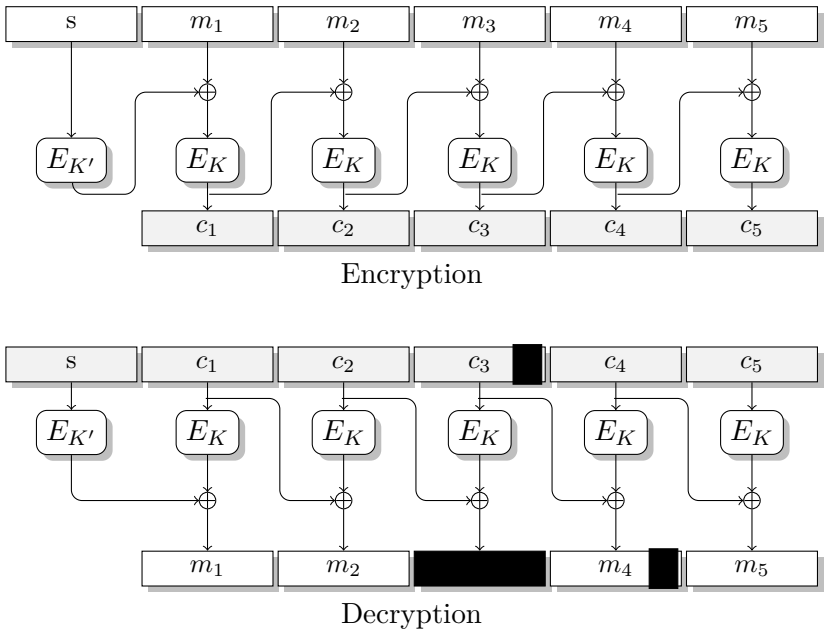

Figure 2.6: Malleability attack on CBC-ESSIV using a block cipher (E_K,D_k) where $D_k = E_k^-$. Bit flipping ("X") in the i-th ciphertext block compromises the entire i-th block and exactly one bit in the i+1-th block.

for encryption and decryption. Moreover, it necessitates only one AES call for each plaintext block[8].

Wide Tweakable Block Ciphers (WTBC) go further in limiting tampering ability of the adversary but require more AES calls per block. A WTBC is based on a standard block cipher that processes input blocks through multiple passes in order to simulate a block cipher over the input size. This is why they are known to be slow[Fer06] and much less used than XTS for example. In FDE context, their block size is the sector size so an adversary, belonging to threat model -3-, corrupting one bit ciphertext will lead to randomize the entire plaintext sector (illustration in Figure 2.8). EME2[HR04, IEE11], CMC[HR03] and XCB[MF07, IEE11] are examples of WTBC. Recently, a new WTBC, Adiantum[CB18], was introduced by Biggers and Crowley that is more performant that XTS when there is no cryptographic accelerator.

[8] At the cost of extra computations over $GF(2^{128})$.

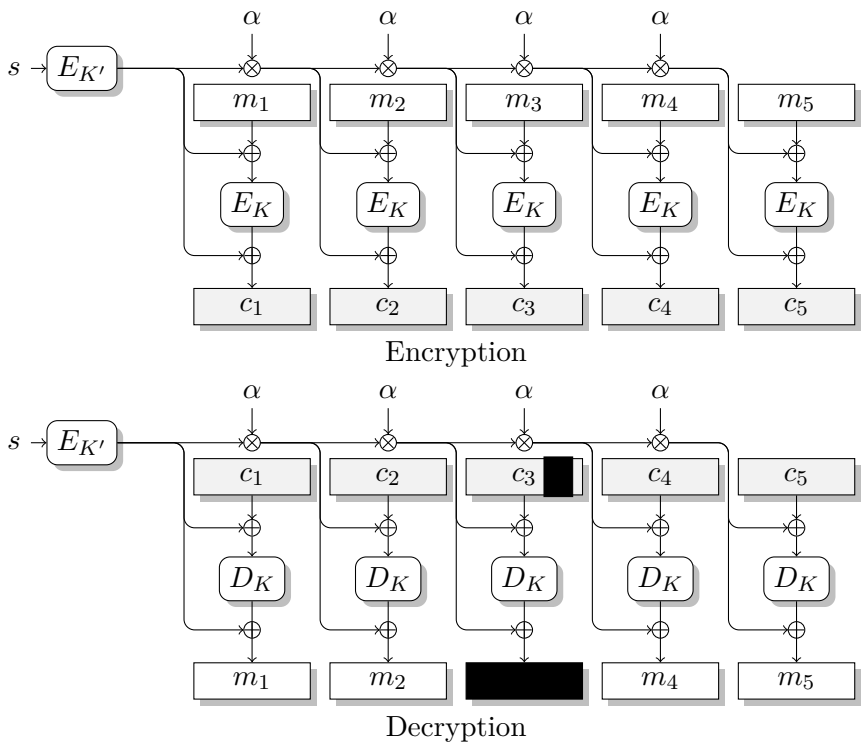

Figure 2.7: Malleability attack on XTS (here without ciphertext stealing) using a block cipher (E_K,D_k) where $D_k = E_k^-$. Bit flipping ("X") in the i-th ciphertext block compromises the i-th plaintext block (black block) after decryption.

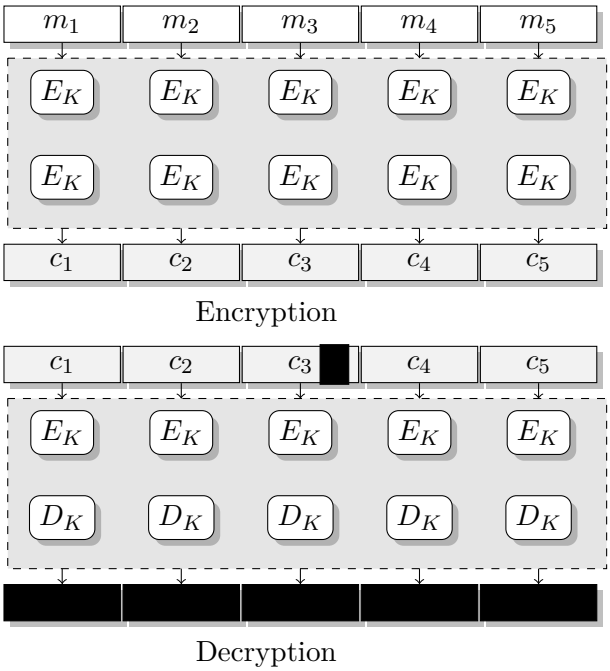

Figure 2.8: Malleability attack on WBTC using a block cipher (E_K,D_k) where $D_k = E_k^-$. Bit flipping ("X") in the i-th ciphertext block compromises all the plaintext (black blockq) after decryption.

2.3 Data Authentication

In the context of full disk encryption the strongest notion of integrity is the *poor man's authentication* which means that if an adversary modifies the ciphertext (here a sector content), one can hope that this modification will lead to a random change in the corresponding plaintext in such a way that the system or the user will detect this tampering. To be able to detect all the illegitimate data modifications, **data authenticity** [9] is required in addition to encryption. In this paper integrity and authenticity refer to the same security notion.

Data authenticity enables to detect if the adversary modified the disk content with a message authentication scheme also called MAC for Message Authentication Code (MAC). It takes as input a plaintext and returns a tag. This primitive is a keyed algorithm based on block-ciphers [fS11a] like CBC-MAC and CMAC or a hash functions [fS11b] like the HMAC family. MACs are preferred over signatures because their computational time is smaller, which is crucial given the amount of data to process on a disk and the expected throughput. Authenticity for disk content is not a new subject [HGX+09, BMM10]; it is the purpose of the standard [IEE18] and was the aim of different recent projects: the dm-integrity framework [BPM18] and dm-x [CJK+17] and StrongBox [IGFH18]. Having data authentication is out of FDE scope due to the storage of additional data [SWZ05, KMV17, BPM18], which is the reason why the definition of new models are required. These new models bring new challenges: how should we store tags, which algorithms minimize the added computational latency (e.g. additional to encryption only latency) and additional data storage. The purpose of this section is to clarify the models where data authentication is possible and to provide solutions and their analysis. Some solutions are practical in the sense that existing frameworks implement the corresponding mechanism and other are, for now, only theoretical.

2.3.1 Local Authenticity

A naive solution is to compute a tag over all the disk content using a MAC and, only one tag is stored for the whole disk. However, then each time a sector is read or written, this tag has to be verified or re-generated which means processing all disk sectors; this is too much time-consuming. A better and natural mechanism is to compute a **local** tag for each sector to keep independence between each sector which costs to store a tag per sector. We call this per sector data authentication **local** or **spatial authentication**.

ADE Model. The possibility to store tags for each data sector gives a model that diverges from the FDE model called **the Authenticated Disk Encryption (ADE) model** in the sequel of the paper. Depending on how these tags are stored (see section 2.3.1.2), we can consider storing more than the local tags to make the cryptographic primitive stronger. That is why from here local tags refer to additional data stored for data protection including cryptographic tags, IV, etc... In this model, the adversary has access to the disk at several times and can modify its content: data sectors and tags sectors. This adversary is part of the threat model -3-, his goal is to break confidentiality or to build a forge for at least one data sector (e.g. data sector d and the corresponding tag τ). He wins if data confidentiality is broken of a sector content or if for the data sector number s, he builds a fresh data sector d^* and it corresponding tag τ^* where verification succeeds. A mechanism secure in the ADE model does not cover replay attacks [vDRSD07]: an adversary having a copy of the disk (or

[9] Data authenticity is different from user authentication.

a part of it) at the time t can replace the disk content at any moment after that time (replay attacks in theft with recovery described in section 2.1.1).

2.3.1.1 Authenticated Encryption (AE)

Here, we can choose among generic compositions or authenticated encryption.

Generic compositions. [BN00] There are three classical generic compositions: MAC-Then-Encrypt, MAC-and-Encrypt or Encrypt-Then-MAC. An encryption scheme and a MAC are composed to achieve data confidentiality and data authenticity, so two keys are needed. These keys can be derived from a master key or generated independently. These solutions store exactly the same amount of extra data (without taking into account the keys) and the choice between them has to be made according to security and calculation efficiency. From a security point of view, the composition Encrypt-Then-MAC is known to be secure if the underlying primitives are secured[10]. For example, XTS-AES can be composed with HMAC-SHA-2 as in [BPM18], but the simple approach will lead to **sector reordering attacks**. As the MAC of a sector does not take as input the sector number, tag reordering will not be detected: an adversary changing the location of a sector together with its tag will obtain a valid MAC verification. The standard for authenticated encryption for storage devices [IEE18] specifies that additional data for each sector should be added and the MAC ensures its authenticity. This is also the case for authenticated encryption.

Authenticated encryption schemes (AE). They are specific schemes that provide confidentiality and authenticity at the same time by design. They take usually a single key and use derivation function(s) to obtain the required key material. For a given plaintext, it outputs a ciphertext and a tag. As they are dedicated to this purpose, they can be more efficient. In 2009, some schemes were standardized among them CCM [fS09, IEE18], OCB [fS09] and GCM [fS09, IEE18] with AES as block cipher. AES-GCM is a widely deployed Authenticated encryption scheme: it is used in protocols such as SSH [IS09], TLS [Res08], IPSec [VM05] and storage[IEE18]. Few years ago, the CAESAR competition [Ber13] was organized with the goal to identify a portfolio of authenticated ciphers that offer advantages over AES-GCM and are suitable for widespread adoption. Specifically, robustness and tolerance against nonce misuse [VV17] (a missing feature of AES-GCM) have been the leitmotiv for the emergence of seven finalists [Ber18] ACRON, AEGIS, Ascon, COLM, Deoxys-II, Morus and OCB.

2.3.1.2 Local Tag Storage

Different ways of storing local tags **on the disk** can be implemented and depending on the chosen strategy; the seek time can be more or less large.

Location. Local tags storage solutions can be: (1) to extend physical sector size; (2) to implement virtual sector smaller than physical sector (illustration in Figure 2.9) or (3) to keep the same sector size and dedicate some of them to store tags.
Solution (1) was proposed in[CMLS15] by arguing that it is feasible for manufacturers to produce disks with bigger sector size because some space is already reserved for checksums.

[10]This is not straightforward as explained in [NRS14], some care must be taken.

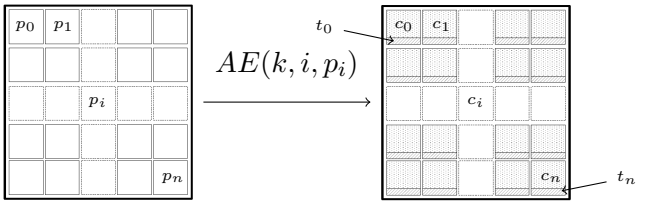

Figure 2.9: Disk Encryption and local authentication tag within each sector. Dashed boxes represent encrypted data and hatched boxes tags. AE stands for Authenticated Encryption.

In this case, the only additional step is to modify the device mapper or the device controller to take into account authentication, but this solution can not be implemented in all deployed disks. Changing physical sector size is expensive so the majority of manufacturers will wait for a standard. Solutions (1) relies on manufacturers.
Solution (2) avoids to wait a new disk format; it is compliant with existing disks. The counterpart is implementing a virtualization layer. A basic virtualization strategy is to split the physical sector into two parts: data and local tags. An example is the FreeBSD disk encryption GELI [Daw14]: 480 bytes of data and a 32 bytes for the tag are stored in a 512 bytes physical sector.
Solutions (3) breaks atomicity contrary to solutions (1) and (2). If, for any reason, some sectors are written and the corresponding local tags are not (or the other way around), the data authenticity check will fail. If a sector writing operation on the disk is interrupted, the previous sector content should be recovered otherwise authenticity check will fail. An examplis dm-integrity: it implements interleaved meta-data sectors [Bro18] where the meta-data include local tags. A fixed number n of consecutive sectors are processed, and the corresponding meta-data is stored in the next sector (see Figure 2.10). The user can choose the number n. The software dm-integrity allows the possibility to manage recovery on write failure with journals: for example, it can save an old data content of unfinished sector write. Enabling journals is an option and experiment with dm-integrity shows that enabling them have severe impacts on performances.

Discussion. Disk sectors are now dedicated to store actual data (D_d sectors) and local tags (D_{la} sectors) for solution (3) but this additional data estimation cost is applicable for all solutions. We have $D_s = D_d + D_{la}$ where D_s is the total number of sectors. Let τ_s be the tag size and S_s the sector size. The maximum number of tags a sector can store is $TS = S_s/\tau_s$. Then we have $D_{la} = D_d/TS$ and $D_d = (TS \times D_s)/(TS + 1)$ which means that the data sectors represent $(100 \times\ TS)/(TS + 1)$ percent of the disk and local tags represent $100/(TS + 1)$ percent. For a disk with a sector size of 4096 bytes ($S_s = 2^{12}$ Bytes) where 1 Terabyte (2^{40} Bytes) of actual data are stored, we have $D_d = 2^{40-12}$ sectors. Let's take the example of the following generic composition: the encryption primitive is AES-XTS encryption where the tweak is the sector number and the MAC primitive is HMAC-256. The HMAC is computed over the concatenation of the ciphertext and the sector number to avoid reordering attacks. The local tag size per sector data is 32 Bytes which corresponds to the tag produced by HMAC-256 ($\tau_s = 2^5$ Bytes). Then a sector stores 128 local tags ($TS = 2^7$) which gives $D_{la} = 2^{28-7}$ sectors (8 GB). Additional data represent only 0,8% of the disk. It

is important to keep in mind that this estimation does not consider journals.

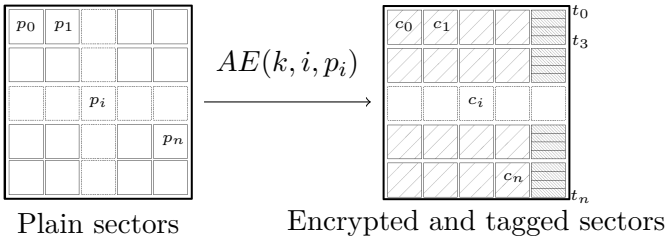

Figure 2.10: Interleaved Meta-data. Large dashed boxes represent encrypted data and small dashed boxes meta-data.

Data authentication protecting against replay attacks is a desirable property; we can call this security notion **temporal authenticity** or **global authenticity**. The next section aims at analyzing how we can get such property and at estimating its cost.

2.3.2 Global Authentication and Incremental Cryptography

Global authenticity is a strong security notion that aims at protecting all the local tags from tampering. A simple solution is to store all these tags in **persistent secure memory**, which means the adversary cannot tamper with it.

Secure Memory (SM). Obviously, such a component is not an accessible part of the disk otherwise downgrade attacks are still applicable. Now, we will assume that such memory exists otherwise replay attacks cannot be prevented in the context of full disk encryption [HGX+09, BPM18] and also in the context of secure cloud storage with block-level encryption [OR05, CJK+17]. Disk and cloud storage are similar from a theoretical point of view in the sense that the server can be seen as a big remote disk. These two use cases are different; nonetheless the data authenticity problem for data stored in a physical disk or in a distant disk (cloud) can be solved similarly. For disk protection, we can consider a Secure Element[11] as a SM but it can store only a limited amount of data typically few kilobytes to few megabytes[STM17]. Whereas for cloud storage, the SM can be associated with a dedicated client local storage where it is possible to store data that cannot be tampered with by the server. In this case, the SM can be a whole disk partition that the local OS protects from any server access, yielding in few gigabytes of "secure memory" size.
Once again, the naive idea is to compute a global tag over all the local tags. Then reading a sector will lead re-computation over all the local tags which is too much time-consuming.

FADE Model. Once more, we have a different model that is slightly deviating from the ADE one called the **Fully Authenticated Disk Encryption (FADE)** where the adversary has the possibility to perform any modification on the disk. It breaks data authentication if it succeeds to build a forge for a sector s different from the last legitimate one. In this model, the adversary can attempt replay attacks.

Global MACs. To be secure in the FADE model, a global MAC is needed, and it should have specific properties to be effective in the FDE context.

[11] This element usually embeds a small and persistent memory.

-**Security**: It has to guarantee the authenticity of the local tags.
-**Speed**: It has to be fast for tag generation and tag verification. It has to minimize cryptographic operations but also the added read and write access to the disk: tag generation and verification must be relative to a data unit.
-**Locality**: If the verification fails, the index of tampered sectors should be easy to find.
-**Minimal storage in the disk**: If additional data has to be stored in the disk it should be as short as possible to lose a minimum space in the disk.
-**Minimal storage in secure memory** As argued above, a global tag has to be stored in secure memory that has limited storage space.

Incremental Cryptography.

Incremental MACs can fulfil some properties above. Incremental cryptography was introduced by Bellare, Goldreich and Goldwasser in 1994 [BGG95] and it is an attractive feature that enables to efficiently update a cryptographic output like a ciphertext, a signature or an authentication tag after modifying the corresponding input. A MAC can be incremental regarding some update operations like replacing, deleting or inserting a block in the input. In our case, the input is the concatenation of the sector local tags computed with a classical MAC e.g, not an incremental MAC then for each modification in a sector; the local tag will be recomputed, and the global tag (output of incremental MAC) will be updated.

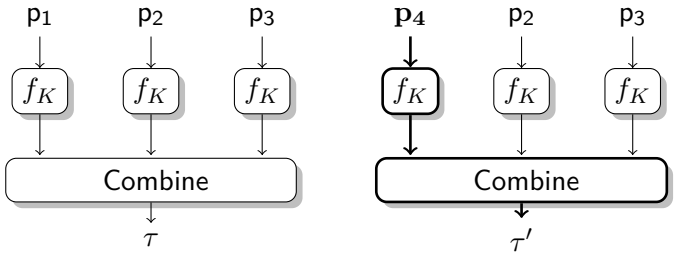

Figure 2.11: Overview of incremental replace operation. The function f_K denotes the MAC primitive. The inputs τ_1, τ_2, τ_3 give the tag τ. After replacing the value τ_1 by τ_4, the tag τ is updated with τ'.

Usually, this efficiency comes from some independent MAC over all the inputs (values p_i in Figure 2.11) and the resulting output block are combined to obtain the global tag τ. This independent processing seems to fit the independence of the disk sectors: as the sectors are modified independently, we can imagine a root tag τ for all the disk that is updated for each sector modification without recomputing the root tag from scratch (otherwise it would be inefficient). A disk is a fixed number of sectors, and local tags depend on the sector number so having an incremental MAC regarding the replace operation only is sufficient. The chaining Xor-Scheme [BGG95][KV18], Xor-MAC [BGR95a] and GMAC [KVW04] and Merkle tree [BGG95] are incremental MACs. These algorithms use a keyed function f_K[12] and, they are compared in table 2.1 with regard to the computational time of their tag generation algorithm, the replace update operation and the tag verification algorithm. The table gives the number of calls to f_K for a disk composed of n sectors which means that the input of the tagging algorithm is the concatenation of the sector local authentication tags. In

[12]To be more precise, f_K is a pseudo-random function.

	Tag	Replace	Verify	Storage	L
Chaining XS	n	4	n	1	N
Xor-MAC	$n+1$	2	$n+1$	1	N
Merkle tree	n	$log(n)$	$log(n)$	n	Y

Table 2.1: Operation and Storage costs of incremental MACS for n-block input. The column L is for *Locality property.*

the following, *log* is for binary logarithm. For n inputs, the Xor-Scheme and the Xor-MAC have a constant time replacing operation (respectively 4 and 2 calls to f_K) and the storage cost is only a tag size. The main drawback is the verification cost: to verify the authenticity of 1 sector tag; it requires n calls to f_K. Merkle tree is the only incremental scheme that has the locality property and a trade-off between replace and verify operation delays. The counterpart is that it requires more storage space, but it is not necessary to store the entire tree. In the next section, the Merkle tree storage will be discussed in details.

2.3.3 Merkle Tree

A Merkle tree is an authentication data structure where leaves are the values to protect, and each node is the MAC[13] of the concatenation of its children. In the following, Merkle trees are binary trees which means that each node is the MAC of its two children. This construction is used to ensure data authenticity at block level [OR05, HGX+09, HWF15, dm-19b], in cloud storage context [OR07, HPPT08, CJK+17] and at file level [RCPS07] due to its incremental and locality properties. For a disk of n sectors, Merkle tree leaves are the n local tags where

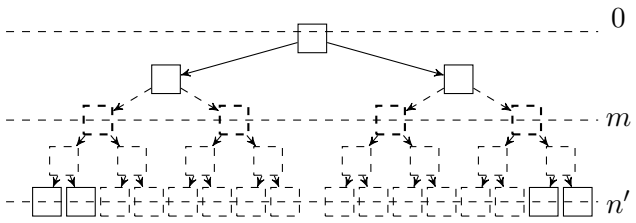

Figure 2.12: Perfect Merkle tree of n' levels. Nodes of level m only are stored ($m \leq n'$).

$\mathsf{n} = 2^{\mathsf{n}'}$.
This estimation is given for a perfect binary tree. A tree is said to be a *perfect binary tree* when all its nodes have two children, and all leaves have the same depth. So the number of actual data is exactly 2^n where n is an integer. In this paper, all the Merkle tree storage cost are computed according to a number of leaves (tags) equal to a power of 2 e.g. where $n = 2^{n'}$ and $n' = log(n)$. The entire tree consists of $2^{\mathsf{n}'} - 1$ nodes. Otherwise, the value n' can be set to $\log(n) + 1$ which is the worst case e.g. non-optimized tree. Let L be the output size of the MAC algorithm f_K used to compute the Merkle tree nodes. Then storing an entire Merkle tree takes $(2^{\mathsf{n}'} - 1)L$ bits.

[13]To be more precise, Merkle tree uses a pseudo-random function which is the case of MACs.

There are different memories where it can be stored: the SM, the disk and the RAM. But in any case, a copy of the trusted root has to be stored in the SM. This value must not be tampered with in order to recompute the path from the sector local tag to the root and check the obtained root and the stored one. To simplify the analysis, we distinguish two ways of storing the tree: either it is entirely stored in the same memory which is said to be a *stand-alone storage*, or the tree is split into different memories which is said to be an *hybrid storage*. It is possible to store some strategic nodes and to recompute the corresponding subtrees when needed.

The nowadays RAM modules can store a large amount of data; typically between 4 GB and 16 GB. In addition to current computation data, the RAM could store a part of the Merkle tree. In the following, an estimation of the update/verify times of a sector and, the storage cost in the different memories is given depending on the storage configurations. Some of the following solutions are suitable for disk protection but, once again it depends on the device: for a laptop, the disk size is in the range of gigabyte or terabyte, the RAM size is rarely larger than 32 GB and is in average around 8 GB. The secure memory is usually only few kilobytes. The **access latencies** of the different memories are also a parameter to take into account.

Cost	**S1**	**S1'**	**S2**	**S3**	**S4**	**S5**
Up/Ver	n'	1	$2^{\mathsf{n}'}-1$	n'	n'	n'
RAM	-	-	-	$2^{\mathsf{n}'}-1$	-	$2^{\mathsf{n}'}-1$
Disk	0	0	0	0	$2^{\mathsf{n}'}-1$	$2^{\mathsf{n}'}-1$
SM	$2^{\mathsf{n}'}-1$	$2^{\mathsf{n}'}$	1	1	1	1

Table 2.2: Estimation of Stand-alone storage of Merkle tree with n actual data sectors where $\mathsf{n}=2^{n'}$. Up/Ver stands for update and verify cost.

2.3.3.1 Stand-alone Storage

Table 2.2 gives 5 possibilities for Merkle tree storage where the entire Merkle tree is stored in the SM, the disk or the RAM.

- S1: The entire Merkle tree is stored in the SM and for each read and write operation, the update/verify operation on the Merkle tree costs n' calls f_K. This solution does not use disk and RAM storage but the update/verify which has a reasonable cost. If it is possible to store $2^{\mathsf{n}'}-1$ tags then it should be possible to store $2^{\mathsf{n}'}$ which corresponds to solution S1'. In this case, all the local tags can be stored in the SM and, as it is assumed that this memory is tamper proof, the global authentication scheme is no longer needed: the local tags are stored directly in SM. Then the Merkle tree is not needed any more. If the SM is big enough, this solution should be considered otherwise the following possibilities, where only a small value (the root) is stored in SM, are more suitable.

- S2: The only value stored is the Merkle tree root and it is stored in the SM. The advantage is that the storage on the disk, in the SM and RAM is minimized but the

update/verify cost is the worst: the entire Merkle tree has to be recomputed for each update/verify operation. For this reason, this solution seems to be unsuitable for disks.

- S3: This solution considers a context where the RAM is big enough to store the entire Merkle tree and operates in rated conditions. The access time to Merkle tree nodes in the RAM is smaller than in the disk but the counterpart is that the Merkle tree will not be maintained when the system is switched off. It has to be recomputed each time the system is switched on. This solution could be interesting if saving disk storage is a priority. In fact, it stores only the root in SM and no additional data in the disk. The update/verify operation cost is reasonable but a re-computation delay due to the volatility of the RAM is added.

- S4: This solution has the same settings than S3 except that the Merkle tree is not stored in RAM but in the disk. It has the advantage to remove the re-computation delay, but this time the additional storage in the disk is the entire Merkle tree size. Moreover, the time to read and write a node will be larger as it is stored in the disk. Unsurprisingly, this solution could be implemented in an optimized manner by some developers as solution S5.

- S5: This solution combines solutions S3 and S4: the tree is stored in the disk and also in the RAM. Each time the device is switched on, the Merkle tree is copied from the disk to the RAM which avoids the re-computation delay of solution S3. This solution is the most efficient from a speed point of view.

For a reasonable amount of data, solution S1' can be considered for cloud storage as few gigabytes can be stored in the client side, but it seems unpractical for a stand-alone disk. Solution S5 stores only the root in the SM and is the quickest solution described above, which makes it the most suitable for stand-alone disk.

Cost	**S1**	**S1'**	**S2**	**S3**	**S4**	**S5**
Up/Ver	28	1	268435455	28	28	28
RAM	-	-	-	8 GB	-	8 GB
Disk	0	0	0	0	8 GB	8 GB
SM	8 GB	8 GB	32 B	32 B	32 B	32 B

Table 2.3: Estimation of Stand-alone storage of Merkle tree where $\mathsf{n} = 2^{28}$ and $L = 32$ Bytes

Discussion. Table 2.3 gives the Merkle tree storage cost in RAM, in the disk and the SM with the same settings than in section 2.3.1.2. As explained in the analysis of S2, the update and verify operations cost a large amount of calls to the function f_K. The same problem will emerge each time the device is switched on for solution S3 due to storage in RAM. Solutions S1 and S1' are not suitable for the rather small sized SM we discussed in section 2.3.2. Unsurprisingly, solution S5 seems to be the most suitable for disk protection.
In these settings, 8 GB are needed to store local tags and 8 GB for global storage then for solutions S4 and S5, the FADE mechanism costs 1,5% of the entire disk.

Cost	H1	H2	H3	H4	H5
Up/Ver	$2^{\mathsf{n}'-m}-1+m$	$2^{\mathsf{n}'-m}-1$	$\mathsf{n}'-m$	$\mathsf{n}'-m$	n'
RAM	-	-	$2^{\mathsf{n}'}-2^{m+1}$	-	$2^{\mathsf{n}'}-2^{m+1}$
Disk	$2^{m+1}-1$	0	0	$2^{\mathsf{n}'}-2^{m+1}$	$2^{m+1}-1$
SM	1	2^m	2^m	2^m	1

Table 2.4: Estimation of Hybrid storage of Merkle Tree with n actual data sectors where $\mathsf{n}=2^{n'}$ and $m \leq n'$. Up/Ver stands for update and verify cost.

2.3.3.2 Hybrid Storage

Table 2.4 presents possibilities where the Merkle tree nodes of level m, where $m \leq \mathsf{n}'$, are stored. Here, the performance depends on the choice of m and, it has to be instantiated in table 2.4. The optimum value of m depends on many factors: the disk, the SM and the RAM sizes, the CPU, the cryptographic algorithms etc. Due to this variety of factors, this analysis is only theoretical: it provides insights about the main tendencies of each implementation strategy. It would be nonetheless interesting to test these solutions in different devices to adjust the level m and compare their real performances.

- H1: This solution minimizes the storage in SM by storing the Merkle tree root only. The top of the Merkle tree, from the root to level m, is stored in the disk. For each update/verify operation, recomputing the corresponding subtree costs 2^m-1 calls to f_K and the update/verify of the top of the tree cost m calls to f_K. If a verification fails, it will not be possible to find exactly which sector was tampered with, in the best case, only the node at level m can be given as an information to the user.

- H2: The nodes at the level m (2^m nodes) are stored in the SM. Here, these nodes do not need data authenticity check; they are assumed to be authentic as they are stored in the SM. For each update, the needed subtree is recomputed which costs $2^{\mathsf{n}'-m}-1$ calls to f_K. Here, the SM has to store $2^m L$ bits. Depending on the value m and the SM size, it might be possible for a disk. For instance, if the SM can store 1 MB and the $L=256$ bits, then the maximum value of m is 12.

- H3: The nodes of the level m are stored in the SM and, the m subtrees are stored in the RAM. Each update operation costs going through the subtree, which requires $n'-m$ calls. Storage in the RAM involves a re-computation time when the device is switched on.

- H4: The nodes of the level m are stored in the SM and all the subtrees (m) are stored in the disk. Unlike, the previous solution, accessing tree nodes in the disk takes more time.

- H5: The Merkle tree is split between all the memories: the root in SM, the top in the disk and the corresponding subtrees in the RAM. Because of the storage in the RAM, a recomputing delay is added each time the device is switched off. The update/verify time is acceptable.

Solution H1 seems to be the most suitable for disk encryption, the SM storage is reduced to the Merkle tree root, the upper part is stored in the disk, and some layers have to be recomputed. These solutions seem to remain acceptable for disk encryption as long as the value m gives decent storage cost in the SM and, also in the RAM. Solutions where update/verify time is minimized, seems to be better as this latency has a direct impact on the disk performances.

Cost	**H1**	**H2**	**H3**	**H4**	**H5**
Up/Ver	8207	8191	13	13	28
RAM	-	-	8 GB	-	8 GB
Disk	2 MB	0	0	8 GB	2 MB
SM	32 B	1 MB	1 MB	1 MB	32 B

Table 2.5: Evaluation of the Hybrid storage of Merkle tree with $n = 2^{28}$, $m = 15$, $L = 32$ Bytes.

Discussion. Table 2.5 gives an estimation for the value $m = 15$ with the same settings than in the discussion for stand-alone storage. The value m was chosen to have maximum storage in SM equal to 1 MB. We can note that the update and verification time is shorter for solution H3 and H4 whereas it is quite long for solutions H1 and H2. Storage in RAM in solutions H3 and H5 adds a re-computation delay that should be limited as much as possible. Solution H4 seems to be the best solution if the SM can store 1 MB otherwise H5 has to be considered. The storage cost for the entire FADE mechanism for H4 is close to solutions S4 and S5, which represents about 1,5% of the disk, however, the update and verify time is better.

Cost	**H1**	**H2**	**H3**	**H4**	**H5**
Up/Ver	32	7	3	3	28
RAM	-	-	6 GB	-	6 GB
Disk	2 GB	0	0	6 GB	2 GB
SM	32 B	1 GB	1 GB	1 GB	32 B

Table 2.6: Evaluation of the Hybrid storage of Merkle tree with $n = 2^{28}$, $m = 25$, $L = 32$ Bytes.

In section 2.3.2, the SM was limited to a few megabytes and corresponds to realistic figures of current state of the art of Secure Elements. Other technologies embedding more secure memory have raised the industry attention in the last years. Examples of such solutions are those belonging to the Trusted Execution Environment (TEE) ecosystem, with TrustZone, Intel SGX and so on. In the TEE paradigm, a Secure Element is ”emulated” in the form of a sandboxed execution mode (the ”Secure World”) of a general purpose processor, yielding in more computing power and storage space when compared to classical Secure Elements. Even though the storage space of such solution is usually shared with the so called ”Non Secure World”, more and more SoC vendors embed non-volatile memory dedicated to the TEE with hardware security isolation insurance. In such components, the SM considered in our models could reach hundreds of megabytes to gigabytes of internal storage, and table 2.6

gives figures in a case where 1 gigabyte can be stored in the SM. Software implementations of FADE using TEE as a SM have been proposed in [HWF15]: the Secure Block Device Library uses CMAC and Merkle trees to bring confidentiality and integrity to Trusted Applications data at rest storage. Temporal integrity is ensured whenever the underlying SoC provides a physical secure storage with non-tampering properties: the Merkle tree root as well as the master encryption key are then stored inside it.
In tables 2.2, 2.3, 2.4, 2.5 and 2.6, the path of the Merkle tree (or a part of it in the case of hybrid storage) stored explicitly in RAM is updated and verified for each read and write operation. As attacks in RAM are excluded in the presented models, the verification path can be omitted to speed up the read operation of a sector. Doing so for the write operations is more tedious: after some number of writes, several paths in the tree are updated. These update leads to the refreshing the tree in SM (which can be reduced to the root or more nodes depending on the chosen solution) just before powering down the device. In case of failure (device out of battery for instance), the device has to be able to update the tree in SM otherwise the integrity check will fail for all sectors lately updated. Hence, this optimization seems suitable for SEDs that embed their own emergency batteries, leaving enough time to perform such updates.

Protection	**Name**
FDE	Bitlocker[Fer06] Veracrypt[ver19] dm-crypt[dm-19a] FileVault[CGM13] CRYHOD[Pri19]
ADE	dm-integrity[BPM18] GELI[Daw14]
FADE	Secure Block Device Library[HWF15] dm-x [CJK+17] StrongBox [IGFH18]

Table 2.7: Some implementations giving different levels of security for a stand-alone disk.

Conclusion. Today, products implementing disk encryption are widespread. Most of these products use the classical FDE mode of operation, namely XTS. However, adding efficient data authenticity in the strong model (FADE) with tight constraints is an important yet unsolved practical challenge. Developers and researchers have begun to address the problem since a few years which gives some preliminary solutions (see table. 2.7). The global authentication mechanism is new from a cryptographic perspective, and the question is whether we can do better than Merkle trees. From an architectural point of view, the introduction of a secure memory in laptops as well as in smartphones seems to be a strong requirement to have global authentication. This is out of scope in this thesis, but there is a crucial question for the deployment of FADE solutions: if the secure memory exists then it is accessible to a standard FADE software for embedding authentication data (such as a root of a Merkle tree), or is it only accessible to device manufacturers, hence entailing proprietary implementations? The next step is to implement FADE mechanism with the Merkle tree in different devices with different configurations: stand-alone storage, hybrid storage and in this case we should find the trade-off for the value m.

Chapter 3

Full Disk Encryption

Contents

3.1 Disk encryption methods and Security notions . . . 38
3.1.1 Disk Encryption Methods . . . 38
3.1.2 Security Notions for FDE . . . 41
3.2 FDE Security with Unique First Block . . . 48
3.2.1 CBC-ESSIV Security . . . 48
3.2.2 IGE-ESSIV Security . . . 52
3.3 FDE Security with a Diversifier . . . 54
3.3.1 Solid State Drive . . . 55
3.3.2 Case Studies . . . 58

The term Full Disk Encryption (FDE) is commonly used when every sector of a disk volume is encrypted. There is typically no space to store any additional data, such as an IV or a MAC. As explained by Ferguson [Fer06], generic solutions to store additional data will at least double the number of read and write operations, and will significantly reduce the available disk space. They also change the disk layout, which makes it extremely complicated to enable FDE on existing disks.
With this restriction, FDE cannot offer authentication, but at best "poor-man's authentication" [Fer06], which is to hope that ciphertext changes will result in a plaintext that is random enough to make the application crash. It can also not achieve chosen-plaintext indistinguishability: when the same data is encrypted twice at the same sector index, the resulting ciphertexts will be identical.
Additional efficiency constraints may be imposed on FDE as well. For example, it can be desirable to perform encryption and/or decryption in parallel, which is not possible for inherently sequential constructions where one plaintext block of a sector cannot be processed until all previous plaintext blocks are processed. If there is not enough memory available to store an entire sector, it may be required that encryption and decryption are online, meaning that one block of a sector may only depend on the preceding blocks.
These implementation constraints impose severe restrictions on the algorithms that can be used for FDE. However, a general problem in the domain of FDE is that the security properties of the resulting constructions are not always well-understood. And, cryptographers often complain about the absence of well-defined cryptographic goals for FDE (see e.g. Rogaway [Rog11]), which are prerequisites to find a good-trade-off between security and efficiency.

In this chapter, we want to measure "how much security is left" within the constraints of FDE. In order to do so, we introduce a theoretical framework to capture that an FDE algorithm behaves as "randomly as possible" subject to different practical constraints. We consider settings where the encryption oracle can be *random-up-to-repetition*, *random-up-to-prefix* or *random-up-to-block*. For each of the attack settings in the framework, we list an efficient construction that achieves security within this setting. We recall existing security results, and provide new proofs, in particular in the *unique-first-block* (ufb) setting where the Operating System (OS) or application ensures that the first n bits of the plaintext will not be repeated for a particular sector number, where n is the block size of the underlying block cipher.
Our model recalls that the modes of operation CBC (Cipher Block Chaining) and IGE (Infinite Garble Extension), even with a secret IV, do not achieve the security properties that developers often wrongly assume for these constructions. As already shown by Bellare, Boldyreva, Knudsen and Namprempre [BBKN12], CBC and IGE are not IND-CPA secure up-to-prefix. We will prove, however, that both constructions are IND-CPA secure under the ufb assumption.
Regarding chosen-ciphertext attacks, we point out that Added Redundancy Explicit Authentication (AREA) [Fru05] is not secure when used with CBC or IGE, even when the IV is secret. The insecurity of constructions such as AREA was already shown in 2001 by Jutla [Jut01], but has nevertheless not yet been pointed out in the context of FDE. We recall that there exist constructions that are secure in this setting, such as TC2 and TC3 [RZ11].
Secondly, we revisit the FDE constraints from an engineering point of view. We show that it is possible to produce different ciphertexts for the same plaintext at a particular sector index, without storing additional data. Our solution applies to solid state drives (SSDs), where we

show how the SSD firmware can be modified to associate a *diversifier* to every sector. This is done without modifying the data structures of the SSD, but by forcing data to be written to a particular Logical Unit Number (LUN).
For any particular sector, the diversifier value must be unique. However, as we will explain later, additional requirements are necessary for performance reasons. When looking at all sectors at any particular point in time, each diversifier value should occur roughly the same number of times. Additionally, this diversifier value can typically only be a few bits long. These requirements put the diversifier in a class by itself, and not as a specific case of a random IV or a nonce (i.e. a number that is only used once).
When we benchmarked our solution in a modified EagleTree simulator [DSB+13], we found that it increases the average latency by at most 12 % for reads and 2 % for writes, and that it reduces the SSD throughput (read and write combined) by less than 4 %.

3.1 Disk encryption methods and Security notions

The problem of FDE has been researched extensively, see for example Rogaway [Rog11] for a provable security treatment, or Fruhwirth [Fru05] for an implementer's perspective. The formal requirements of disk encryption are often not clearly stated.
FDE is a topic that has gathered significant interest from industry and standardization and often leads to application-specific solutions due to the special requirements of full disk encryption. Of particular interest are the elephant diffuser used in Microsoft's BitLocker [Fer06], or IEEE P1619's XTS standard [IEE08], which later became a NIST recommendation [Dwo10] as well.
We assume that adversary has access to the disk volume at any time. The adversary has (partial) knowledge and even control of the plaintext, and can even change the ciphertext as well. We, therefore, go beyond just "single point-in-time permanent offline compromise" (see e.g. [HSTM15, Gjø05b]). Read and write operations are assumed to be atomic (on a sector level), so we do not consider blockwise adaptive attacks [JMV02].
Sound key management is required to avoid that the plaintext contains the key or any function of the key [Hal06]. Physical access threats (e.g. cold boot, DMA, evil maid, or hot plug attacks [MF15, GM14]) are also outside the scope of FDE model.

3.1.1 Disk Encryption Methods

Data is read and written in a sector-addressable device by fixed-length units called sectors, usually 512 or 4096 bytes long. The OS can access a specific sector by its *sector number* s. We consider the case of an encrypted disk volume where data is encrypted by the OS before being stored.

We list the modes of operation that frequently appear in the context of FDE, whether it be in academic literature or practical implementations. We also mention other modes with interesting security or efficiency properties.

ECB (Electronic Codebook). In the simplest encryption mode, the plaintext is divided into blocks of n bits, and each block is encrypted separately using an n-bit block cipher. It can readily be used for FDE, even though it is well-known that it does not provide adequate security.

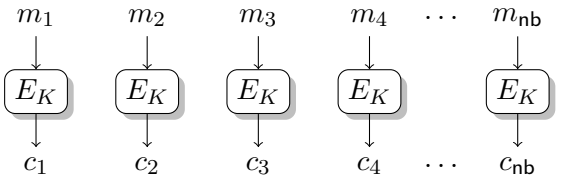

Figure 3.1: Description of the mode of operation ECB (*Electronic Codebook*) where E is a block cipher.

CTR (Counter). This mode uses a counter (incremented for each block) that is encrypted and then XORed with the plaintext block to output the ciphertext block. Typically, the counter is the sector number, bit-shifted to the left over a sufficient number of bits so that the least-significant bits can represent a counter for the number of blocks in one sector. CTR mode is IND-CPA secure [BDJR97] under the assumption that the counter is a nonce. In the context of FDE, this assumption does not hold as sectors can be overwritten.

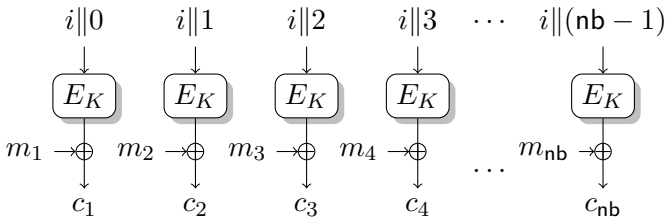

Figure 3.2: Description of the counter mode (CTR) where E is a block cipher and i is the counter.

Cipher Block Chaining (CBC). In this mode, each plaintext block is XORed with the previous ciphertext block (or an IV for the first plaintext block) before being encrypted. To be able to achieve the IND-CPA security notion, it is well-known that the IV has to be a random value [BDJR97]. However, for FDE, the first natural idea is to use the sector number as an IV. Fruhwirth [Fru05] proposed to use as an IV the encryption of the sector number by the block cipher keyed with an independent key (see Figure 3.3).[1]

Infinite Garble Extension (IGE). IGE was proposed by Campbell [Cam78] as a variant of CBC mode where each block of plaintext is XORed with the *next* ciphertext block (see Figure 3.3). For FDE, since the sector number is not secret, we will consider the variant where the IV is the encryption of the sector number s in which s is not XORed to the first ciphertext block. We will refer to this mode as IGE-ESSIV.

XEX with CipherText Stealing (XTS). XTS [IEE08] applies a tweakable block cipher to every n-bit block of a sector, where the tweak depends on the sector number and the

[1] Two distinct keys are needed: the message is encrypted with key K, and the IV is encrypted with key $K' \neq K$ (see Figure 3.3), in order to avoid an attack by Rogaway [Rog04b].

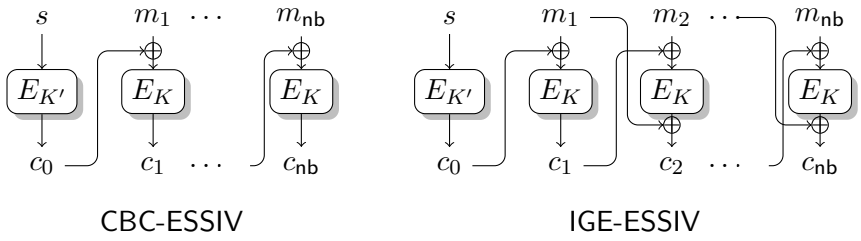

Figure 3.3: Description of the CBC-ESSIV and IGE-ESSIV modes of operation where E is a block cipher and the keys K and K' are independent.

index of the block within the sector. It uses *ciphertext stealing* when the sector size is not a multiple of n bits; however, such sectors sizes are not considered in this thesis.

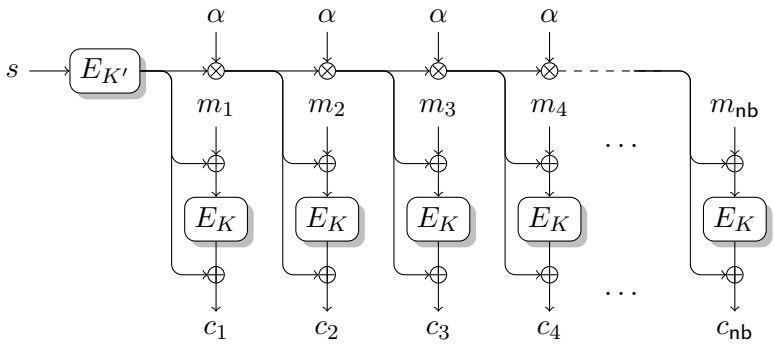

Figure 3.4: Description of the XEX mode of operation where E is a block cipher and K and K' are independant key.

Tweakable Ciphers TC1, TC2 and TC3. These modes for tweakable block ciphers were defined by Rogaway and Zhang in [RZ11]. The difference between these constructions is the way the tweak is used:

- In TC1 (see Figure 3.5), the tweak is the previous ciphertext block as in the HCBC mode [BBKN12].
- In TC2 (see Figure 3.5), the tweak is the concatenation of the previous ciphertext block and the previous plaintext block as in the HCBC2 mode [BBKN12];
- In TC3 mode (see Figure 3.6), the tweak is the XOR of the previous ciphertext block and the previous plaintext block as in the MHCBC2 mode [Nan08].

In FDE context, it is natural to consider a mode where the first block tweak is simply the sector number s.

Wide Tweakable Block ciphers WTBC. In the context of FDE, the block size of a WTBC is equal to the sector size, and the sector number is used as the tweak input. From a security point of view, any change in the plaintext or ciphertext affects the entire sector. A

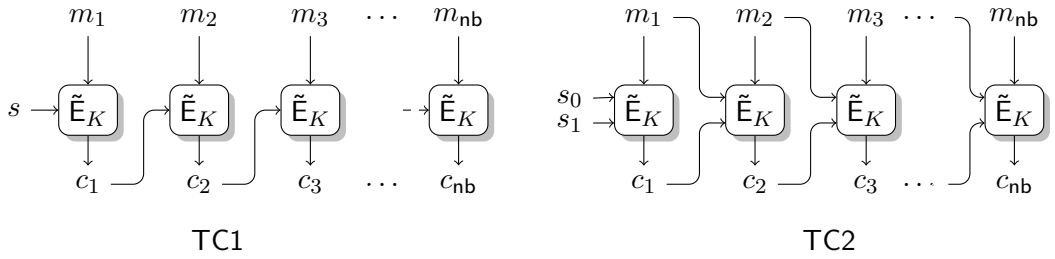

Figure 3.5: Description of the TC1 and TC2 modes of operations where $\tilde{\mathsf{E}}$ is a tweakable block cipher. In FDE context, the tweak can be the sector number s (for TC2, $s_0||s_1 = \mathsf{P}(s)$ where P is a permutation.

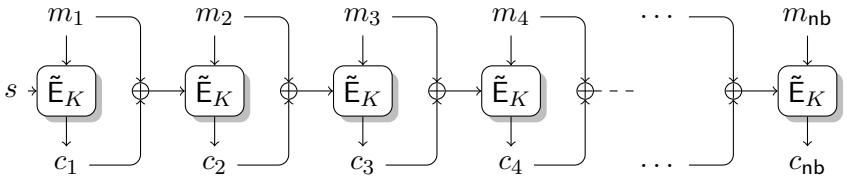

Figure 3.6: Description of the TC3 mode of operation where $\tilde{\mathsf{E}}$ is a tweakable block cipher. The tweak can be the sector number s.

WTBC is typically realized using smaller (tweakable) block ciphers, for example in the EME (Encrypt-Mix-Encrypt) [HR04] mode.

Our goal in this chapter is to analyse these constructions and to evaluate their security in different models.

3.1.2 Security Notions for FDE

In this section, we formalize several security notions for FDE. We first give a formal syntactic definition of block-cipher-based FDE.

It is assumed that the plaintext of a sector is a multiple of n which is the block cipher size. All plaintexts are nb blocks of n bits. $m[i]$ denotes the i-th block of the plaintext m such that $m = m[1]||m[2]||...||m[\mathsf{nb}]$ where $||$ denotes concatenation of strings. IND-CPA-xx corresponds to IND-CPA up-to-block, IND-CPA up-to-prefix, IND-CPA up-to-repetition and IND-CPA.

Definition 3.1.1 (FDE scheme)**.** A *block-cipher-based FDE scheme* is a 7-tuple of parameters and algorithms $\mathsf{FDE} = (\mathsf{KS}, \mathsf{SS}, \mathsf{BS}, \mathsf{MS}, \mathsf{kg}, \mathsf{enc}, \mathsf{dec})$ such that:

- KS is the key space, SS is the sector space;
- BS is the block message space and MS is the message space such that $\mathsf{MS} := \mathsf{BS}^{\mathsf{nb}}$ where $\mathsf{BS} = \{0,1\}^{\mathsf{n}}$; n and nb are a fixed positive integers;
- $\mathsf{FDE.kg}$ is the (probabilistic) key generation algorithm that takes no input and returns a key $K \in \mathsf{KS}$;

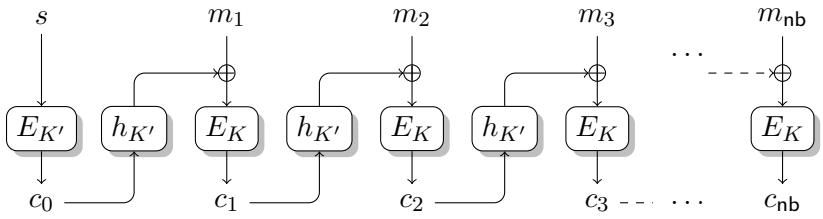

Figure 3.7: Description of the HCBC1 mode of operation where E_K is a block cipher, $h_{K'}$ a keyed hash function. The keys K and K' are independent.

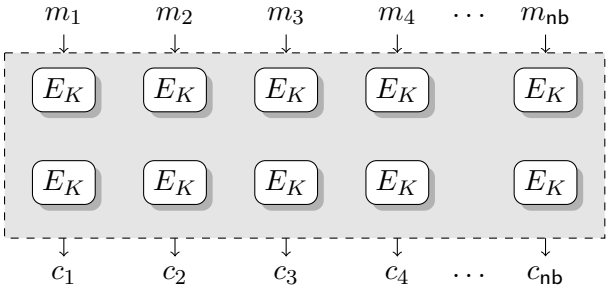

Figure 3.8: WTBC where E_K is either a block cipher or a tweakable block cipher. WTBC can use several call to E_K to process one message block.

- $\mathsf{FDE.enc}$ is the (deterministic) encryption algorithm that takes as input a key $K \in \mathsf{KS}$, a sector number $s \in \mathsf{SS}$ and a *plaintext* $m \in \mathsf{MS}$ and outputs a *ciphertext* $c \in \mathsf{MS}$;
- $\mathsf{FDE.dec}$ is the (deterministic) decryption algorithm that takes as input a key $K \in \mathsf{KS}$, a sector number $s \in \mathsf{SS}$ and a ciphertext $c \in \mathsf{MS}$ and outputs a *plaintext* $m \in \mathsf{MS}$,

such that $\forall (K, s, m) \in \mathsf{KS} \times \mathsf{SS} \times \mathsf{MS} : \mathsf{FDE.dec}(K, s, \mathsf{FDE.enc}(K, s, m)) = m$.

In the rest of the thesis $\mathsf{FDE.nb}$ denotes the number of blocks $m[i] \in \mathsf{FDE.BS}$ of message $m \in \mathsf{FDE.MS}$.

For each security notion, we define two variants: security under Chosen-Plaintext Attack (CPA) where the adversary is given access to the ENCRYPT procedure and security under Chosen-Ciphertext Attack (CCA) where the adversary is also given access to the DECRYPT procedure. The adversary is not allowed to query the decryption of a ciphertext that was previously returned by ENCRYPT or vice versa.

INDISTINGUISHABILITY UP-TO-BLOCK. In this definition, each ciphertext block depends deterministically on the plaintext block, the sector number s and the block position in the plaintext, but behaves as "randomly as possible" subject to this constraint. The corresponding game, described in Figure 3.9, uses a random permutation family $\Pi_1 : \mathsf{FDE.SS} \times \{1, .., \mathsf{FDE.nb}\} \times \mathsf{FDE.BS} \to \mathsf{FDE.BS}$ simulated by lazy sampling that takes as input a sector number $s \in \mathsf{FDE.SS}$, a position number $i \in \{1, .., \mathsf{FDE.nb}\}$ and a message block $m \in \mathsf{FDE.BS}$ and returns a cipher block $c \in \mathsf{FDE.BS}$. For each couple (s, i), Π_1 defines

a random permutation such that $\Pi_1^-(s, i, (\Pi_1(s, i, m)) = m$. We specifically introduce this setting to describe the security goal of XTS, see Rogaway [Rog11] for a formal definition (using a filter function) of what is known to leak by the XTS mode.

```
procedure INITIALIZE
b ↞ {0,1}; K ↞ FDE.kg()
Π1 ← ⊥

procedure ENCRYPT(s, m)
if b = 0 then c ← FDE.enc(K, s, m)                    ▷ Real world
else                                                   ▷ Random world
   for i from 1 to nb
      if Π1(s, i, m[i]) = undef then
         Π1(s, i, m[i]) ↞ Rng Π1(s, i, .)
      c[i] ← Π1(s, i, m[i])
   c ← c[1]|| ... ||c[nb]
return c

procedure DECRYPT(s, c)
if b = 0 then m ← FDE.dec(K, s, c)                    ▷ Real world
else                                                   ▷ Random world
   for i from 1 to nb
      if Π1^-(s, i, c[i]) = undef then
         Π1^-(s, i, c[i]) ↞ Dom(Π1(s, i, .)
      m[i] ← Π1^-(s, i, c[i])
   m ← m[1]||..||m[nb]
return m

procedure FINALIZE(b')
return (b = b')
```

Figure 3.9: Game "up-to-block" for the IND-CCA-block security notion.

INDISTINGUISHABILITY UP-TO-PREFIX. In this definition, for $i \in \{1, \ldots, \mathsf{nb}\}$, the i-th ciphertext block depends deterministically on the sector number $s \in \mathsf{SS}$ and all previous plaintext blocks at position j for $j \in \{1, \ldots, i\}$, but again behave as "randomly as possible" subject to this constraint. The corresponding game is described in Figure 3.11 and it uses a random permutation family $\Pi_2 : \mathsf{FDE.SS} \times \mathsf{FDE.BS}^t \times \mathsf{FDE.MS} \to \mathsf{FDE.MS}$ lazily sampled that takes as input a sector number $s \in \mathsf{FDE.SS}$, a plaintext prefix $m' \in \mathsf{FDE.BS}^t$ such that $t \leq \mathsf{nb}$, a plaintext $m \in \mathsf{FDE.MS}$ and returns a ciphertext $c \in \mathsf{FDE.MS}$. For each couple (s, m'), Π_2 defines a random permutation such that $\Pi_2^-(s, m', \Pi_2(s, m', m)) = m$. . This notion corresponds to security notion described in [BBKN12] for *online ciphers*.

INDISTINGUISHABILITY UP-TO-REPETITION. In this definition, each ciphertext block depends deterministically on the plaintext block and the sector number s, but behaves as "randomly as possible" subject to this constraint. The corresponding game, described in Figure 3.10,

```
procedure INITIALIZE
b ↞{0,1}; K ← FDE.kg()

procedure ENCRYPT(s, m)
if b = 0                                   ▷ Real world
   c ← FDE.enc(K, s, m)
else                                       ▷ Random world
   if Π3(s, m) = undef then
      Π3(s, m) ↞ Rng(Π3(s, .))
   c ← Π3(s, m)
return c

procedure DECRYPT(s, c)
if b = 0                                   ▷ Real world
   m ← FDE.dec(K, s, c)
else                                       ▷ Random world
   if Π3^-(s, c) = undef then
      Π3^-(s, c) ↞ Dom(Π3(s, .))
   m ← Π3^-(s, c)
return m

procedure FINALIZE(b')
return (b = b')
```

Figure 3.10: Game "up-to-repetition" for the IND-CPA-repetition security notion.

uses a random permutation $\Pi_3 : \mathsf{FDE.SS} \times \mathsf{FDE.MS} \to \mathsf{FDE.MS}$ that takes as input a sector number $s \in \mathsf{SS}$ and a message $m \in \mathsf{MS}$ and returns a ciphertext $c \in \mathsf{MS}$. It is the best achievable notion for (length-preserving) deterministic encryption [BBO07].

Remark 3.1.2. If a construction is IND-CCA under one of these notions, it is also IND-CPA under the corresponding security notion.

Remark 3.1.3. If a construction is not IND-CPA under one of these notions, it is also not IND-CCA under the corresponding security notion.

As different ideal-world encryption oracles are used in the various security notions, it is trivially possible to distinguish between the encryption oracles. For example, for a fixed sector number and position on the plaintext, an IND-CPA up-to-block construction always returns the same ciphertext block. This construction does not reach IND-CPA up-to-prefix security, which requires indistinguishability up to the longest common prefix for a fixed sector number. It also does not satisfy IND-CPA up-to-repetition security, which requires indistinguishability up to repetition of the plaintext for a given sector number. Conversely, a construction that achieves IND-CPA up-to-prefix security will also not be IND-CPA up-to-block nor IND-CPA up-to-repetition using similar reasoning.

Analysis of Existing Constructions. We now analyse the FDE modes of operation described in subsection 3.1.1 with respect to these security notions. These results are

Table 3.1: The security of FDE modes of operation when no diversifier is used. Here, ✓ means that there is a security proof, and ✗ means that there is an attack. Proofs of the security results can be found in Sect. 3.1.2. XTS: see [IEE08]. TC1, TC2 and TC3 [RZ11] are generalizations of the HCBC1 [BBKN12], HCBC2 [BBKN12] and MHCBC [Nan08] constructions. WTBC: wide tweakable block cipher. The * symbol indicates that the property holds for some constructions, but not for others. Here, $x \geq \log_2(\ell)$.

	ECB	CTR	CBC	CBC	IGE	XTS	TC1	TC2/3	WTBC
IV →	n/a	$s \ll x$	s	$E_{K'}(s)$	$E_{K'}(s)$	s	s	s	s
IND-CPA-block	✗	✗	✗	✗	✗	✓	✗	✗	✗
IND-CPA-prefix	✗	✗	✗	✗	✗	✗	✓	✓	✗
IND-CPA-repetition	✗	✗	✗	✗	✗	✗	✗	✗	✓
IND-CPA	✗	✗	✗	✗	✗	✗	✗	✗	✗
IND-CCA-block	✗	✗	✗	✗	✗	✓	✗	✗	✗
IND-CCA-prefix	✗	✗	✗	✗	✗	✗	✗	✓	✗
IND-CCA-repetition	✗	✗	✗	✗	✗	✗	✗	✗	✓
IND-CCA	✗	✗	✗	✗	✗	✗	✗	✗	✗
online enc./dec.	✓	✓	✓	✓	✓	✓	✓	✓	✗
parallelizable enc.	✓	✓	✗	✗	✗	✓	✗	✗	✓*
parallelizable dec.	✓	✓	✓	✓	✗	✓	✓	✗	✓*

summarized in Table 3.1. The properties shown in the three last lines of Table 3.1 are relevant implementation properties but are not taken into account in the security proofs.

ECB mode. Unsurprisingly, ECB is not IND-CPA for any of our three security notions.

CTR mode. CTR is not IND-CPA for any of our three security notions. An adversary can simply query the encryption of $m[1] = 0^{\mathsf{n}}||m[2]||..||m[\mathsf{nb}]$ and $m[2] = 1^{\mathsf{n}}||m[2]||..||m[\mathsf{nb}]$ for the same sector number s (where $m[2], ..., m[\mathsf{nb}]$ can be any n-bit blocks). The first blocks of the obtained ciphertexts $c[1]$ and $c[2]$ will always satisfy $c_1[1] = c_2[1] \oplus 1^{\mathsf{n}}$, whereas this property holds only with probability $2^{-\mathsf{n}}$ in all three random worlds.

CBC mode. The attack on the CTR mode also applies to the variant of the CBC mode where the sector number is used as an IV. In the context of FDE, this attack is known as a Saarinen's watermarking attack [Saa04]. Bellare et al. [BBKN12] describe an attack on the CBC-based online cipher that shows that CBC-ESSIV is not IND-CPA up-to-prefix. This attack can be adapted to construct an adversary $\mathcal{A}$ for our three security notions:

1. $\mathcal{A}$ arbitrarily chooses $(\mathsf{nb} - 1)$ blocks $m[2], .., m[\mathsf{nb}]$ and a sector number s. $\mathcal{A}$ builds two plaintexts $m_1 = 0^{\mathsf{n}}||m[2]|| \ldots ||m[\mathsf{nb}]$ and $m_2 = 1^{\mathsf{n}}||m[2]|| \ldots ||m[\mathsf{nb}]$ that differ only in their first block.

2. $\mathcal{A}$ queries the ENCRYPT procedure to obtain the encryption of m_1 and m_2 on sector number s: $c_b[1]||c_b[2]|| \ldots ||c_b[\mathsf{nb}] \leftarrow \textsc{Encrypt}(s, m_b)$ for $b \in \{1, 2\}$.

3. $\mathcal{A}$ builds $m_3 = 1^{\mathsf{n}}||m_3[2]||m[3]|| \ldots ||m[\mathsf{nb}]$ where $m_3[2] = m[2] \oplus c_1[1] \oplus c_2[1]$ and queries the ENCRYPT procedure to obtain its encryption: $c_3[1]||...||c_3[\mathsf{nb}] \leftarrow \textsc{Encrypt}(s, m_3)$

4. $\mathcal{A}$ returns 0 to the FINALIZE procedure if $c_3[2] = c_1[2]$ and 1 otherwise.

The equality $c_3[2] = c_1[2]$ is always satisfied in the real world but holds only with probability 2^{-n} in all three random worlds.

IGE-ESSIV mode. The previous attack on the CBC-essiv mode can easily be adapted to show that IGE-essiv is not IND-CPA for any of our three security notions. The attack is identical except that the adversary $\mathcal{A}$ checks whether the equality $c_3[2] = c_1[2] \oplus 1^n$ holds or not.

XTS mode. As explained above, in XTS every plaintext block is encrypted separately using a tweakable block cipher, where the tweak is derived from the sector number and index of the block within the sector. As argued by Rogaway [Rog11], XTS is IND-CPA up-to-block secure. For syntactic reasons, it is not IND-CPA up-to-prefix nor IND-CPA up-to-repetition.

TC1, TC2, TC3 modes. Rogaway and Zhang [RZ11] showed that TC1 is IND-CPA up-to-prefix secure but not IND-CCA up-to-prefix. They also proved that TC2 and TC3 are IND-CCA up-to-prefix (and thus also IND-CPA up-to-prefix) secure. For syntactic reasons, they are not IND-CPA up-to-block nor IND-CPA up-to-repetition.

WTBC modes. Halevi and Rogaway showed that EME is IND-CCA up-to-repetition (and thus IND-CPA up-to-repetition) secure in [HR04]. For syntactic reasons, these modes are not IND-CPA up-to-block nor IND-CPA up-to-prefix.

```
procedure INITIALIZE
b ⇜{0,1}; K ← FDE.kg() ; t ← 0

procedure ENCRYPT(s, m)
t ← t + 1
if b = 0 then                                          ▷ Real world
   c ← FDE.enc(K, s, m)
else                                                   ▷ Random world
   m_{s,t} ← m
   (index, p) ← findLCP(m_{s,t}, 'm')
   for i from 1 to p
      c[i]_{s,t} ← c[i]_{s,index}
   for i from p + 1 to k
      Π2(s, m[1]..m[i-1], m[i]_{s,t}) ⇜ Rng Π2(s, m[1]..m[i-1], .)
      c[i]_{s,t} ← Π2(s, m[1]m[2]...m[i-1], m[i]_{s,t})
return c_{s,t}

procedure DECRYPT(s, c)
t ← t + 1
if b = 0                                               ▷ Real world
   m ← FDE.dec(K, s, c)
else                                                   ▷ Random world
   (index, p) ← findLCP(c_{s,t}, 'c')
   for i from 1 to p
      m[i]_{s,t} ← m[i]_{s,index}
   for i from p + 1 to k
      Π2^-(s, m[1]..m[i-1], c[i]_{s,t}) ⇜ Dom Π2(s, m[1]..m[i-1], .)
      m[i]_{s,t} ← Π2^-(s, m[1]m[2]...m[i-1], c[i]_{s,t})
return m_{s,t}

findLCP(d_{s,t}, 'd') for d ∈ {m, c}
p ← 0; pref ← 0; index ← 0
for j from 1 to t − 1
   for i from 1 to nb
      while d[i]_{s,j} = d[i]_{s,t}
         p ← i
if p > pref
   pref ← p; index ← j
return (index, pref)

procedure FINALIZE(b')
return (b = b')
```

Figure 3.11: Game "up-to-prefix" for the IND-CCA-prefix security notion.

3.2 FDE Security with Unique First Block

Because encryption in the context of FDE is deterministic and length-preserving, encrypting the same plaintext twice will always result in an identical ciphertext. The OS or application may, therefore, want to use a particular encoding of the plaintext, in order to ensure that the ciphertext will not be repeated. This corresponds to Bellare and Rogaway's Encode-Then-Encipher approach [BR00] to ensure strong privacy. One thus has to determine which encoding is sufficient to ensure security against CPA. In the context of security "up-to-block," this would require a large overhead, since encoding is then required for every block of a sector (typically 128 bits). However, for schemes that are IND-CPA "up-to-repetition" or "up-to-prefix," it is sufficient to ensure that the beginning of every message is unique. This can be done by prepending either random data or a counter, as suggested by Bellare and Rogaway [BR00].

In this section, we consider variants of the previous security notions with *unique first block* (ufb) and we prove IND-CPA security for CBC-ESSIV and IGE-ESSIV under this assumption. An application that is aware of this restriction, can therefore format its input such that the first block of every sector is unique. In subsection 3.3.2, we give a concrete example of such an application.

Relation to Previous Security Notions. The only difference between ufb model and the previous model is that for a given sector number s, $\mathcal{A}$ cannot make two queries to encrypt plaintexts that have same first block. So if a construction is secure under a security notion described in subsection 3.1.2, it is still the case in this model. Furthermore, the IND-CPA up-to-prefix, IND-CPA up-to-repetition and IND-CPA security notions become equivalent. This is easy to see: if the first block of plaintext is not repeated, then this is sufficient to ensure that the plaintext prefix or the entire plaintext is not repeated either.

Security Results. In this paragraph, we analyse the FDE modes of operation described in subsection 3.1.1 with respect to these security notions for unique first block. These results are summarized in Table 3.2.

ECB and CTR. The attacks described in Section 3.1.2 do not make queries with the same first block and the same sector number, and therefore still apply.

CBC-ESSIV. For this mode, in the attack of Section 3.1.2, $\mathcal{A}$ makes forbidden queries with the same first block. Theorem 3.2.1 states that CBC-ESSIV achieves IND-CPA-ufb security if the underlying block cipher E is a Pseudo-Random Function (PRF) [BR05]. We used the code-based game playing framework [BR06] for the security proof in subsection 3.2.1.

IGE-ESSIV. Theorem 3.2.2 states that the mode IGE-ESSIV achieves IND-CPA-ufb security and the security proof is given in subsection 3.2.2.

The TC1, TC2, TC3 and WTBC constructions become IND-CPA with the ufb restriction because TC1/2/3 were IND-CPA up-to-prefix and WTBC constructions were IND-CPA up-to-repetition as explained in Section 3.1.2.

3.2.1 CBC-ESSIV Security

Theorem 3.2.1. ***TheIND-CPA-ufb Security of CBC-ESSIV***
Let $E : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ *be a block cipher. Let* $\mathcal{A}$ *be an IND-CPA-ufb adversary*

IV →	ECB n/a	CTR $s \ll x$	CBC s	CBC $E_{K'}(s)$	IGE $E_{K'}(s)$	XTS s	TC1 s	TC2/3 s	WTBC s
IND-CPA-block	✗	✗	✗	✗	✗	✓	✗	✗	✗
IND-CPA-prefix	✗	✗	✗	✓	✓	✗	✓	✓	✓
IND-CPA-repetition	✗	✗	✗	✓	✓	✗	✓	✓	✓
IND-CPA	✗	✗	✗	✓	✓	✗	✓	✓	✓
IND-CCA-block	✗	✗	✗	✗	✗	✓	✗	✗	✗
IND-CCA-prefix	✗	✗	✗	✗	✗	✗	✗	✓	✓
IND-CCA-repetition	✗	✗	✗	✗	✗	✗	✗	✗	✓
IND-CCA	✗	✗	✗	✗	✗	✗	✗	✗	✓
online enc./dec.	✓	✓	✓	✓	✓	✓	✓	✓	✗
parallelizable enc.	✓	✓	✗	✗	✗	✓	✗	✗	✓⋆
parallelizable dec.	✓	✓	✓	✓	✗	✓	✓	✗	✓⋆

Table 3.2: The security of FDE modes of operation when no diversifier is used, but the first plaintext block unique for any given sector. Here, ✓ means that there is a security proof, and ✗ means that there is an attack.

against the FDE scheme obtained from the CBC-ESSIV mode on E such that $\mathcal{A}$ runs in time t and makes at most q queries to the ENCRYPT *procedure. There exists an adversary $\mathcal{B}$ (attacking the PRF security of E) such that:*

$$\mathbf{Adv}^{\mathsf{ind\text{-}CPA\text{-}ufb}}_{cbc-ESSIV}(\mathcal{A}) \leq 8 \cdot \mathbf{Adv}^{\mathsf{prf}}_{E}(\mathcal{B}) + \frac{q^2(\mathsf{nb}+1)^2}{2^{\mathsf{n}-1}}$$

where $\mathcal{B}$ runs in time at most $t' = t + O(q + \mathsf{n}q(\mathsf{nb}+1))$ and makes at most $q' = q(\mathsf{nb}+1)$ queries to its oracle.

Proof. Let $\mathcal{G}_0$ be the real-or-random security game as defined in Figure 3.11. In the following, we consider a sequence of modified games $\mathcal{G}_1, \ldots, \mathcal{G}_4$. We wish to upper bound the advantage of an adversary $\mathcal{A}$, which by definition is

$$\mathbf{Adv}^{\mathsf{ind\text{-}CPA\text{-}ufb}}_{cbc-ESSIV}(\mathcal{A}) = 2 \cdot Pr[\mathcal{G}_0^{\mathcal{A}} \Rightarrow \text{true}] - 1.$$

- **Game** $\mathcal{G}_1$ is identical to game $\mathcal{G}_0$ except that instead of the keyed block cipher encryption $E(K', \cdot)$ with a key K' randomly chosen in the FDE scheme obtained from the CBC-ESSIV, we use a random function F' instead. As in the previous security notions, the function F is generated *via* the lazy sampling method. The game $\mathcal{G}_1$ is given in detail in Figure 3.12.

  We consider an adversary $\mathcal{B}_1$ attacking the PRF security of E which runs $\mathcal{A}$ and simulates these two games by replacing the evaluation of E or F' by its own oracle (so that if $\mathcal{B}_1$ is in the real world, $\mathcal{A}$ is in the game $\mathcal{G}_0$, but if $\mathcal{B}_1$ is in the random world, $\mathcal{A}$ is in the game $\mathcal{G}_1$). If $\mathcal{A}$ makes q queries to the ENCRYPT procedure then $\mathcal{B}_1$ makes $q' = q(\mathsf{nb}+1)$ queries to its own oracle and we have:

$$Pr[\mathcal{G}_0^{\mathcal{A}} \Rightarrow 1] - Pr[\mathcal{G}_1^{\mathcal{A}} \Rightarrow 1] \leq \mathbf{Adv}^{\mathsf{prf}}_{E}(\mathcal{B}_1).$$

Game $\mathcal{G}_1$	Game $\mathcal{G}_2$
procedure ENCRYPT(s, m)	procedure ENCRYPT(s, m)
if $b = 0$ **then**	**if** $b = 0$ **then**
$c[0] \twoheadleftarrow \{0,1\}^{\mathsf{n}}$	$c[0] \twoheadleftarrow \{0,1\}^{\mathsf{n}}$
if $s \in \mathsf{Dom}(\mathsf{F}')$ **then** $c[0] \leftarrow \mathsf{F}'(s)$	**if** $s \in \mathsf{Dom}(\mathsf{F}')$ **then** $c[0] \leftarrow \mathsf{F}'(s)$
else $\mathsf{F}'(s) \leftarrow c[0]$	**else** $\mathsf{F}'(s) \leftarrow c[0]$
for i **from** 1 **to** nb	**for** i **from** 1 **to** nb
$X \leftarrow c[i-1] \oplus m[i]$	$c[i] \twoheadleftarrow \{0,1\}^{\mathsf{n}}$
$c[i] \leftarrow E(K, X)$	$X \leftarrow c[i-1] \oplus m[i]$
$c \leftarrow c[1]\|\|c[2]\|\| \ldots \|\|c[\mathsf{nb}]$	**if** $X \in \mathsf{Dom}(\mathsf{F})$ **then** $c[i] \leftarrow \mathsf{F}(X)$
else $c \twoheadleftarrow \{0,1\}^{\mathsf{nb}\cdot\mathsf{n}}$	$\mathsf{F}(X) \leftarrow c[i]$
return c	$c \leftarrow c[1]\|\|c[2]\|\| \ldots \|\|c[\mathsf{nb}]$
	else $c \twoheadleftarrow \{0,1\}^{\mathsf{nb}\cdot\mathsf{n}}$
	return c

Figure 3.12: Game $\mathcal{G}_1$ and game $\mathcal{G}_2$ for CBC-ESSIV IND-CPA-ufb

- **Game** $\mathcal{G}_2$ is identical to game $\mathcal{G}_1$ except that instead of the keyed block cipher encryption $E(K, \cdot)$ with a key K randomly chosen in the FDE scheme obtained from the CBC-ESSIV, we use a random function F instead. As in the previous security notions, the function F is generated *via* the lazy sampling method. The game $\mathcal{G}_2$ is given in detail in Figure 3.12.

  Again we consider an adversary $\mathcal{B}_2$ attacking the PRF security of E which runs $\mathcal{A}$ and simulates these two games by replacing the evaluation of E or F by its own oracle (so that if $\mathcal{B}_2$ is in the real world, $\mathcal{A}$ is in the game $\mathcal{G}_0$, but if $\mathcal{B}_2$ is in the random world, $\mathcal{A}$ is in the game $\mathcal{G}_1$). If $\mathcal{A}$ makes q queries to the ENCRYPT procedure than $\mathcal{B}_2$ makes $q' = q(\mathsf{nb} + 1)$ queries to its own oracle and we have:

$$Pr[\mathcal{G}_1^{\mathcal{A}} \Rightarrow 1] - Pr[\mathcal{G}_2^{\mathcal{A}} \Rightarrow 1] \leq \mathbf{Adv}_E^{\mathsf{prf}}(\mathcal{B}_2).$$

- **Game** $\mathcal{G}_3$ (see Figure 3.13) is identical to game $\mathcal{G}_2$ except that we add Boolean flags in the pseudo-code for the remainder of the proof (see Figure 3.13). We have

$$Pr[\mathcal{G}_2^{\mathcal{A}} \Rightarrow 1] = Pr[\mathcal{G}_3^{\mathcal{A}} \Rightarrow 1].$$

- **Game** $\mathcal{G}_4$: The games $\mathcal{G}_3$ and $\mathcal{G}_4$ (see Figure 3.13) are identical except if the boolean flag bad_2 is set to true. We thus have

$$Pr[\mathcal{G}_3^{\mathcal{A}} \Rightarrow 1] - Pr[\mathcal{G}_4^{\mathcal{A}} \Rightarrow 1] \leq Pr[\mathcal{G}_3^{\mathcal{A}} \text{ sets } \mathsf{bad}_2].$$

- **Game** $\mathcal{G}_5$: In game $\mathcal{G}_5$ (see Figure 3.13), the generated ciphertexts in the ENCRYPT procedure are independent of the bit b and we have $Pr[\mathcal{G}_4^{\mathcal{A}} \Rightarrow 1] = 1/2$.

We first evaluate $Pr[\mathcal{G}_5^{\mathcal{A}} \text{ sets } \mathsf{bad}_1]$. For the i-th request to the ENCRYPT procedure, the probability to have a collision for the output of the F' is at most $\frac{(i-1)}{2^{\mathsf{n}}}$ then for all the q

Game $\mathcal{G}_3$ | Game $\mathcal{G}_4$ | Game $\mathcal{G}_5$

procedure ENCRYPT(s, m)
if $b = 0$ **then**
  $c[0] \twoheadleftarrow \{0,1\}^{\mathsf{n}}$
  if $s \in \mathsf{Dom}(\mathsf{F}')$ **then** $c[0] \leftarrow \mathsf{F}'(s)$
  else
    if $c[0] \in \mathsf{Rng}(\mathsf{F}')$ **then** $\mathsf{bad}_1 \leftarrow true$
    $\mathsf{F}'(s) \leftarrow c[0]$
  for i **from** 1 **to** nb
    $X \leftarrow c[i-1] \oplus m[i]$
    $X \twoheadleftarrow \{0,1\}^{\mathsf{n}}$
    $c[i] \twoheadleftarrow \{0,1\}^{\mathsf{n}}$
    if $X \in \mathsf{Dom}(\mathsf{F})$ **then** $c[i] \leftarrow \mathsf{F}(X)$; $\mathsf{bad}_2 \leftarrow true$
    $\mathsf{F}(X) \leftarrow c[i]$
  $c \leftarrow c[1]||c[2]||\ldots||c[\mathsf{nb}]$
else $c \twoheadleftarrow \{0,1\}^{\mathsf{nb}\cdot\mathsf{n}}$
return c

Figure 3.13: Games $\mathcal{G}_3$, $\mathcal{G}_4$ and $\mathcal{G}_5$ CBC-ESSIV IND-CPA-ufb. The framed statement is included in game $\mathcal{G}_3$ only. The statement overlined in gray is included in game $\mathcal{G}_5$ only.

queries:

$$Pr[\mathcal{G}_5^{\mathcal{A}} \text{ sets } \mathsf{bad}_1] \leq \sum_{i=1}^{q} \frac{(i-1)}{2^{\mathsf{n}}} = \frac{q(q-1)}{2^{\mathsf{n}+1}}.$$

If the Boolean flag bad_1 is not set to true, then in the security game, the adversary $\mathcal{A}$ never queried the ENCRYPT procedure with two different sector numbers s and s' such that $\mathsf{F}(s) = \mathsf{F}(s')$. In particular, for each query, the value $c[0]$ is random and not revealed to $\mathcal{A}$ and by the unique-first-block assumption, it is never used twice with the same plaintext block $m[1]$. We thus have:

$$\begin{aligned} &Pr[\mathcal{G}_4^{\mathcal{A}} \text{ sets } \mathsf{bad}_2 \text{ but not } \mathsf{bad}_1] \\ &= Pr[\mathcal{G}_5^{\mathcal{A}} \text{ sets } \mathsf{bad}_2 \text{ but not } \mathsf{bad}_1] \end{aligned}$$

Finally, in game $\mathcal{G}_5$, we have

$$Pr[\mathcal{G}_5^{\mathcal{A}} \text{ sets } \mathsf{bad}_2] \leq \sum_{i=1}^{(\mathsf{nb}+1)q-1} \frac{i}{2^{\mathsf{n}}} \leq \frac{q^2(\mathsf{nb}+1)^2}{2^{\mathsf{n}+1}}.$$

Summing up, we obtain that $\mathbf{Adv}^{\mathsf{ind\text{-}CPA\text{-}ufb}}_{cbc-essiv}(\mathcal{A})$ is upper-bounded by

$$2 \cdot \mathbf{Adv}^{\mathsf{prf}}_{E}(\mathcal{B}') + 2 \cdot \mathbf{Adv}^{\mathsf{prf}}_{E}(\mathcal{B}'') + \frac{q^2(\mathsf{nb}+1)^2}{2^{\mathsf{n}}} + \frac{q(q-1)}{2^{\mathsf{n}}}.$$

Considering an adversary $\mathcal{B}$ attacking the PRF security of E that simply runs $\mathcal{B}_1$ with

probability 1/2 and $\mathcal{B}_2$ with probability 1/2, we have

$$\mathbf{Adv}^{\mathsf{prf}}_E(\mathcal{B}) \geq \frac{1}{2}\mathbf{Adv}^{\mathsf{prf}}_E(\mathcal{B}_1) \text{ and } \mathbf{Adv}^{\mathsf{prf}}_E(\mathcal{B}) \geq \frac{1}{2}\mathbf{Adv}^{\mathsf{prf}}_E(\mathcal{B}_2)$$

and we obtain the claimed bound.

□

Adversary. The following attack shows that CBC-ESSIV does not achieve IND-CCA-ufb up-to-prefix security:

1. $\mathcal{A}$ chooses a plaintext $m_1 = m_1[1]||m_1[2]||\ldots||m_1[\mathsf{nb}] \in \{0,1\}^{\mathsf{nb}\cdot\mathsf{n}}$ and a sector number s and queries the Encrypt procedure with (s, m_1) to obtain $c_1[1]||c_1[2]||\ldots||c_1[\mathsf{nb}]$.

2. $\mathcal{A}$ builds a ciphertext $c_2 = c_2[1]||c_2[2]||\ldots||c_2[\mathsf{nb}]$ with $c_2[1] = c_1[2]$, $c_2[2] = c_1[2]$ and arbitrary $c_2[i] \in \{0,1\}^{\mathsf{n}}$ for $i \in \{3,\ldots,\mathsf{nb}\}$ and query the Decrypt procedure with (s, c_2) to obtain a plaintext $m_2 = m_2[1]||m_2[2]||\ldots||m_2[\mathsf{nb}]$.

3. $\mathcal{A}$ outputs 0 if $m_2[2] = m_1[3]$ and 1 otherwise.

The equality $m_2[2] = m_1[3]$ is always satisfied in the real world but this property holds only with probability $2^{-\mathsf{n}}$ in the random world.

3.2.2 IGE-ESSIV Security

The following theorem states that the mode IGE-ESSIV achieves IND-CPA-ufb security:

Theorem 3.2.2. [The IND-CPA-ufb Security of IGE-ESSIV]
Let $E : \{0,1\}^{\mathsf{n}} \times \{0,1\}^{\mathsf{n}} \to \{0,1\}^{\mathsf{n}}$ be a block cipher. Let $\mathcal{A}$ be an IND-CPA-ufb adversary against the FDE scheme obtained from the IGE-ESSIV mode on E such that $\mathcal{A}$ runs in time t and makes at most q queries to the Encrypt *procedure. There exists an adversary $\mathcal{B}$ (attacking the PRF security of E) such that:*

$$\mathbf{Adv}^{\mathsf{ind\text{-}CPA\text{-}ufb}}_{ige-ESSIV}(\mathcal{A}) \leq 8 \cdot \mathbf{Adv}^{\mathsf{prf}}_E(\mathcal{B}) + \frac{q^2(\mathsf{nb}+1)^2}{2^{\mathsf{n}-1}}$$

where $\mathcal{B}$ runs in time at most $t' = t + O(q + \mathsf{n}q(\mathsf{nb}+1))$ and makes at most $q' = q(\mathsf{nb}+1)$ queries to its oracle.

Proof. This proof is similar to CBC-ESSIV proof. Let $\mathcal{G}_0$ be the real-or-random security game as defined in Figure 3.11. In the following, we consider a sequence of modified games $\mathcal{G}_1, \ldots, \mathcal{G}_5$. We wish to upper bound the advantage of an adversary $\mathcal{A}$ which by definition is

$$\mathbf{Adv}^{\mathsf{ind\text{-}CPA\text{-}ufb}}_{ige-ESSIV}(\mathcal{A}) = 2 \cdot Pr[\mathcal{G}_0^{\mathcal{A}} \Rightarrow \mathsf{true}] - 1.$$

- $\mathcal{G}_1$ - $\mathcal{G}_3$**:** (see Figure 3.14) The only point that changes between CBC-ESSIV games $\mathcal{G}_0$ to $\mathcal{G}_4$ is that the ciphertext block is not the output of the PRF Π but the output XORed with the previous plaintext block.

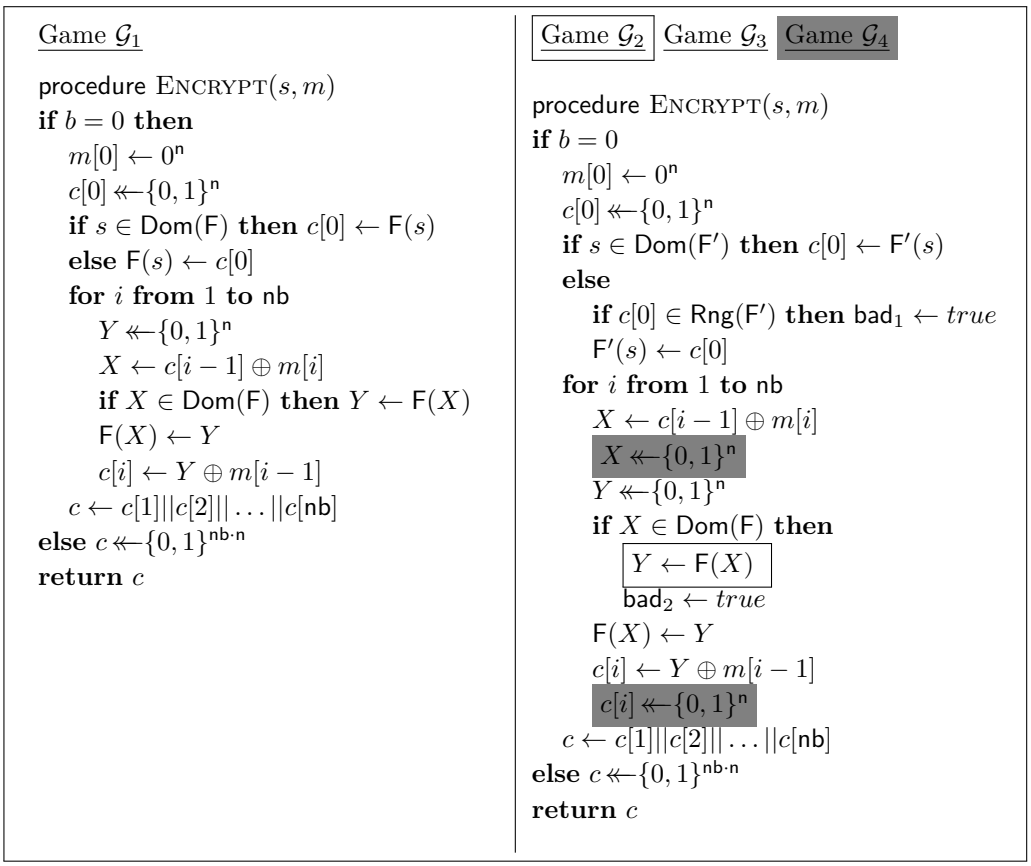

Figure 3.14: Games $\mathcal{G}_1$ to $\mathcal{G}_4$ for IGE-ESSIV IND-CPA-ufb

- $\mathcal{G}_4$**:** (see Figure 3.14) We now consider IGE-ESSIV's $\mathcal{G}_4$. In $\mathcal{G}_4$, the ciphertext blocks $c[i]$ are the result of an XOR between the previous plaintext block and the PRF F output that is why the ciphertext blocks $c[i]$ are randomly chosen in $\{0,1\}^{\mathsf{n}}$.

□

ADVERSARY. The following attack (inspired by Rohatgi [Jut00]) shows that IGE-ESSIV does not achieve IND-CCA-ufb up-to-prefix security:

1. $\mathcal{A}$ chooses a plaintext $m_1 = m_1[1]||m_1[2]||\ldots||m_1[\mathsf{nb}] \in \{0,1\}^{\mathsf{nb}\cdot\mathsf{n}}$ and a sector number s and queries the ENCRYPT procedure with (s, m_1) to obtain $c_1[1]||c_1[2]||..||c_1[\mathsf{nb}]$.

2. $\mathcal{A}$ builds a ciphertext $c_2 = c_2[1]\ldots c_2[2]||\ldots||c_2[\mathsf{nb}]$ with $c_2[1] = c_1[1]$, $c_2[2] = m_1[2] \oplus c_1[3] \oplus m_1[1]$ and arbitrary $c_2[i] \in \{0,1\}^{\mathsf{n}}$ for $i \in \{3,\ldots,\mathsf{nb}\}$ and query the DECRYPT procedure with (s, c_2) to obtain a plaintext $m[]_2 = m_2[1]||m_2[2]||\ldots||m_2[\mathsf{nb}]$.

3. $\mathcal{A}$ outputs 0 if $m_2[2] = m_1[3] \oplus c_1[2] \oplus c_1[1]$ and 1 otherwise.

The equality $m_2[2] = m_1[3] \oplus c_1[2] \oplus c_1[1]$ is always satisfied in the real world, but this property holds only with probability $2^{-\mathsf{n}}$ in the random world.

3.3 FDE Security with a Diversifier

Typically, IND-CPA cannot be reached for FDE, as the deterministic nature of FDE means that identical plaintexts will result in identical ciphertexts. We worked around this problem in the previous section by imposing a restriction on the plaintext: the first plaintext block must be unique. Now, we introduce another way to achieve IND-CPA, without restricting the plaintext, but still without storing additional data. Instead, we will use a diversifier j, which will be associated to every sector. To be able to stay within the constraints of FDE, it should somehow be possible to assign a diversifier to every sector without using additional storage. Possible candidates in the particular case of SSDs will be considered in subsection 3.3.1. For now, it is enough to consider that for each encryption, a diversifier is picked among $\{0,1\}^d$, in such a way this diversifier is never repeated for a particular sector. Then two identical plaintexts with the same sector number will have different ciphertexts, a property that could previously not be achieved within the context of FDE. The combination of the sector number s and the diversifier j is used instead of the sector number in FDE constructions. The combination proposed is simply the concatenation between these two values $s||j$ such as $s \in \{0,1\}^\sigma$, $j \in \{0,1\}^d$ and $\mathsf{n} = d + \sigma$.

For the analysis in this section, it suffices that the diversifier is never repeated for a particular sector. As such, the security analysis is the same as if the diversifier were a nonce. However, we will explain in subsection 3.3.1 that efficiency reasons require that at any particular point in time, all diversifier values should occur roughly the same number of times and that the diversifier must be a rather short value, typically only a few bits.

Security Results. IND-CPA up-to-repetition becomes equivalent to IND-CPA security: the only difference between these notions is that if $\mathcal{A}$ asks to encrypt twice the same query (s, m) the answer will be the same ciphertext, but these queries are not allowed any more under the diversifier model. Moreover, in IND-CCA game, the adversary is not allowed to query the decryption of a ciphertext what was previously encrypted, or vice versa. It can, therefore, be seen that IND-CCA up-to-repetition becomes equivalent to IND-CCA. Similarly, since the adversary $\mathcal{A}$ is not allowed to encrypt twice with the same pair $s||j$, the IND-CPA up-to-block property is also equivalent to the other IND-CPA security notions.

Table 3.3 summarizes the security properties achieved by the FDE modes of operation when used with a diversifier. The IND-CPA attacks of Theorem 3.1.2 still carry over to ECB and CBC with a sector-number IV. However CTR mode becomes secure as the counter value is not repeated [BDJR97]. The IND-CPA security of XTS, TC1, TC2, TC3 and WTBC follows from the fact that the tweak is not reused. For CBC-essiv and IGE-essiv, the IND-CPA security follows from the proof of Sect. 3.2: now the first block may be reused, but the IV is unique.

Let us explain the attacks under IND-CCA in Table 3.3:

- This following attack shows that CTR is not IND-CCA-xx: $\mathcal{A}$ encrypts $(s||j, m)$ with m any plaintext and any $s||j$ and receives c then $\mathcal{A}$ decrypts $(s||j, c')$ where $c' = c \oplus 0^{\mathsf{n}-1}1$. $\mathcal{A}$. Then, $m'[1] = m[1] \oplus 0^{\mathsf{n}-1}1$ is always satisfied in the real world but holds only with probability $2^{-\mathsf{n}}$ in the ideal world.

- CBC-ESSIV and IGE-ESSIV are not secure: the attacks of section 3.2 still apply, as they did not perform two encryptions with the same sector number.

IV →	ECB n/a	CTR $s\|\|j \ll x$	CBC $s\|\|j$	CBC $E_{K'}(s\|\|j)$	IGE $E_{K'}(s\|\|j)$	XTS $s\|\|j$	TC1 $s\|\|j$	TC2/3 $s\|\|j$	WTBC $s\|\|j$
IND-CPA-block	✗	✓	✗	✓	✓	✓	✓	✓	✓
IND-CPA-prefix	✗	✓	✗	✓	✓	✓	✓	✓	✓
IND-CPA-repetition	✗	✓	✗	✓	✓	✓	✓	✓	✓
IND-CPA	✗	✓	✗	✓	✓	✓	✓	✓	✓
IND-CCA-block	✗	✗	✗	✗	✗	✓	✗	✗	✗
IND-CCA-prefix	✗	✗	✗	✗	✗	✗	✗	✓	✗
IND-CCA-repetition	✗	✗	✗	✗	✗	✗	✗	✗	✓
IND-CCA	✗	✗	✗	✗	✗	✗	✗	✗	✓
online enc./dec.	✓	✓	✓	✓	✓	✓	✓	✓	✗
parallelizable enc.	✓	✗	✓	✗	✗	✓	✗	✗	✓⋆
parallelizable dec.	✓	✓	✓	✓	✗	✓	✓	✗	✓⋆

Table 3.3: The security of FDE modes of operation when a diversifier is used. Here, ✓ means that there is a security proof, and ✗ means that there is an attack.

- For syntactical reasons, an encryption scheme can only be IND-CCA up-to-block, IND-CCA up-to-prefix, or IND-CCA up-to-repetition. XTS is only IND-CCA up-to-block, TC2 and TC3 are only IND-CCA up-to-prefix, and WBTC are only IND-CCA up-to-repetition.

As shown in Table 3.3, the diversifier shows how to reach IND-CPA security for most commonly-used FDE encryption modes. It also succeeds in providing IND-CCA security for WTBC constructions, which not achievable in a "classical" FDE model.

3.3.1 Solid State Drive

We now recall the basics of SSD storage, so that we can explain how to modify only the firmware of SSDs to associate a *diversifier* to every sector. This diversifier allows us to encrypt the same plaintext in distinct ways for the same sector number.

SSDs are flash memory devices that are gradually replacing the magnetic Hard Disk Drive (HDD) due to their reliability and performance. Just like HDDs, they are sector-addressable devices. They are indexed by a sector number that is also known as a Logical Block Address (LBA). This ensures that the physical details of the storage device are not exposed to the OS, but are managed by the firmware of the storage device. For SSDs, the Flash Translation Layer (FTL) stores the mapping between LBAs and Physical Block Addresses (PBAs). This FTL is necessary to ensure an even distribution of writes to every sector number (*wear leveling*) and to invalidate blocks so that they can later be recycled (*garbage collection*). This FTL is necessary due to the physical constraints of flash storage: any physical address can only be written a limited number of times, and rewriting individual sectors is not possible: invalidated sectors can only be recovered in multiples of the *erase block size*.

SSD Components. The flash memory of an SSD is hierarchically organized as a set of

flash chips called packages, which are further divided in dies, planes, blocks[2] and pages. Every page consists of one or more sectors, and is the smallest unit that can be written. The smallest unit that can be erased is a block. Invalidated blocks must be erased before writing, and the number of erasures and writes to every block is limited for flash storage. To abstract away the notion of packages, dies and planes, the Open NAND Flash Interface (ONFI) standard introduces the notion of Logical Unit Number (LUN) as the minimum granularity of parallelism for flash storage. As operations can be issued to several planes in parallel, a LUN corresponds to a plane.

Introducing a Diversifier. In the context of SSD storage, we will show how to associate a *diversifier* to every LBA. This diversifier will not be stored, and will not modify the data structures of the SSD. Instead, the diversifier will impose an additional restriction on the FTL, meaning that the diversifier determines which PBAs can store the data corresponding to a particular LBA. The intuition is that if the diversifier is selected "randomly" for every write operation, the data will be spread out evenly over the SSD, and the SSD performance should not be affected too much. We will verify this by implementing and benchmarking our proposed solution in subsection 3.3.1. When a write command is issued, there are various ways to specify a diversifier value for a given LBA. We will prefer to transmit this information in one operation. As such, we do not only avoid the performance drawbacks of issuing several operations to write one sector, but we also do not need to worry about inconsistent states when operations are lost, modified, or reordered. In particular, we propose to send the diversifier along with the sector data as part of a "fat" sector that is already supported in SATA (Serial ATA) interface for storage devices. This diversifier value will be returned to the OS when a large sector read command is issued. In an attempt to make the diversifier as long as possible, we may want to consider the optimal (yet completely unrealistic) scenario: the diversifier uniquely specifies the physical page to which the data must be written. But, even then, the size of the diversfier is typically be very short: e.g., only 24 bits in case of an 128 GB SSD with 8 kB pages. We will, in fact, choose the diversifier to be much shorter, so that a practical implementation is possible that minimally modifies the SSD firmware. This diversifier cannot be selected at random, as it would be too short to avoid repetitions for a particular sector. However, it can also not be a counter, as all diversifier values should be used roughly an equal number of times over all sectors to spread the writes over the entire disk layout.

In our solution, we want to avoid that the SSD needs to send data back to the host for decryption and re-encryption under a different diversifier, as this would affect the SSD performance quite drastically. Therefore, wear leveling and garbage collection operations may not change the diversifier. A straightforward solution is then to make the diversifier correspond to the LUN (or a set of LUNs), which is what we will implement and benchmark in the following section. The OS can freely select the diversifier values; however they may not be repeated for any particular sector, and all diversifier values should be used roughly the same number of times. In subsection 3.3.2, we give a concrete example of an application where the OS implements such diversifier. More specifically, we show how the OS can ensure randomness and uniqueness even for very short diversifier values.

[2]These blocks should not be confused with the blocks of the block cipher, nor with the "block" (actually "sector") in the term Logical Block Address (LBA).

EagleTree Benchmarks. In order to confirm that the concept of a diversifier is not just feasible but also efficient to implement, we implemented this feature in the EagleTree SSD simulator [DSB+13] and performed various benchmarks.
Our modified EagleTree simulator is based on the latest commit of EagleTree on GitHub[3] of February 23, 2016, to which we made three small modifications. In particular:

- We added an additional `diversifier` data member to the `Event` class, which is set to a random value every time the operating system generates a write operation in `generate_io()`.
- The `get_free_block_pointer_with_shortest_IO_queue()` function in the class `Block_manager_parent` now has a `diversifier` argument added to it. It restricts I/O operations to the `package` that is specified by the value of `diversifier`.
- In the block manager `Block_manager_parallel`, the `choose_best_address()` and `choose_any_address()` functions were redefined. For write operations that are issued either by the operating system or by the garbage collector, the `package` will be restricted to the value of the diversifier. This value is passed on when `get_free_block_pointer_with_shortest_IO_queue()` is called within this function. No modifications are made for reads and writes to sectors that are not visible to the OS, such as for the mapping information I/O generated by the FTL.

The device that is simulated, consists of eight packages, each containing four dies of 256 blocks. Each block consists of 128 pages of 4096 bytes. EagleTree currently does not support multiple planes per die. The page read, page write, bus ctrl, bus data and block erase delays are 5 µs, 20 µs, 1 µs, 10 µs and 60 µs respectively. We assume that any latencies incurred by the OS (including sector encryption and decryption) are negligible with respect to these numbers. The SSD has an overprovisioning factor of 0.7.

We simulate an SSD configured with DFTL and the greedy garbage collection policy. The base benchmarks use a simple block scheduler that assigns the next write to whichever package is free. We then compare this performance to various choices of the diversifier value, which determines the package for every write.

The workload used in our benchmarks is the same as the example in EagleTree's `demo.cpp` file: first a large write is made to the entire logical address space. This write is performed in random order, but without writing to the same address twice. Once this large write is finished, two threads are started up: one performs random reads, and the other performs random writes in the address space. After three million I/O operations, the simulation is stopped.

The benchmark results are shown in Table 3.4. They suggest not to choose the number of diversifier values to be equal to the number of packages, as the impact on performance is quite significant. Compared to the benchmarks without diversifier, the throughput of the reads and writes drop by 25 % and 7 % respectively. The average latency increases by 44 % for reads, and 8 % for writes. In this setup, the reads and writes of each garbage collection operation are restricted to one package, and this affects performance quite significantly. Reads suffer more than writes; this is mainly because there is no significant drop in performance for the initial write operations to fill up the SSD.

[3] https://github.com/ClydeProjects/EagleTree

Table 3.4: EagleTree Benchmarks for various diversifier sizes.

	diversifier size (bits)			
	0	1	2	3
read latency (µs)	28.292	28.862	31.619	40.640
write latency (µs)	32.009	32.070	32.470	34.560
read throughput (IOPS)	20860	20493	19050	15634
write throughput (IOPS)	31240	31181	30797	28935
garbage collector reads	1284043	1295530	1356043	1489623
garbage collectorwrites	1284043	1295530	1356041	1489622
erasures	18569	18765	19661	21634

To avoid the large impact on performance, we must, therefore, choose the number of diversifier values to be smaller than the number of packages. When the diversifier is two bits, our simulations show an increase of the average latency of 12 % for reads and 1 % for writes and a reduction of throughput of 9 % for reads and 1 % for writes. The total throughput reduction (reads and writes combined) is at most 4 %. For a diversifier of one bit, latency and throughput are affected by less than 2 %. We also looked into the number of garbage collection and the number of erasures. They worsen by less than 6 % for a diversifier of two bits, but by 16 % for a diversifier of three bits.

3.3.2 Case Studies

We now present three case studies for FDE. They apply the theoretical framework developed in this paper, in order to obtain novel practical solutions that advance state of the art. For each of the case studies, we explain the advantages of our solutions over existing techniques. The examples are rather concrete for the sake of clarity, but the practical relevance of our results is not limited to these particular examples.

3.3.2.1 Online Ciphers for FDE

Case Description. We consider an FDE application for which both encryption, and decryption must be online. Off-line modes of operation cannot be implemented due to a practical restriction: the encryption and decryption must be performed by a hardware security module (HSM) that does not have enough memory to store an entire sector. With this consideration in mind, what FDE security notions can be achieved by this application?

Solution. As encryption and decryption must be online, we know that two plaintexts with a common prefix will result in ciphertexts that have a common prefix as well. In the "up-to-prefix" setting, Table 3.1 explains that neither the commonly-used CBC (e.g. in Microsoft's BitLocker [Fer06]) nor its "improved" IGE variant (see e.g. Fruhwirth [Fru05]) provides chosen-plaintext or chosen-ciphertext security. However, these security properties can be achieved by TC2 and TC3.

Advantages. We are unaware of any applications of TC2 and TC3 in FDE. However, when encryption and decryption must be online, these constructions provide up-to-prefix chosen-ciphertext security in a setting where the commonly-used CBC and the often-proposed IGE are insecure.

3.3.2.2 FDE-Aware Database Applications

Case Description. After intensive benchmarking, a database application was found to achieve an insufficiently high throughput when encryption is enabled. Without encryption, the performance of the database application is within acceptable limits. To avoid the security risks of storing data without encryption, a decision was made to use a hardware-based FDE based on CBC-ESSIV. Database applications are often already aware of the sector size to align read and write operations to sector boundaries,[4] and these operations already contain some wastage.[5] This is also the case for the application that we consider, which additionally also knows whether FDE is enabled for a particular disk volume. For the given application, at least eight bytes of padding are added to the end of every record in order to ensure sector alignment. Is it possible for this application to achieve indistinguishability under chosen plaintexts, which implies that identical sector plaintexts will result in distinct ciphertexts?

Solution. CBC-ESSIV is a common solution for FDE, but unfortunately, Table 3.1 explains that it does not achieve any of the IND-CPA notions that are defined in this paper. However, Table 3.2 shows that IND-CPA can be achieved for CBC-ESSIV if the first plaintext block for every sector is unique. The application can decide to reduce the padding at the end of every record by eight bytes, and instead prepend an eight-byte time stamp. Assuming that this time stamp will not repeat, the first plaintext block of every sector is unique, and indistinguishability under chosen plaintext attacks holds. Of course, the application should detect whether FDE using CBC-ESSIV is enabled for a particular disk, and return an error if this is not the case.

Advantages. To the best of our knowledge, existing solutions would encrypt at the application level, at the disk level using FDE, or would use a combination of both. The FDE is unaware of the application, as it performs encryption at the sector level. However what if the application is aware of the FDE? If that is the case, we explain how the application can generate plaintexts of a particular structure in order to obtain a stronger notion of FDE security, but without performing any encryption at the application level.

3.3.2.3 On-the-Fly Firmware Updates

Case Description. A medical monitoring system continuously measures the patient's vital signs. The device has an Internet connection, which it uses to transmit its measurements, as well as to install security-critical on-the-fly firmware updates. The firmware is stored in off-chip Flash memory. Hackers could easily read out previous versions of the firmware, as they were stored without encryption. To prevent such attacks, the company has decided

[4] https://blogs.msdn.microsoft.com/psssql/2011/01/13/sql-server-new-drives-use-4k-sector-size/
[5] https://docs.oracle.com/cd/E18283_01/server.112/e17120/onlineredo002.htm

to use FDE because there is no space to store additional data such as IVs, nonces or tags. However, the company identifies a problem with FDE: if hackers know which sectors in the Flash memory are modified by a security update, they can compare these sectors with older firmware. These hackers can then sniff around in the same part of the code to quickly rediscover the security vulnerability, and possibly exploit it before the devices in the field are updated. How can this problem be avoided?

Solution. It seems that the company wants to represent sectors with identical plaintexts by different ciphertexts, in order to hide which sectors are affected by a firmware update. Classical FDE does not allow this, as there is no space to store additional data such as an IV or nonce. However, Flash-based storage allows us to achieve security against chosen plaintexts, by modifying the Flash controller to introduce a diversifier. Depending on the security requirements against chosen ciphertexts, Table 3.3 proposes various solutions. Initially, the diversifier will be the most significant d bits of the hash of the sector number, where d is the size of the diversifier in bits. Every firmware update will have a version number, which will be XORed to the diversifier of every sector. Clearly, the number of firmware updates must be less than the number of diversifier values. Under this restriction, all diversifiers will be unique for a particular sector. Note that the diversifier cannot be a counter, as this would result in a significant degradation of performance. However, it can also not be chosen at random, as this will likely result in collisions due to the small size of the diversifier.

Advantages. An alternative solution may be to randomize all the instructions for every firmware update, but this technique is very error-prone and it will be difficult (if not impossible) to ensure that the device remains in a consistent state, in particular if power is lost during an on-the-fly firmware update. Our solution avoids this: for sectors that are unaffected by the update, the ciphertext changes but the plaintext remains the same. Changing the FDE key is not an option for an on-the-fly update: while the device is being updated, there is no non-volatile memory available to store the mapping between encryption keys and sector numbers. However, changing the FDE key is possible during scheduled maintenance, when updates do not need to be performed on the fly. The firmware update counter can then be reset. Therefore, as long as the number of on-the-fly updates before scheduled maintenance is less than d, the small size of the diversifier does not present a problem.

Conclusion. We presented a theoretical framework for disk encryption, and we analysed several existing constructions against chosen-plaintext and chosen-ciphertext attacks, under different notions of the ideal-world encryption oracle: up-to-repetition, up-to-prefix, or up-to-block. Using this model, we recalled that IGE-ESSIV does not have chosen-ciphertext-security under any of the notions that we consider, which shows that the AREA construction proposed by Fruhwirth [Fru05] is insecure. Nevertheless, we proved that IGE-ESSIV and even CBC-ESSIV could provide security under chosen-plaintext attacks, under the assumption that the first block of a plaintext is never repeated for the same sector number. We also revisited FDE from an engineering perspective and showed how to modify the firmware of a solid-state drive to associate a short "diversifier" to every sector-plaintext pair (s, m). This diversifier makes it possible to encrypt the same plaintext into different ciphertexts, something that was previously impossible without additional storage.

Chapter 4

Key-Dependent Message Security

Contents

4.1 KDM Security via Splitting and Forgetting Technique 65
4.1.1 Analysis via Forgetful Oracle Replacement 68
4.1.2 KDM Security of the Ideal Cipher 72
4.2 Security of Even–Mansour Ciphers 76
4.2.1 KDM Attacks on Even–Mansour 76
4.2.2 KDM Security of Even–Mansour Ciphers 78
4.3 KDM Security using the H-coefficient technique 84
4.3.1 H-coefficient and KDM Security 84
4.3.2 KDM Security with a Generic Lemma 85
4.4 Security of Key Alternating Feistel Ciphers 88
4.4.1 KDM Security of Four-Round Key-Alternating Feistel 89
4.4.2 Sliding attack for r-rounds 100
4.5 Even-Mansour KDM security with H-coefficients 102
4.5.1 Security of 1-round Even-Mansour 102

Early on, the seminal paper of Goldwasser and Micali [GM84] pointed out that semantic security may not hold if the adversary gets to see an encryption of the secret key. This practice was generally perceived as a dangerous use of an encryption scheme but several studies have revealed that this security notion is both theoretically and practically important (like in FDE mechanisms such as BitLocker [BHHO08] where the encryption key may be stored in the page file and thus encrypted along with the disk content). An encryption scheme is said to be *Key-Dependent Message (KDM) secure* if it is secure even against an attacker who can encrypt messages that depend on the secret key. Formally, security is defined with respect to a set Φ of functions ϕ mapping keys to messages for which the adversary can obtain key-dependent encryptions. This security notion was first formalized by Black, Rogaway and Shrimpton [BRS03] for symmetric encryption and was subsequently extensively studied for both symmetric and asymmetric cryptosystems (see, e.g., [BRS03, HK07, HU08, MTY11, DS14, App14, LLJ15]).

The Even-Mansour (EM) construction introduced in [EM93] is the simplest block-cipher known based on a single public permutation P on n-bit strings (see Figure 4.1). It uses two independent n-bit keys (K_1, K_2) and on input an n-bit plaintext m, it outputs

$$\mathsf{EM}^{\mathsf{P}}((K_1, K_2), m) = K_2 \oplus \mathsf{P}(K_1 \oplus m) \ .$$

Its generalization, the *iterated Even–Mansour* construction (also known as the *key-alternating*

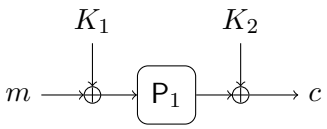

Figure 4.1: The Even–Mansour cipher (1-round).

cipher) was proposed by Daemen and Rijmen [DR01] as an abstraction of the design paradigm of substitution-permutation networks. This construction has become the object of abundant analysis and many recent block-ciphers follow this design (e.g., Present [BKL+07] and PRINCE [BCG+12]). If one models the underlying permutations as public random permutations, it is sometimes possible to prove the nonexistence of generic attacks against the iterated Even–Mansour construction (i.e., attacks that are possibly independent of a particular instantiation of the permutations $\mathsf{P}_1, \mathsf{P}_2, \ldots, \mathsf{P}_r$). If the adversary is only given black-box oracle access to these random permutations, the iterated Even–Mansour cipher was proved to achieve several security notions such as traditional indistinguishability (see [CLL+14] and references therein), security against related-key attacks [FP15, CS15], security in the multi-user setting [ML15, HT16] or indifferentiability from ideal ciphers[1] (see [DSST17] and references therein).

Another simple construction used to build secure block-ciphers is the Feistel Network. Feistel Network, introduced in the seminal Luby-Rackoff paper [LR86], is a construction that enables to build a $(\mathsf{n}_1 + \mathsf{n}_2)$-bits pseudorandom permutation family from a smaller random function family that takes n_1-bits inputs and gives n_2-bits outputs. The network is a

[1] This security notion roughly ensures that the construction "behaves" in some well-defined sense as an ideal cipher.

repetition of a simple network (the one-round Feistel network as shown in Figure 4.2) based on pseudorandom functions that can be the same for all rounds or different. Starting from the Luby-Rackoff result that the 3-round Feistel scheme is a pseudorandom permutation [LR86], Patarin [Pat90] proved that four rounds is indistinguishable from a strong pseudorandom permutation (where strong means that CCA adversary is considered). Other analysis gave better bounds for r rounds [Mau93, Pat98, MP03]. Dai and Steinberger[DS16] proved that the 8-rounds Feistel network is indifferentiable from random permutations and Barbosa and Farshim gave the analysis in the related key model[BF15].

Some Feistel block ciphers are *balanced* Feistel networks; the input is split into two equal length values L, R and we have a n-bits to n-bits round function (e.g. $\mathsf{n}_1 = \mathsf{n}_2$). For instance, DES and Simon [BTCS+15] are balanced whereas Bear, Lion [AB96], Misty [Mat97] and RC6 [IK01] are unbalanced [HR10]. Usually, the round function of practical block-ciphers (non idealized) is instantiated with a public random function and a round key K as shown in Figure 4.2. This special case of Feistel cipher is known as the Key-Alternating Feistel [LS15] and is of interest as it is a more practical construction; DES is a 16 rounds balanced KAF where all the public round functions are the same and where each subkey is derived from a master key.

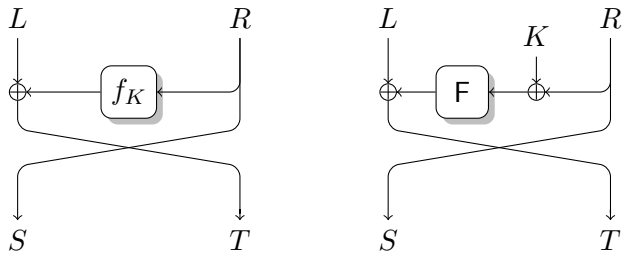

Figure 4.2: Descriptions of the round function of Feistel Network (Left) and the one round KAF (Right).

To be more formal, the Key-Alternating Feistel cipher [LS15] (KAF) is a Feistel network where the i-th round pseudorandom function F_i is instantiated by $\mathsf{F}_i(K_i, x) = f_i(K_i \oplus x)$ where the round functions f_i is a public function. The KAF construction is said to be idealized when the public functions f_i are random functions. Lampe and Seurin [LS15] analysed the indistinguishability of this construction for r rounds (see Figure 4.15) and gave a birthday security bound up to $2^{\frac{r'\mathsf{n}}{r'+1}}$ for $6r'$-rounds KAF using the H-coefficient technique. In these settings the adversary has to distinguish two systems $(\mathsf{KAF}, f_1, \ldots, f_r)$ and $(\mathsf{P}, f_1, \ldots, f_r)$ where f_i are the public random functions accessible for the adversary and P is a secret random permutation. They also noticed that two rounds of a KAF can be seen as a single-keyed EM cipher. Guo and Lin [GL15] proved that the 21-rounds KAF* construction, which is a variant of the KAF construction where the key K_i is XORed after the application of the public random function f_i, is indifferentiable from a random permutation. The most recent work [GW18] is the analysis of the KAF for short keys and in the multi-user settings. In the following, we consider only balanced KAF.

The analysis of the KDM security of Even-Mansour ciphers (section 4.2) and the KDM security of the Key-Alternating Ciphers uses two different methods. The Even-Mansour

ciphers are analysed in a new modular framework called the *splitting and forgetting technique* an detailed in section 4.1. This framework is usable for other block-ciphers and possibly also other forms of security under correlated inputs (like Related-Key Attack security) and it is based on a two-stage adversary: an adversary $\mathcal{A}$ against the KDM security of any cipher should be able to encrypt messages that depends on the key without knowing this key; that is why an intermediate adversary $\mathcal{B}$ which knows the key is used as an intermediate between $\mathcal{A}$ and the encryption oracle. We apply this framework to analyse the KDM security of the Ideal Cipher (subsection 4.1.2) and the 1 and 2-round Even-Mansour construction. This work was partially published [FKV17], at IACR Transactions on Symmetric Cryptology in 2018, with Pooya Fashim and Damien Vergnaud.

We give another method to analyse the KDM security of block-ciphers; based on the H-coefficient [Pat09] technique. The H-coefficient technique enables to upper bound the advantage of an adversary playing an indistinguishability game between an ideal world and a real world. To do so, the probability distributions of the transcripts resulting from the interactions of an adversary in the ideal world and in the real world are analysed. For the KDM setting, we introduce an intermediate game, the *perfect world*, where the key is not used and for each different input function the encryption oracle implements a random permutation. We apply twice the H-coefficient technique: first *generically* (this step is independent from the analysed block-cipher) between the ideal world and the perfect world and then between the perfect world and the real world for a specific block-cipher. The details of this methodology is given in section 4.3 and is applied to the Key-Alternating Feistel in section 4.4. The one-round Even-Mansour cipher is analysed with both techniques.

4.1 KDM Security via Splitting and Forgetting Technique

Our approach is to start with a block-cipher and gradually modify its oracles with independent ones until we arrive at a construction whose outputs are uniformly and independently distributed. In the particular case of Even–Mansour ciphers, we will replace at most two of the underlying permutations (namely P_1 and P_r) with oracles that completely randomize the outputs of the cipher (in both directions for decryption and encryption queries respectively); see Figure 4.3.

We consider a general security game where an adversary $\mathcal{A}$ has access to an oracle through two different interfaces. The approach consists in studying the conditions under which the security game can be modified (in an indistinguishable way for $\mathcal{A}$) so that the second interface provides access to an independent instance of this oracle. We also analyse the conditions under which this oracle can further be replaced by a *forgetful* oracle that completely removes dependency of outputs on inputs. For KDM security, we then have to prove that this replacement by forgetful oracles (after splitting) can be performed indistinguishably if the set of key-dependent messages functions that the adversary has at its disposal satisfies certain well-defined conditions. These conditions reduce to checking that the adversary $\mathcal{A}$ does not query these oracles on the same inputs (in the backward and forward direction in the case of random permutations). This general technique allows us to analyse the r-round iterated EM construction in a unified way. It is potentially applicable for an arbitrary number of rounds r but here we only apply to three EM cases with $r = 1, 2$ (the latter with(out) permutation reuse) and also the KDM security of the ideal cipher.

We first show that our *"splitting and forgetting"* technique is applicable to analyse the

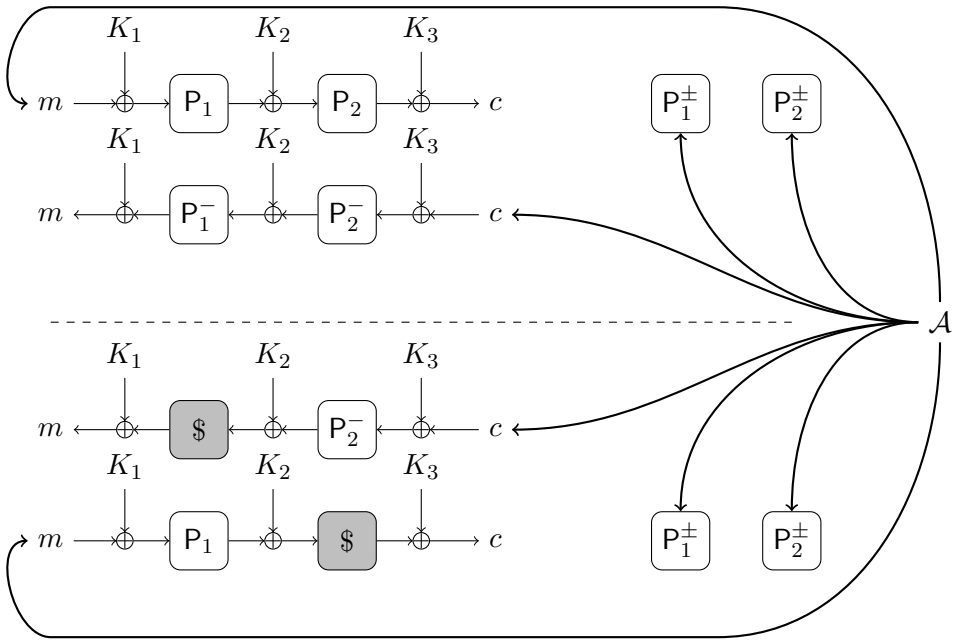

Figure 4.3: KDM-CCA analysis for 2-round Even–Mansour.

KDM security of the ideal cipher. Halevi and Krawczyk [HK07] prove that the ideal cipher achieves KDM security if one restricts the function class Φ to be a singleton and containing a function that is independent of the ideal cipher itself. Using our strategy, we prove the KDM security of the ideal cipher against adversaries with significantly larger classes of KDM functions, including functions that may depend on the ideal cipher. In the particular case where the functions are independent of the ideal cipher itself, it is sufficient to assume that the set of functions is *claw-free*, i.e., when distinct functions disagree on random inputs.

We then analyse the KDM security of the 1-round EM construction in the random-permutation model. We consider only sets of functions that are independent of underlying permutation (but our method can be extended to handle functions that depend on the underlying random permutation). We first present a simple attack that excludes the practically relevant case of KDM security with respect to the identity function (and more generally any offset of the key). On the positive side, we prove using our framework that the 1-round EM construction actually achieves KDM security under chosen-ciphertext attacks if the set of functions available to the attacker is claw-free and *offset-free*, i.e., when functions do not offset the key by a constant.

We apply the above method to study the KDM security of the 2-rounds EM construction in two configurations. We present a simple *slide attack* [BW99] on a variant with both permutation and key reuse where $K_1 = K_2 = K_3$ and $\mathsf{P}_1 = \mathsf{P}_2$ are used within the construction. The set of KDM functions considered contains the identity function (or more generally any key offsets). We also present a simple attack with complexity $2^{n/2}$ on the most general version. We then apply the framework to prove that 2-round EM achieves KDM security under chosen-ciphertext attacks if the set of functions available to the attacker is only claw-free as long as different permutations are used. When one reuses the same permutation $\mathsf{P}_1 = \mathsf{P}_2$ (because of efficiency reasons or because only one "good" public permutation is

available), we prove that EM achieves KDM-CCA security if the set of functions available to the attacker is claw-free and also *offset-xor-free*, meaning that functions do not output offsets of xor of two of the keys.

Our framework is general enough to be applied to other symmetric constructions and/or other security models. Indeed, we believe this approach can be used to re-derive the RKA security of EM ciphers [FP15] or that for Feistel networks [BF15] in a more modular way.

KDM FUNCTIONS. A key-dependent-message (KDM) function/circuit for key space $\mathcal{K}$ and message space $\mathcal{M}$ is a deterministic and stateless circuit $\phi : \mathcal{K} \longrightarrow \mathcal{M}$. A KDM set Φ is simply a set of KDM functions ϕ on the same key and message spaces. We assume membership in KDM sets can be efficiently decided. An *oracle* KDM function $\phi^{\mathcal{O}} : \mathcal{K} \longrightarrow \mathcal{M}$ is a KDM function with oracle gates.

KDM SECURITY. We now formalize security of block-ciphers under Key-Dependent Message and Chosen-Ciphertext Attacks (KDM-CCA). We do this in the $\mathcal{O}$-hybrid model of computation where oracle access to $\mathcal{O}$ sampled from some oracle space OSp is granted to all parties. For example, in the context of Even–Mansour ciphers, $\mathcal{O}(i, x, \sigma \in \{\pm\}) := \mathsf{P}_i^{\sigma}(x)$ for some random permutations $\mathsf{P}_i^{\pm}$. We therefore grant access to $\mathcal{O}$ to the KDM functions and the adversary. Security is now defined in the standard way via indistinguishability from the ideal cipher under a random key as shown in Figure 4.4.

Game $\text{KDM-CCA}^{\mathcal{A},\Phi}_{\text{BC}^{\mathcal{O}}}$	Proc. $\text{KDMEnc}(\phi^{\mathcal{O}})$
$\mathcal{O} \twoheadleftarrow \text{OSp}$	If $\phi^{\mathcal{O}} \notin \Phi$ Return $\perp$
$L \leftarrow [\,]$	$M \leftarrow \phi^{\mathcal{O}}(K)$; $C \leftarrow \mathsf{E}^{\mathcal{O}}(K, M)$
$b \twoheadleftarrow \{0,1\}$	If $b = 1$ Then $C \leftarrow i\mathsf{E}(K, M)$
$K \twoheadleftarrow \mathcal{K}$	$L \leftarrow L : C$; Return C
$(i\mathsf{E}, i\mathsf{D}) \twoheadleftarrow \text{Block}(\mathcal{K}, \mathcal{M})$	
$b' \twoheadleftarrow \mathcal{A}^{\mathcal{O},\text{KDMEnc},\text{Dec}}$	Proc. $\text{Dec}(C)$:
Return $(b' = b)$	If $C \in L$ Return $\perp$
	If $b = 1$ Return $i\mathsf{D}(K, C)$
	Return $\mathsf{D}^{\mathcal{O}}(K, C)$

Figure 4.4: Game defining Φ-KDM-CCA security for a block-cipher.

The adversary can ask for key-dependent encryption for functions $\phi^{\mathcal{O}} \in \Phi$ and decryption of ciphertexts of its choice.[2] To allow for expressive KDM sets and rule out trivial attacks, we do not allow decryption of ciphertexts that were obtained from the encryption oracle (as otherwise the key can be recovered by decrypting key-dependent ciphertexts). Given block-cipher $\text{BC}^{\mathcal{O}}$ and an oracle KDM set Φ, we define the advantage of an adversary $\mathcal{A}$ against $\text{BC}^{\mathcal{O}}$ with respect to Φ as

$$\mathbf{Adv}^{\text{kdm-cca}}_{\text{BC}^{\mathcal{O}}}(\mathcal{A}, \Phi) := 2 \cdot \Pr\left[\text{KDM-CCA}^{\mathcal{A},\Phi}_{\text{BC}^{\mathcal{O}}}\right] - 1 \ .$$

Feasibility of Φ-KDM-CCA security very much depends on the KDM functions available in Φ. For instance, if Φ contains the constant functions only, we recover the standard (strong)

[2] Note that we do not allow for key-dependent ciphertexts (KDC) in this work as the practical motivations are somewhat limited.

PRP notion of security, which is feasible under standard assumptions, in the RPM. On the other hand, the set Φ cannot be arbitrary. Consider for instance functions ϕ_i that zero all bits of K except the i-th one. Then using encryptions of $\phi_i(K)$ as well as those for $M_{b,i} := 1^{i-1}|b|0^{n-i}$ for $i = 1, \ldots, n$ and $b \in \{0, 1\}$ once can recover the key one bit at a time. For other sets, however, feasibility may or may not be possible. In the coming sections, we study this question for the EM ciphers.

4.1.1 Analysis via Forgetful Oracle Replacement

Our strategy to prove KDM security for block-ciphers is to gradually modify their internals until we arrive at constructions whose outputs are uniformly and independently distributed. For instance, in the case of EM ciphers we will replace one or more of their underlying permutations with *forgetful* random oracles. These will completely randomize the outputs of the cipher. To argue that this replacement can be performed indistinguishably, we will impose certain restrictions on how the adversary can interact with the cipher, and in particular the set of KDM functions at its disposal will be restricted.

Some notations. Throughout \$ denotes a *forgetful* oracle over some domain and range that on each input in the domain (even repeated ones) returns a uniformly chosen random element from the range. For a deterministic oracle machine $M^{\mathcal{O}}$ we denote by $\mathbf{Q}(M^{\mathcal{O}}(x))$ the list of query/answer pairs made to and received from $\mathcal{O}$ when M is run on input x. We recall that, for a list L of pairs (x, y), which may have repeats, we denote by $\mathsf{Dom}(L)$ the list of first entries x and by $\mathsf{Rng}(L)$ the list of second entries.

In this section we present a more general result that comes with a number of advantages: (1) it allows reusing parts of the analyses across different constructions, (2) it highlights the overall proof strategy and how various assumptions are used with it, and (3) it is potentially applicable to setting beyond KDM security, and/or to other constructions.

4.1.1.1 A framework for security analyses

The block-ciphers that we analyse are constructed in a model of computation where all parties have access to some oracle $\mathcal{O}$.[3] These oracles will be sampled from some oracle space OSp. We start with two assumptions on oracles that are of interest to us.

Splittability. Let $\mathbf{sp}(L_1, L_2)$ be a binary relation on lists L_1 and L_2. We say oracle $\mathcal{O}$ *splits* under $\mathbf{sp}$ if access to $\mathcal{O}$ through two interfaces can be modified in an indistinguishable way so that the second interface provides access to an independent instance $\mathcal{O}'$. Formally, we define the advantage of $\mathcal{D}$ in the split game as

$$\mathbf{Adv}^{\text{split}}_{\text{OSp}}(\mathcal{D}) := 2 \cdot \Pr\left[\text{Split}^{\mathcal{D}}_{\text{OSp}}\right] - 1 \ ,$$

where game $\text{Split}^{\mathcal{D}}_{\text{OSp}}$ is shown in Figure 4.5.

An alternative definition would quantify over all $\mathcal{D}$ that do not trigger $\mathbf{sp}$. Although $\mathbf{sp}$ is publicly checkable, this does not necessarily mean that every $\mathcal{D}$ can be modified to one with comparable advantage that never triggers $\mathbf{sp}$: the relation also depends on oracle outputs, which are outside the control of the distinguisher. Although, relations that we study here have the extra property that $\mathcal{D}$ *can* be modified to avoid triggering $\mathbf{sp}$, not all $\mathcal{D}$ will be able

[3] Access to multiple oracles can be modeled via domain separation.

Game $\text{Split}^{\mathcal{D}}_{\text{OSp}}$	Proc. $\text{O}(x)$	Proc. $\text{CHAL}(x)$
$\mathcal{O}, \mathcal{O}' \twoheadleftarrow \text{OSp}$	$y \leftarrow \mathcal{O}(x)$	If $b = 0$ Then $y \twoheadleftarrow \mathcal{O}(x)$
$b \twoheadleftarrow \{0,1\}$	$L_1 \leftarrow L_1 : (x, y)$	Else $y \leftarrow \mathcal{O}'(x)$
$b' \twoheadleftarrow \mathcal{D}^{\text{O},\text{CHAL}}$	Return y	$L_2 \leftarrow L_2 : (x, y)$
If $\mathbf{sp}(L_1, L_2)$ Then $b' \leftarrow 0$		Return y
Return $(b = b')$		

Figure 4.5: Game defining oracle splittability with respect to relation **sp**.

to perform this check. In particular certain *two-stage* distinguishers $\mathcal{D}$ cannot check for **sp** as the information needed for this check is spread among its two stages. For such $\mathcal{D}$ we need to keep **sp** in the game description.

FORGETFUL SWITCHING. We define the advantage of an algorithm $\mathcal{D}$ in the switch game as

$$\mathbf{Adv}^{\text{forget}}_{\text{OSp}}(\mathcal{D}) := 2 \cdot \Pr\left[\text{Forget}^{\mathcal{D}}_{\text{OSp}}\right] - 1 \ ,$$

where game $\text{Forget}^{\mathcal{D}}_{\text{OSp}}$ is formalized in Figure 4.6. Relation **fg** in this game will typically check for some form of repetitions in oracle queries. Replacing an oracle with a forgetful one removes any dependency of outputs on inputs.

Game $\text{Forget}^{\mathcal{D}}_{\text{OSp}}$	Proc. $\text{CHAL}(x)$
$\mathcal{O} \twoheadleftarrow \text{OSp}$; $b \twoheadleftarrow \{0,1\}$	If $b = 0$ Then $y \twoheadleftarrow \mathcal{O}(x)$
$b' \twoheadleftarrow \mathcal{D}^{\text{CHAL}}$	Else $y \leftarrow \$(x)$
If $\mathbf{fg}(L)$ Then $b' \leftarrow 0$	$L \leftarrow L : (x, y)$
Return $(b = b')$	Return y

Figure 4.6: Game defining forgetful switching property.

Consider now a modified split game (m-Split) that totally drops the **sp** check. We have the following result.

Theorem 4.1.1. *Let* OSp *be a lazily samplable oracle. Let* **fg** *(resp.* **sp***) be, by slight abuse of notation, the event that* $\mathcal{D}$ *triggers the check* **fg** *(resp.* **sp***) in the m-Split game. Then for any* $\mathcal{D}$ *in the modified split game we have a* $\mathcal{D}'$ *such that*

$$\mathbf{Adv}^{\text{m-split}}_{\text{OSp}}(\mathcal{D}) \leq \mathbf{Adv}^{\text{split}}_{\text{OSp}}(\mathcal{D}) + 2 \cdot \mathbf{Adv}^{\text{forget}}_{\text{OSp}}(\mathcal{D}') + \Pr[\mathcal{D} \textit{ sets } \mathbf{fg} | b = 1] + \Pr[\mathcal{D} \textit{ sets } \mathbf{sp} | b = 1] \ .$$

Proof. The proof follows 6 games hops and applies the the fundamental lemma of game playing as follows. Below, we let G_i be the event that $\mathcal{D}$ outputs $b' = 0$ in game i: $\Pr[G^{\mathcal{D}}_i] = \Pr[b' = 0]$.

Game_0**:** This is the $\text{m-Split}^{\mathcal{D}}_{\text{OSp}}$ game with $b = 0$ (i.e., with respect to oracles $(\mathcal{O}, \mathcal{O})$): $\Pr[G^{\mathcal{D}}_0] = \Pr[\text{m-Split}^{\mathcal{D}}_{\text{OSp}} | b = 0]$.

Game_1**:** In this game we introduce the **sp** check. This only increases the probability that $b' = 0$: $\Pr[G^{\mathcal{D}}_0] - \Pr[G^{\mathcal{D}}_1] \leq 0$.

Game_2: In this game we use an independent oracle $\mathcal{O}'$ for the challenge oracle. A direct reduction shows that: $\Pr[G_1^{\mathcal{D}}] - \Pr[G_2^{\mathcal{D}}] \leq \mathbf{Adv}^{\text{split}}_{\text{OSp}}(\mathcal{D})$.

Game_3: In this game we drop the **sp** check. Note that this game is identical to m-Split$^{\mathcal{D}}_{\text{OSp}}$ game with $b = 1$. This game and the previous one are identical until a flag $\mathbf{sp}_3$ corresponding to check **sp** is set: $\Pr[G_2^{\mathcal{D}}] - \Pr[G_3^{\mathcal{D}}] \leq \Pr[\mathcal{D} \text{ sets } \mathbf{sp}_3 | b = 1]$.

Game_4: In this game we introduce the **fg** check. This only increases the probability that $b' = 0$: $\Pr[G_3^{\mathcal{D}}] - \Pr[G_4^{\mathcal{D}}] \leq 0$.

Game_5: In this game we use a forgetful oracle \$ for the challenge oracle. Since $\mathcal{O}$ is assumed to be lazily samplable (and $\mathcal{O}$ and \$ are independent of $\mathcal{O}'$) via a direct reduction and simulation of $\mathcal{O}$ we get that for some $\mathcal{D}'$: $\Pr[G_4^{\mathcal{D}}] - \Pr[G_5^{\mathcal{D}}] \leq \mathbf{Adv}^{\text{forget}}_{\text{OSp}}(\mathcal{D}')$.

Game_6: Finally, we drop the **fg** check. This game and the previous one are identical until a flag $\mathbf{fg}_6$ is set: $\Pr[G_5^{\mathcal{D}}] - \Pr[G_6^{\mathcal{D}}] \leq \Pr[\mathcal{D} \text{ sets } \mathbf{fg}_6 | b = 1]$.

The analysis of the probability that $\mathcal{D}$ sets $\mathbf{sp}_3$ in Game_3 is pushed forward in Game_6 (Game_4, Game_5 and Game_6 are introduced to analyse this event). Let $\mathbf{sp}_i$ be the flag analogous to $\mathbf{sp}_3$ in Game i. Note that the probability of $\mathbf{sp}_3$ and that of $\mathbf{sp}_4$ are the same as checking condition **fg** has no effect on setting these flags. Using the fact that **sp** is publicly checkable (and hence can be used to define a distinguisher) we get that for some $\mathcal{D}''$

$$\Pr[\mathcal{D} \text{ sets } \mathbf{sp}_4 | b = 1] - \Pr[\mathcal{D} \text{ sets } \mathbf{sp}_5 | b = 1] \leq \mathbf{Adv}^{\text{forget}}_{\text{OSp}}(\mathcal{D}'').$$

Finally, the probability of setting $\mathbf{sp}_6$ is identical to that of $\mathbf{sp}_5$ as, once again, **fg** has no effect on these flags. The theorem now follows by adding the above inequities. □

We now consider a class of *two-stage* adversaries $\mathcal{D} = (\mathcal{A}, \mathcal{B})$. Adversary $\mathcal{A}$ can access the first oracle interface directly: this models the public availability of the oracle. Its access to the second interface, however, is restricted and is through algorithm $\mathcal{B}$ only. This algorithm holds information K unavailable to $\mathcal{A}$. It receives messages z from $\mathcal{A}$ and returns an output after interacting with the oracles through two interfaces. Formally, we say $\mathcal{D}$ is two stage if it can be written in the form shown in Figure 4.7 (left) for some algorithms $\mathcal{A}$ and $\mathcal{B}$. The operation of $\mathcal{D} = (\mathcal{A}, \mathcal{B})$ in the split game is shown on the right. Although algorithm $\mathcal{A}$ can be typically arbitrary, we will put restrictions on the operation of $\mathcal{B}$. For example, in the KDM setting $\mathcal{A}$ will correspond to the KDM adversary and $\mathcal{B}$ will model the operation of a block-cipher on key-dependent messages. More concretely, for 1-round EM:

$$\mathcal{B}^{\text{O},\text{Chal}}(K = (K_1, K_2), z = \phi) := \mathsf{EM}^{\text{O},\text{Chal}}[K_1, K_2](\phi^{\text{O}}(K_1, K_2)) \ .$$

We also assume that algorithm $\mathcal{B}$ is *stateless*; that is, it does not store any local state and each time is run a on fresh K and the incoming input z.[4] This means each instance of $\mathcal{B}(K, z_i)$ can be run independently. We also assume $\mathcal{B}$ is deterministic, and hence also that $\mathcal{A}$ queries $\mathcal{B}$ with distinct inputs z. Finally, we assume that $\mathcal{B}$ has simulatable outputs: its outputs on a random K and any z are indistinguishable from \$ when it is run with respect to oracles $(\mathcal{O}, \$)$.

[4] Any $(\mathcal{A}, \mathcal{B})$ with a stateful $\mathcal{B}$ can be modified to $(\mathcal{A}', \mathcal{B}')$ with stateless $\mathcal{B}'$ and an $\mathcal{A}'$ that sends the entire history of previous messages to $\mathcal{B}'$. This allows $\mathcal{B}'$ to recompute the state of $\mathcal{B}$. This modification however increases the query complexity of $\mathcal{B}$, and might not preserve other properties required from $\mathcal{B}$.

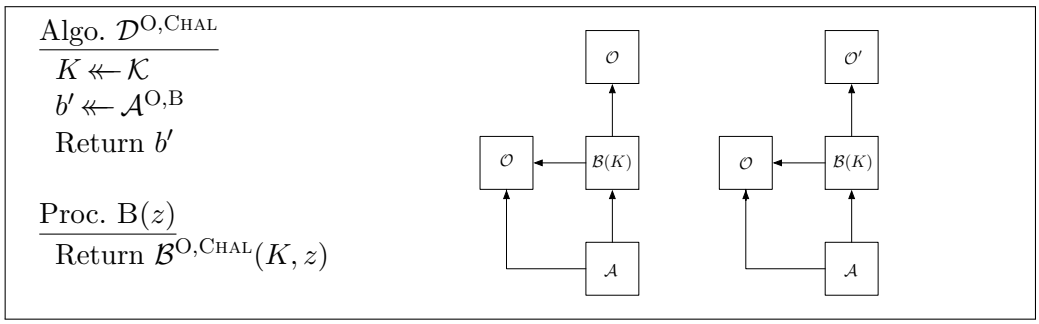

Figure 4.7: Two-stage adversaries and their operation in the split game.

We now consider the probability of setting **sp** or **fg** in m-Split for $\mathcal{D}$ that take the above form. We consider a setting where **sp** and **fg** can be expressed as disjunctions of simpler checks on pairs of distinct entries from the lists. More precisely, we assume for some algorithm **val**:

$$\mathbf{sp}(L_1, L_2) := \bigvee_{i,j} \mathbf{val}(L_1[i], L_2[j]) \quad \text{and} \quad \mathbf{fg}(L_2) := \bigvee_{i \neq j} \mathbf{val}(L_2[i], L_2[j]) \qquad (\star)$$

where $L[i]$ denotes the i-th element of the list L (which may contain repeats). Each clause depends on at most 2 elements. Hence a clause can be set by two entries corresponding to one of the following cases.

Sp1 : A direct $\mathcal{O}$ query of $\mathcal{A}$ and a challenge query of $\mathcal{B}(K, z_1)$ for some z_1.

Sp2 : An $\mathcal{O}$ query of $\mathcal{B}(K, z_1)$ and a challenge query of $\mathcal{B}(K, z_2)$ (possibly with $z_1 = z_2$).

Fg : Two challenge queries made by $\mathcal{B}(K, z_1)$ and $\mathcal{B}(K, z_2)$ with $z_1 \neq z_2$.

Therefore, triggering events Sp1 and Sp2 is equivalent to triggering **sp** and Fg is equivalent to **fg**. Note, for a $\mathcal{B}$ that does not place any $\mathcal{O}$ calls, event Sp2 never happens.

4.1.1.2 Some concrete cases

To applying the above theorem to KDM attacks, we start by observing that random oracles H, permutations $\mathsf{P}^{\pm}$ and ideal cipher $\mathsf{E}^{\pm}$ can be expressed as a single oracle $\mathcal{O}$ via encodings

$$\mathcal{O}(x) := \mathsf{H}(x) \ , \qquad \mathcal{O}(\sigma, x) := \mathsf{P}^{\sigma}(x) \ , \qquad \mathcal{O}(\sigma, K, x) := \mathsf{E}^{\sigma}(K, x) \ .$$

Using the standard PRP/PRF switching lemma [BR06] these oracles enjoy forgetful switching with respect to checks $\mathbf{fg}_{\mathrm{ro}}$, $\mathbf{fg}_{\mathrm{rp}}$, and $\mathbf{fg}_{\mathrm{ic}}$ defined via Equations $(\star)$ and

$$\begin{aligned}
\mathbf{val}_{\mathrm{ro}}((x_1, y_1), (x_2, y_2)) &:= (x_1 = x_2) \ , \\
\mathbf{val}_{\mathrm{rp}}((\sigma_1, x_1, y_1), (\sigma_2, x_2, y_2)) &:= (\sigma_1 = \sigma_2 \wedge x_1 = x_2) \vee (\sigma_1 \neq \sigma_2 \wedge (x_1 = y_2 \vee x_2 = y_1)) \ , \\
\mathbf{val}_{\mathrm{ic}}((\sigma_1, k_1, x_1, y_1), (\sigma_2, k_2, x_2, y_2)) &:= (k_1 = k_2) \wedge \mathbf{val}_{\mathrm{rp}}((\sigma_1, x_1, y_1), (\sigma_2, x_2, y_2)) \ .
\end{aligned}$$

Note that these conditions are publicly checkable. The advantage terms for q-query adversaries $\mathcal{D}$ and domain size 2^n are

$$\mathbf{Adv}^{\mathrm{forget}}_{\mathrm{Perm}(n)}(\mathcal{D}) \leq q^2/2^n \qquad \text{and} \qquad \mathbf{Adv}^{\mathrm{forget}}_{\mathrm{Block}(\mathsf{k},n)}(\mathcal{D}) \leq q^2/2^n \ .$$

These oracles also split with respect to $\mathbf{sp}_{\mathrm{ro}}$, $\mathbf{sp}_{\mathrm{rp}}$, and $\mathbf{sp}_{\mathrm{ic}}$ associated to their respective **val** above. This is immediate for random oracles (with advantage zero) as the systems (H, H) and $(\mathsf{H}, \mathsf{H}')$ are identical as long as the two interfaces are not queried on the same input. Splitting for ideal ciphers, and random permutations where $K = \varepsilon$, is proved easily.

Theorem 4.1.2 (Splitting for the ideal cipher)**.** *For any adversary $\mathcal{A}$ making at most q_1 queries to its first oracle and q_2 queries to its second oracle we have that*

$$\mathbf{Adv}^{\mathrm{split}}_{\mathrm{Block}(\mathsf{k},\mathsf{n})}(\mathcal{A}) \leq \frac{q_1 q_2}{2^n} \ .$$

Proof. Consider an adversary $\mathcal{A}$ with oracle access to $\mathcal{O}_1$ and $\mathcal{O}_2$. Algorithm $\mathcal{A}$ cannot ask the same query to its two oracles and it cannot ask to decrypt or encrypt a query to $\mathcal{O}_1$ that has been queried or obtained to $\mathcal{O}_2$ and inversely. $\mathcal{A}$ has to distinguish between two systems (E, E) and $(\mathsf{E}, \tilde{\mathsf{E}})$ where E and $\tilde{\mathsf{E}}$ are two independent ideal ciphers. After the attack $\mathcal{A}$ ends up with two lists L_1 and L_2 containing, respectively, the q_1 queries made to $\mathcal{O}_1$ and the q_2 queries made to $\mathcal{O}_2$. The only event that can enable $\mathcal{A}$ to trigger **sp** is an entry $(\sigma_1, K_1, x_1, y_1) \in L_1$ and another $(\sigma_2, K_2, x_2, y_2) \in L_2$ such that $(\sigma_1 \neq \sigma_2 \wedge (x_1 = y_2 \vee x_2 = y_1))$. The probability of this event is bounded by $q_1 q_2 / 2^n$. □

Relation between splitting and switching. Any oracle with forgetful replacement also allows for splitting: start with $(\mathcal{O}, \mathcal{O})$, replace *both* oracles to get $(\$, \$)$ and now switch the first oracle back to get $(\mathcal{O}, \$)$. This reduction, however, restricts the class of attacks that can be considered. Indeed in this reduction we would need to rely on $\mathbf{fg}(L_1 : L_2) \vee \mathbf{fg}(L_1)$ which imposes no repeat queries to the first oracle. This oracle is also used by $\mathcal{B}$, and hence we would have to assume that it does not place repeated queries to it. It might appear that this is not a problem as "without loss of generality" such repeat queries can be dealt with using lists. This, however, is not the case as different instances of $\mathcal{B}$ often cannot freely communicate with each other their *local* lists.

4.1.2 KDM Security of the Ideal Cipher

The KDM security for an ideal cipher is formulated as in Figure 4.4 with respect to an oracle $\mathcal{O}$ that implements an ideal cipher and an oracle $\mathcal{O}'$ that implements an indepedent ideal cipher. A trivial construction $\mathrm{BC}^{\mathcal{O}}$ that simply uses $\mathcal{O}$ to encipher and decipher inputs. We formulate a set of sufficient conditions on a KDM set Φ that allows us to establish KDM security for the ideal cipher.

Definition 4.1.3 (Claw-freeness)**.** *We define the (single-try) claw-freeness advantage of $\mathcal{A}$ against a KDM set Φ as*

$$\mathbf{Adv}^{\mathrm{cf}}_{\mathrm{OSp},\Phi}(\mathcal{A}) := \Pr[\phi_1^{\mathcal{O}} \neq \phi_2^{\mathcal{O}} \wedge \phi_1^{\mathcal{O}}(K) = \phi_2^{\mathcal{O}}(K) : \mathcal{O} \twoheadleftarrow \mathrm{OSp}; K \twoheadleftarrow \{0,1\}^{\mathsf{k}}; (\phi_1^{\mathcal{O}}, \phi_2^{\mathcal{O}}) \twoheadleftarrow \mathcal{A}^{\mathcal{O}}] \ .$$

We define the (multi-try) claw-freeness advantage $\mathbf{Adv}^{\mathrm{mcf}}_{\mathrm{OSp},\Phi}(\mathcal{A})$ by considering $\mathcal{A}$ that return two lists *of sizes q_1 and q_2 and claws are checked for two distinct ϕ's coming from the two lists. A simple guessing arguments shows that for any multi-try $\mathcal{A}$ there is a single-try $\mathcal{A}'$ such that:*

$$\mathbf{Adv}^{\mathrm{mcf}}_{\mathrm{OSp},\Phi}(\mathcal{A}) \leq q_1 q_2 \cdot \mathbf{Adv}^{\mathrm{cf}}_{\mathrm{OSp},\Phi}(\mathcal{A}') \ .$$

Informally, Φ is claw-free if the above advantage is "small" for every "reasonable" $\mathcal{A}$. When the KDM function are independent of $\mathcal{O}$ we may omit sampling of OSp from the game and notation.

The KDM set corresponding to xoring constants into the key:

$$\Phi^{\oplus} := \{\phi_i[\Delta] : (K_1, \ldots, K_{r+1}) \mapsto K_i \oplus \Delta : \Delta \in \mathcal{K}\} \cup \{(K_1, \ldots, K_{r+1}) \mapsto \Delta : \Delta \in \mathcal{K}\} \ .$$

is claw-free since the probability that $K_i \oplus \Delta_1 = K_j \oplus \Delta_2$ is 0 if $i = j$ and $\Delta_1 \neq \Delta_2$, and is negligible if $i \neq j$.

Query-independence. We define the query-independence advantage of $\mathcal{A}$ against a KDM set Φ with respect to oracle space $\mathrm{OSp} := \mathrm{Block}(\mathsf{k}, n)$ as

$$\mathbf{Adv}^{\mathrm{qi}}_{\mathrm{Block}(\mathsf{k},n),\Phi}(\mathcal{A}) := \Pr\left[\phi_1^{\mathcal{O}}(K) \in \mathbf{Q}_K^{+}(\phi_2^{\mathcal{O}}(K)) \text{ or } C \in \mathbf{Q}_K^{-}(\phi_2^{\mathcal{O}}(K)) : \right.$$
$$\left. \mathcal{O} \twoheadleftarrow \mathrm{Block}(\mathsf{k}, \mathsf{n}); K \twoheadleftarrow \{0,1\}^{\mathsf{k}}; (C, \phi_1^{\mathcal{O}}, \phi_2^{\mathcal{O}}) \twoheadleftarrow \mathcal{A}^{\mathcal{O}}\right] \ .$$

Here we have used the convention $\mathcal{O}(\sigma, K, M) := \mathsf{E}^{\sigma}(K, M)$. Note that any oracle-free KDM set is query-independent (*i.e.* has zero query-independence advantage).

We now prove that the ideal cipher is KDM secure for claw-free and query-independent KDM sets.

Theorem 4.1.4 (Ideal cipher KDM security)**.** *Let Φ be a KDM set for keys of length k and messages of length n. Suppose Φ is claw-free and query-independent as defined above. Then the ideal cipher is* Φ-KDM-CCA *secure. More precisely, for any adversary $\mathcal{A}$ against the* Φ-KDM-CCA *security of the ideal cipher for* $\mathrm{Block}(\mathsf{k}, \mathsf{n})$*, there is an adversary $\mathcal{C}_1$ against the claw-freeness of Φ and an adversary $\mathcal{C}_2$ against the query-independence of Φ such that*

$$\mathbf{Adv}^{\text{kdm-cca}}_{\mathrm{Block}(\mathsf{k},\mathsf{n}),\Phi}(\mathcal{A}) \leq q_1 q/2^{\mathsf{n}} + 2q^2/2^{\mathsf{n}} + q_1 q/2^{\mathsf{k}} + q^2(\mathbf{Adv}^{\mathrm{cf}}_{\mathrm{Block}(\mathsf{k},\mathsf{n}),\Phi}(\mathcal{C}_1) + 2/2^{\mathsf{n}}) +$$
$$+ 2q^2(\mathbf{Adv}^{\mathrm{qi}}_{\mathrm{Block}(\mathsf{k},\mathsf{n}),\Phi}(\mathcal{C}_2) + q_\phi/2^{\mathsf{n}}) \ .$$

Here q_1 is an upper bound on the number of direct queries of $\mathcal{A}$ to the ideal cipher (in either direction), q an upper bound on the number of challenge queries (globally), and q_ϕ an upper bound on the number of oracle queries of KDM functions. Adversaries $\mathcal{C}_1$ and $\mathcal{C}_2$ place at most q_1 queries to their ideal cipher oracles.

Proof. Let $\mathcal{A}'$ be a KDM-CCA adversary. We consider a two-stage adversary $(\mathcal{A}, \mathcal{B})$ against the modified split game as follows. Algorithm $\mathcal{A}$ runs $\mathcal{A}'$ and answers its ideal cipher queries using its own ideal-cipher oracle. It answers a KDM query ϕ of $\mathcal{A}'$ by forwarding $(+, \phi)$ to its $\mathcal{B}$ algorithm that is shown in Figure 4.8. It answers a decryption query C of $\mathcal{A}'$ by forwarding $(-, C)$ to $\mathcal{B}$. KDM functions are deterministic and stateless and we assume $\mathcal{A}'$ does not place repeat queries. Hence neither does $\mathcal{A}$. Recall that according to the rules of the KDM game no ciphertext obtained from encryption can be subsequently decrypted. We also assume, without loss of generality, that if a message M is obtained as a result of a ciphertext, then the *constant* function mapping keys to M cannot be queried to the encryption oracle (since the result is already known). We note that algorithm $\mathcal{B}$ is stateless, deterministic and places a single Chal query. It is also simulatable as when Chal implements \$ then so does $\mathcal{B}$.

It is easy to see that when Chal implements the original (non-replaced) oracle (i.e., when $b = 0$ in the m-Split game), algorithms $(\mathcal{A}, \mathcal{B})$ runs $\mathcal{A}'$ in the KDM game with $b = 0$. When

Algo. $\mathcal{B}^{\mathrm{O},\textsc{Chal}}(K,(+,\phi))$	Algo. $\mathcal{B}^{\mathrm{O},\textsc{Chal}}(K,(-,C))$
$M \leftarrow \phi^{\mathrm{O}}(K)$	
$C \leftarrow \textsc{Chal}(+,K,M)$	$M \leftarrow \textsc{Chal}(-,K,C)$
Return C	Return M

Figure 4.8: Algorithm $\mathcal{B}$ used for KDM analysis of the ideal cipher.

Chal implements a replaced ideal-cipher oracle, algorithms $(\mathcal{A},\mathcal{B})$ run $\mathcal{A}'$ in the KDM game with $b=1$. We emphasize that we are relying on the *modified* split game here as the split game performs the **sp** that does not exist in the KDM game. Hence

$$\mathbf{Adv}^{\text{kdm-cca}}_{\text{Block}(\mathsf{k},\mathsf{n})}(\mathcal{A}',\Phi) \le \mathbf{Adv}^{\text{m-split}}_{\text{Block}[\mathsf{k},\mathsf{n}]}(\mathcal{A},\mathcal{B}) \ .$$

Applying Theorem 4.1.1, it remains to bound the probability that $\mathcal{B}$ meets the three validity events Sp1, Sp2 and Fg with respect to $\mathbf{sp}_{\text{ic}}(L_1,L_2)$ and $\mathbf{fg}_{\text{ic}}(L_2)$ based on $\mathbf{val}_{\text{ic}}$ defined above.

Let us start with the Fg event: it is triggered with $z_1 \neq z_2$. Suppose $\sigma_1=\sigma_2=+$. In this case adaptivity can be ignored since the event does not depend on the output of the $\mathcal{B}$ oracle. Hence $\mathcal{C}$ must output $\phi_1^? \neq \phi_2^?$ such that

$$(+,K,\phi_1^{\mathcal{O}}(K)) = (+,K,\phi_2^{\mathcal{O}}(K))$$

This is equivalent to winning claw-freeness for Φ. When $\sigma_1=\sigma_2=-$ the event cannot be triggered as the ciphertexts must be distinct.

Let us consider now $\sigma_1=+$ and $\sigma_2=-$. Then $\mathcal{C}$ outputs $(+,\phi_1)$, receives a random value R, and then outputs $(-,C_2)$. Let R' be the output for the latter. Now it is either that (1) $\mathcal{B}(+,\phi_1)$ queries forward challenge on R', or (2) $\mathcal{B}(-,C_2)$ queries backward challenge on R. The former takes place with probability $1/2^{\mathsf{n}}$ as R' is chosen after ϕ_1. The latter can be triggered when $C_2=R$. But this is a disallowed queried by the rules of the KDM game: no encryption output can be decrypted.

Let $\sigma_1=-$ and $\sigma_2=+$. Then $\mathcal{C}$ outputs $(-,C_1)$, receives a random value R, and then outputs $(+,\phi_2)$. Let R' be the output for the latter. Now it is either that (1) $\mathcal{B}(-,C_1)$ queries backward challenge on R'. This happens with probability $1/2^{\mathsf{n}}$. Or that (2) $\mathcal{B}(+,\phi_2)$ queries forward challenge on R. This can be triggered in two ways: (2.1) ϕ_2 is different from the constant function mapping all inputs to R. In this case a claw is found. (2.2) ϕ_2 is the constant function mapping to R. But we have disallowed such queries.

Let us now look at Sp1. Since queries always include keys, the value x output by $\mathcal{C}$ must also include the key. The probability of guessing the key (given possibly a random value R) is at most $1/2^{\mathsf{k}}$.

If the KDM functions are oracle-independent, event Sp2 cannot be triggered and the analysis is done. For oracle-dependent KDM Sp2 can be triggered with z_1 and z_2 which correspond to either two forward or one forward and one backward query. (Since backward queries are oracle-independent, Sp2 cannot be triggered using two backward $\mathcal{B}$ queries.)

Suppose $i=1$. If $z_1=(-,*)$ then $L_1'=[\,]$. So we assume $z_1=(+,\phi_1)$. In what follows L_1' is formed first and then L_2'.

(1) Suppose $z_2 := (+, \phi_2)$. Then L'_2 consists of a single forward entry. (1.1) A forward entry in L'_1 with a forward entry in L'_2 trigger Sp2. This violates query-independence with a reduction that simply simulates R for $\mathcal{C}$. (1.2) A backward entry in L'_1 with a forward entry in L'_2 trigger Sp2. (1.2.1) An input in L'_1 matches an output in L'_2. Since the output in L'_2 is chosen randomly and independently of inputs in L'_1, this happens with probability at most $q_\phi/2^n$, assuming the KDM functions make at most q_ϕ oracle queries. (1.2.2) An output in L'_1 matches an input in L'_2. The outputs in L'_1 are random subject to permutativity. Value R seen by $\mathcal{C}$ is simply a random value independent of outputs in L'_1. Hence the probability of the single entry in L'_2 matching one of the outputs in L'_1 is at most $q_\phi/2^{\mathsf{n}}$.

(2) Suppose $z_2 := (-, C)$. Then L'_2 consists of a single backward entry. (2.1) A forward entry in L'_1 with a backward entry in L'_2 trigger Sp2. (2.1.1) An input in L'_1 matches an output in L'_2. Since the output in L'_2 is random and independent of L'_1, this happens with probability $q_\phi/2^{\mathsf{n}}$. (2.1.2) An output in L'_1 matches an input in L'_2. Since the outputs in L'_1 are random subject to permutativity and R is random and independent of these values, this happens with probability $q_\phi/2^{\mathsf{n}}$. (2.2) A backward entry in L'_1 with a backward entry in L'_2 trigger Sp2. This violates query-independence.

Suppose now $i = 2$. If $z_2 = (-, *)$ then $L'_1 = [\,]$. So we assume $z_2 = (+, \phi_2)$. In what follows L'_2 is formed first and then L'_1.

(3) Suppose $z_1 := (+, \phi_1)$. Then L'_2 consists a single forward entry. (3.1) A forward entry in L'_1 with a forward entry in L'_2 trigger Sp2. This violates query-independence with a reduction that simply simulates R for $\mathcal{C}$. (3.2) A backward entry in L'_1 with a forward entry in L'_2 trigger Sp2. (3.2.1) An input in L'_1 matches an output in L'_2. Note that the KDM function can be chosen based on R, the output in L'_2. This violated query-independence. (3.2.1) An output in L'_1 matches an input in L'_2. Since the outputs in L'_1 are random subject to permutativity, and L'_2 is chosen before L'_1, this happens with probability $q_\phi/2^{\mathsf{n}}$.

(4) Suppose $z_1 := (-, C)$. Then L'_2 consists of a single backward entry. (4.1) A forward entry in L'_1 with a backward entry in L'_2 trigger Sp2. (4.1.1) An input in L'_1 matches an output in L'_2. Note that the KDM function can be chosen based on R, the output in L'_2. This violated query-independence. (4.1.2) An output in L'_1 matches an input in L'_2. Since the outputs in L'_1 are random subject to permutativity, and L'_2 is chosen before L'_1, this happens with probability $q_\phi/2^n$. (4.2) A backward entry in L'_1 with a backward entry in L'_2 trigger Sp2. This violates query-independence.

Only one of the above cases need to be considered, which justifies the final term in the advantage upper bound. □

Remark 4.1.5. The converse of the above theorem does not hold. The set $\Phi := \{\phi_1(K) := K, \phi_2(K) := K \oplus \mathrm{MSB}(K)\}$ is *not* claw-free as $\phi_1(K) = \phi_2(K)$ with probability 1/2. KDM security with respect to this set, however, can be proven along the following lines. Instead of simulatability of $\mathcal{B}^{\mathcal{O},\$}$, demand simulation with the help of a *claw-detection* oracle. This is an oracle that given ϕ_1 and ϕ_2 returns $(\phi_1(K) = \phi_2(K))$. This means that we can modify the validity game to one which allows $\mathcal{C}$ access to a claw-detection oracle. For oracle-free KDM functions this condition boils down to unpredictability of the key in the presence of a

claw detection oracle. This results in a characterization that is tight, as predicting the key under claws can be easily used to win the KDM game for ideal cipher (since claws can be read off from the outputs of the cipher).

4.2 Security of Even–Mansour Ciphers

This section 4.2 is dedicated to the study of the iterated Even-Mansour ciphers under key-dependent message attacks.

Even-Mansour ciphers. The (iterated) Even-Mansour ciphers consider the problem of constructing a block-cipher with a large key space from a single, or a small number of, permutations. Formally, the r-round Even-Mansour cipher (see Figure 4.9) in a model of computation with r permutations $\mathsf{P}_1^{\pm}, \ldots, \mathsf{P}_r^{\pm}$ with domain $\mathcal{M} = \{0,1\}^n$ is a block-cipher with key space $\mathcal{K} = \{0,1\}^{(r+1)n}$ and enciphering and deciphering algorithms

$$\mathsf{E}^{\mathsf{P}_1,\ldots,\mathsf{P}_r}((K_1,\ldots,K_{r+1}),M) := \mathsf{P}_r(\cdots \mathsf{P}_2(\mathsf{P}_1(M \oplus K_1) \oplus K_2)\cdots) \oplus K_{r+1} \ ,$$
$$\mathsf{D}^{\mathsf{P}_1^-,\ldots,\mathsf{P}_r^-}((K_1,\ldots,K_{r+1}),M) := \mathsf{P}_1^-(\cdots \mathsf{P}_{r-1}^-(\mathsf{P}_r^-(M \oplus K_{r+1}) \oplus K_r)\cdots) \oplus K_1 \ .$$

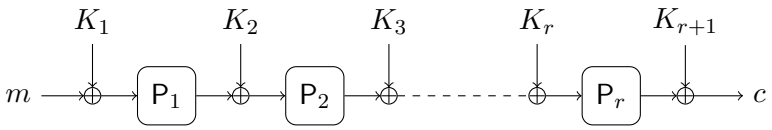

Figure 4.9: The r-round iterated Even–Mansour cipher.

The EM ciphers can be also considered in configurations where (some of the) keys and/or (some of the) permutations are reused in different rounds. We denote the EM cipher where P_i and K_{i+1} are used in round i by $\mathsf{EM}^{\mathsf{P}_1,\ldots,\mathsf{P}_r}[K_1, K_2, \ldots, K_{r+1}]$. The configuration with independent keys K_0 and K_1 achieves security up to approximatively $2^{n/2}$ queries and Dunkelman, Keller and Shamir [DKS12] show that the security bound holds when $K_0 = K_1$. The $r-$iterated EM construction has been deeply studied with $r+1$ independent keys and the tight security bound of $2^{(rn)/(r+1)}$ has been established in [CS14].

4.2.1 KDM Attacks on Even–Mansour

In this section we present KDM attacks on the iterated Even–Mansour ciphers. First, 1-round Even-Mansour is not KDM secure under chosen-plaintext attacks for any set Φ containing functions that offset the key, i.e., with respect to $\phi(K_1, K_2) := K_1 \oplus \Delta$. Indeed, enciphering $\phi(K_1, K_2)$ gives $\mathsf{P}(\Delta) \oplus K_2$ and hence key K_2 can be recovered after computing $\mathsf{P}(\Delta)$. Our result in the next section excludes such Φ.

We next consider 2-round EM in different configurations: the two permutations can be set to be identical or independent, and there are five possible key schedules. The simplest possible (and also the most efficient) construction uses a single random permutation and the same n-bit key in the two rounds. In the resulting scheme $\mathsf{EM}^{\mathsf{P},\mathsf{P}}[K, K, K]$, only one key needs to be securely stored and a unique random permutation has to be implemented.

Unfortunately, this cipher is vulnerable to a *sliding attack*[5] [BW99] if the set Φ contains the key offset functions.

Indeed, if the function $\phi(K) := K \oplus \mathsf{P}^{-1}(0^n)$ belongs to Φ, the attacker can simply query its encryption on it to get $C_1 = \mathsf{EM}^{\mathsf{P},\mathsf{P}}[K,K,K](\phi(K)) = \mathsf{P}(K) \oplus K$. It also obtains the encryption of 0^n as $C_2 = \mathsf{EM}^{\mathsf{P},\mathsf{P}}[K,K,K](0^n) = \mathsf{P}(\mathsf{P}(K) \oplus K) \oplus K$. The attacker can now recover the key as $\mathsf{P}(C_1) \oplus C_2$. The adversary $\mathcal{A}^{\mathsf{P},\mathsf{EM}^{\mathsf{P},\mathsf{P}}[K,K,K]}$, formally described in Figure 4.10 (left), can recover the key and this attack can easily be adapted to any number of rounds if all internal permutations are identical and all keys are equal. We note that $\mathcal{A}^{\mathsf{P},\mathsf{EM}^{\mathsf{P},\mathsf{P}}[K,K,K]}$ can trigger the following event and does not respect $\mathbf{fg}_{\mathrm{rp}}$ for $\mathcal{B}^{\mathsf{P},\textsc{Chal}}$: $(\sigma_1, x_1, y_1) = (\sigma_1, x_1, y_1)$ with $\sigma_1 = +$; $x_1 = \mathsf{P}(\Delta \oplus K) \oplus K$; $y_1 = \mathsf{P}(\mathsf{P}(\Delta \oplus K) \oplus K) \oplus K$.

This attack can be adapted to the key schedule $[K_1, K_2, K_2]$ as described by the adversary $\mathcal{A}^{\mathsf{P},\mathsf{EM}^{\mathsf{P},\mathsf{P}}[K_1,K_2,K_2]}$ shown in Figure 4.10 (right). The function ϕ_2 is now different and aims to cancel the key K_1 and replace it by K_2 to bring the setting back to one where a single key is used.

Adversary $\mathcal{A}^{\mathsf{P},\mathsf{EM}^{\mathsf{P},\mathsf{P}}[K,K,K]}$	Adversary $\mathcal{A}^{\mathsf{P},\mathsf{EM}^{\mathsf{P},\mathsf{P}}[K_1,K_2,K_2]}$
Chooses a value Δ	Chooses a value Δ
$y_1 \leftarrow \mathsf{P}(-,\Delta)$	$y_1 \leftarrow \mathsf{P}(-,\Delta)$
$\phi_1(K) := y_1 \oplus K$	$\phi_1(K) := y_1 \oplus K_1$
$y_2 \leftarrow \text{KDMEnc}(\phi_1)$	$y_2 \leftarrow \text{KDMEnc}(\phi_1)$
$\phi_2(K) := \Delta$	$\phi_2(K) := K_1 \oplus K_2 \oplus \Delta$
$y_2' \leftarrow \text{KDMEnc}(\phi_2)$	$y_2' \leftarrow \text{KDMEnc}(\phi_2)$
$y_1' \leftarrow \mathsf{P}(+, y_2)$	$y_1' \leftarrow \mathsf{P}(+, y_2)$
$K \leftarrow y_1' \oplus y_2'$	$K_2 \leftarrow y_1' \oplus y_2'$
$y \leftarrow \Delta \oplus K$	$y \leftarrow \Delta \oplus K_2$
$y_1'' \leftarrow \mathsf{P}(+, y)$	$y_1'' \leftarrow \mathsf{P}(+, y)$
If $y_2 = y_1'' \oplus K$ Return 1	If $y_2 = y_1'' \oplus K_2$ Return 1
$b \twoheadleftarrow \{0,1\}$	$b \twoheadleftarrow \{0,1\}$
Return b	Return b

Figure 4.10: Adversaries $\mathcal{A}^{\mathsf{P},\mathsf{EM}^{\mathsf{P},\mathsf{P}}[K,K,K]}$ and $\mathcal{A}^{\mathsf{P},\mathsf{EM}^{\mathsf{P},\mathsf{P}}[K_1,K_2,K_2]}$.

We also show that the iterated Even–Mansour construction cannot achieve KDM-CPA security beyond the birthday bound (for any number of rounds $r \geq 2$) if the set Φ contains the key offset functions (even if the random permutations and the keys are different). The adversary can simply query the KDMEnc oracle on $q_1 \geq 1$ different messages (independent of the key) $m_1, \ldots, m_{q_1}$ and store the corresponding plaintext/ciphertext pairs (m_i, c_i) for $i \in \{1, \ldots, q_1\}$ in some hash table (indexed by the ciphertext values). The adversary can then query the KDMEnc oracle on $q_2 \geq 1$ key offset functions $\phi_i(K) = K_1 \oplus \Delta_j$ with different offsets Δ_j for $j \in \{1, \ldots, q_2\}$. For each corresponding ciphertext $z_j = \text{KDMEnc}(\phi_j)$, the adversary then looks for it in the hash table. If there exist c_i such that $z_j = c_i$ for $i \in \{1, \ldots, q_1\}$, then, since EM is a permutation, the adversary knows that $m_i = \Delta_j \oplus K_1$ and can retrieve K_1 as $m_i \oplus \Delta_j$. If $q_1 \cdot q_2 \simeq 2^n$ then, with high probability, the adversary will find such a collision and therefore the first round key. The complexity to find the first

[5] This attack can be generalized readily for iterated r-rounds EM construction if it uses a single random permutation and the same n-bit key for all rounds (irrelevant of the value $r \geq 2$).

round key is thus $O(2^{n/2})$ queries to KDMEnc to find the first key and the attack can be repeated to find the other keys. The overall complexity to recover the full secret key is thus $O((r-1)\cdot 2^{n/2})$ queries to KDMEnc (since the two keys of the last round can be obtained easily as described above).

4.2.2 KDM Security of Even–Mansour Ciphers

In the following, three configurations of Even-Mansour cipher are analysed: $\mathsf{EM}^{\mathsf{P}}[K_1, K_2]$, $\mathsf{EM}^{\mathsf{P}_1,\mathsf{P}_2}[K_1, K_2, K_3]$ and $\mathsf{EM}^{\mathsf{P},\mathsf{P}}[K_1, K_2, K_3]$.

4.2.2.1 One-round Even–Mansour

We study the KDM security of the basic Even–Mansour cipher with only a single round. We show that this construction achieves nontrivial forms of KDM security.

Definition 4.2.1 (Offset-Freeness)**.** *We define the offset-freeness advantage of $\mathcal{A}$ against a KDM set Φ consisting of functions $\phi : \{0,1\}^{rn} \longrightarrow \mathcal{M}$ as*

$$\mathbf{Adv}^{\text{offset}}_{\Phi}(\mathcal{A}) := \Pr[\phi(K_1,.,K_r) = K_i \oplus X : i \in \{1,.,r\}; (K_1,.,K_r) \twoheadleftarrow \{0,1\}^{rn}; (\phi, X) \twoheadleftarrow \mathcal{A}] \ .$$

Our next result shows that one-round EM is KDM-secure against oracle-free claw-free and offset-free KDM sets. Note that xor-ing with constants is *not* offset-free.

Theorem 4.2.2. *Let Φ be an oracle-free KDM mapping $2n$-bit keys to n-bit messages. Suppose Φ is offset-free and claw-free. Then $\mathsf{EM}^{\mathsf{P}}[K_1, K_2]$ is* Φ-KDM-CCA *secure. More precisely, for any adversary $\mathcal{A}$ against the* Φ-KDM-CCA *security of $\mathsf{EM}^{\mathsf{P}}[K_1, K_2]$, there are adversaries $\mathcal{C}_1$ and $\mathcal{C}_2$ against the offset-free and claw-free properties of Φ such that*

$$\mathbf{Adv}^{\text{kdm-cca}}_{\mathsf{EM}^{\mathsf{P}}[K_1,K_2],\Phi}(\mathcal{A}) \leq q_1 q/2^n + 2q^2/2^n + q_1 q(2\cdot\mathbf{Adv}^{\text{offset}}_{\Phi}(\mathcal{C}_1) + 4/2^n) + q^2(2\cdot\mathbf{Adv}^{\text{cf}}_{\Phi}(\mathcal{C}_2) + 2/2^n) \ ,$$

where q_1 is the number of queries of $\mathcal{A}$ to $\mathsf{P}^{\pm}$ and q is the number of challenge queries of $\mathcal{A}$ in either direction.

Proof. The proof structure is analogous to that for the KDM security of the ideal cipher. For $\mathcal{A}'$ a KDM-CCA adversary, we consider a two-stage adversary $(\mathcal{A}, \mathcal{B})$ against the modified split game as follows. Algorithm $\mathcal{A}$ will run $\mathcal{A}'$ as before forwarding its queries to algorithm $\mathcal{B}$ shown in Figure 4.11: the oracle $\mathcal{O}$ implements the random permutations ($\mathsf{P}^{\pm}$) and the oracle $\mathcal{O}'$ implements the random permutation ($\mathsf{P'}^{\pm}$) where P, $\mathsf{P'}$ are independent. We assume $\mathcal{A}$ does not place repeat queries, respects the rules of the KDM game, and if it obtains a message M as a result of decrypting a ciphertext C, it does not query the constant function mapping to M to encryption. Note that algorithm $\mathcal{B}$ is stateless, deterministic, simulatable, and places a single Chal query. This leads to the first two terms in the advantage upper bound.

Since we are only considering oracle-independent KDM functions we do not need to consider event Sp2. We consider Sp1 next. This event corresponds to finding a collision between a direct query of $\mathcal{A}$ and a challenge query of $\mathcal{B}$. The adversary can trigger this event in a number of ways as follows.

(1) Two forward queries (X, ϕ) are such that $\phi(K_1, K_2) \oplus K_1 = X$. This violates offset-freeness. Note that the order of the queries and their adaptivity do not matter as the winning condition is independent of the oracle output.

Algo. $\mathcal{B}^{\mathrm{O},\textsc{Chal}}(K,(+,\phi))$	Algo. $\mathcal{B}^{\mathrm{O},\textsc{Chal}}(K,(-,C))$
$(K_1,K_2) \leftarrow K$	$(K_1,K_2) \leftarrow K$
$X \leftarrow \phi(K_1,K_2) \oplus K_1$	$Y \leftarrow C \oplus K_2$
$Y \leftarrow \textsc{Chal}(+,X)$	$X \leftarrow \textsc{Chal}(-,Y)$
$C \leftarrow Y \oplus K_2$	$M \leftarrow X \oplus K_1$
Return C	Return M

Figure 4.11: Algorithm $\mathcal{B}$ used in the KDM analysis of one-round EM.

(2) Two backward queries (X, C) are such that $C \oplus K_2 = X$. This amounts to guessing K_2, which happens with probability at most $1/2^{\mathsf{n}}$.

(3) A forward X and a backward C are such that: (3.1) $X = R \oplus K_1$ for a possibly known R (the output of $\mathcal{B}$). This amounts to guessing K_1, which happens with probability at most $1/2^n$. (3.2) $C \oplus K_2 = \mathsf{P}(X)$. This amounts to guessing K_2, which happens with probability at most $1/2^{\mathsf{n}}$.

(4) A backward X and a forward ϕ are such that: (4.1) $X = R \oplus K_2$ for a possibly known R. This amounts to guessing K_2, which happens with probability at most $1/2^{\mathsf{n}}$. (4.2) $\phi(K_1,K_2) \oplus K_1 = \mathsf{P}^{-}(X)$. This violates offset-freeness.

The third term in the advantage bound in the statement of the theorem follows from a union bound.

We now consider the Fg event, which corresponding to finding two collisions between two distinct challenge queries of $\mathcal{B}$. The adversary can trigger this event in a number of ways.

(1) Two forward queries are such that $\phi_1 \neq \phi_2$ and $\phi_1(K_1,K_2) \oplus K_1 = \phi_2(K_1,K_2) \oplus K_1$. This violates claw-freeness.

(2) Two backward queries are such that $C_1 \oplus K_2 = C_2 \oplus K_2$ and $C_1 \neq C_2$. This is not possible.

(3) A forward ϕ and a backward C such that C is chosen first and: (3.1) $\phi(K_1,K_2) \oplus K_1 = R \oplus K_1$ where ϕ can possibly depend on R. If $\phi(K_1,K_2)$ is the constant function mapping to R, this query is not allowed by our restriction above. Otherwise a claw with constant function mapping to R is found. (3.2) $C \oplus K_2 = R' \oplus K_2$. Since R' is randomly and independently chosen, this happens with probability $1/2^{\mathsf{n}}$.

(4) A forward ϕ and a backward C such that ϕ is chosen first and: (4.1) $C \oplus K_2 = R \oplus K_2$. Here C can possibly depend on R. This violates the KDM rule that output ciphertexts (R here) are not subsequently decrypted. (4.2) $\phi(K_1,K_2) \oplus K_1 = R' \oplus K_1$. Since R' is randomly and independently chosen, this happens with probability $1/2^n$.

The forth term in the advantage bound follows from a union bound. □

4.2.2.2 Two-round Even–Mansour with independent permutations

As mentioned above, offset-freeness excludes the case of KDM security against key offsets. We ask if by addition of extra rounds to the Even–Mansour ciphers can boost KDM-CCA

security against this class. In this section we show the addition of a single extra round with *independent* permutations is sufficient for this. (In the next subsection, we will consider using a single permutation.) We consider an oracle $\mathcal{O}$ that allows access to two permutations via $\mathcal{O}(i, \sigma, x) := \mathsf{P}_i^{\sigma}(x)$. This is simply an ideal cipher oracle with two key values $i = 1, 2$. Hence splitting and forgetting apply to this oracle. Our proof strategy is as before, but to avoid offset-freeness we only replace the *last* permutation in forward queries and the *first* permutation in backward queries. The oracle $\mathcal{O}$ implements the random permutations $(\mathsf{P}_1^{\pm}, \mathsf{P}_2^{\pm})$ and the oracle $\mathcal{O}'$ implements the random permutations $(\mathsf{P}_1^{+}, \mathsf{P}_1{}'^{-}, \mathsf{P}_2{}'^{+}, \mathsf{P}_2^{-})$ where P_1, P_1', P_2, P_2' are independent. These will be sufficient to ensure that the outputs in both directions are randomized.

Theorem 4.2.3. *Let Φ be a KDM set that is claw-free. Then $\mathsf{EM}^{\mathsf{P}_1,\mathsf{P}_2}[K_1, K_2, K_3]$ is Φ-KDM-CCA secure. More precisely, for any adversary $\mathcal{A}$ against the Φ-KDM-CCA security of $\mathsf{EM}^{\mathsf{P}_1,\mathsf{P}_2}[K_1, K_2, K_3]$, there is an adversary $\mathcal{C}$ against the claw-free property of Φ such that*

$$\mathbf{Adv}^{\text{kdm-cca}}_{\mathsf{EM}^{\mathsf{P}_1,\mathsf{P}_2}[K_1,K_2,K_3],\Phi}(\mathcal{A}) \leq 16q_1q/2^n + 2q^2 \cdot (\mathbf{Adv}^{\text{cf}}_{\Phi}(\mathcal{C}) + 1/2^n) \ ,$$

where q_1 is the number of queries of $\mathcal{A}$ to $\mathsf{P}_i^{\pm}$ (globally) and q is the number of challenge queries of $\mathcal{A}$ in either direction.

Proof. The proof structure is analogous to that previous cases and we describe the associated algorithm $\mathcal{B}$ in Figure 4.12. It is easy to verify that this algorithm responds with KDM queries under $\mathsf{EM}^{\mathsf{P}_1,\mathsf{P}_2}[K_1, K_2, K_3]$ and ϕ and satisfies the requirements of statelessness, determinism, etc. as before. We emphasize that this algorithm does not make use of queries of the form $\textsc{Chal}(1, +, \cdot)$ or $\textsc{Chal}(2, -, \cdot)$. This means that queries to $\mathrm{O}(1, +, X_1)$ (i.e., those to P_1) and queries to $\mathrm{O}(2, -, X_1)$ (i.e., those to P_2^{-}) can be arbitrary and without any restrictions.

Algo. $\mathcal{B}^{\mathrm{O},\textsc{Chal}}(K, (+, \phi))$	Algo. $\mathcal{B}^{\mathrm{O},\textsc{Chal}}(K, (-, C))$
$(K_1, K_2, K_3) \leftarrow K$	$(K_1, K_2, K_3) \leftarrow K$
$X_1 \leftarrow \phi(K_1, K_2, K_3) \oplus K_1$	$X_3 \leftarrow C \oplus K_3$
$X_2 \leftarrow \mathrm{O}(1, +, X_1)$	$X_2 \leftarrow \mathrm{O}(2, -, X_3)$
$X_3 \leftarrow \textsc{Chal}(2, +, X_2 \oplus K_2)$	$X_1 \leftarrow \textsc{Chal}(1, -, X_2 \oplus K_2)$
$C \leftarrow X_3 \oplus K_3$	$M \leftarrow X_1 \oplus K_1$
Return C	Return M

Figure 4.12: Algorithm $\mathcal{B}$ used in the KDM analysis of two-round EM with two permutations.

We start with Sp1. This event can be triggered in one of the following ways corresponding to choice of an input to an internally replaced oracle and a direct oracle query.

(1) Forward inputs ϕ and X such that $\mathsf{P}_1(\phi(K) \oplus K_1) \oplus K_2 = X$. We argue the probability of this event is upper-bounded by $2q_1q/2^n$. There are two cases to be considered: (1.1a) The value $\phi(K) \oplus K_1$ has been queried to P_1. But this means the adversary can use $\mathsf{P}_1(\phi(K) \oplus K_1)$ and X to compute K_2. For each ϕ and all q_1 choices of X the probability is $q_1/2^n$ and thus $q_1q/2^n$ over all ϕ. (1.1b) The value $\phi(K) \oplus K_1$ has not been queried to P_1 and $\mathsf{P}_1(\phi(K) \oplus K_1)$ is randomly chosen outside the view of the adversary over a set of size at least $(2^n - q_1)$. Hence this case happens with probability at most $1/(2^n - q_1)$ which is $\leq 2/2^n$ for $q_1 \leq 2^n/2$. We thus get an overall probability

of $2q_1q/2^n$. We will use this line of argument below and other proofs later on. (1.2) For a known random R, $\mathsf{P}_2(X) = R \oplus K_3$. This amounts to guessing K_3 with probability $q_1q/2^n$ over all ϕ and X.

(2) Backward inputs C and X such that: (2.1) $\mathsf{P}_2^-(C_2 \oplus K_3) \oplus K_2 = X$. This amounts to guessing K_2 with probability $2q_1q/2^n$. (2.2) $R \oplus K_1 = \mathsf{P}_1^-(X)$ for a known value R. This amounts to guessing K_1 with probability $q_1q/2^n$ over all C and X.

(3) Forward ϕ and backward X such that for a known random R: (3.1) $R \oplus K_3 = X$. This amounts to guessing K_3 with probability $q_1q/2^n$. (3.2) $\mathsf{P}_1(\phi(K) \oplus K_1) \oplus K_2 = \mathsf{P}_2^-(X)$. This amounts to guessing K_2 with probability $2q_1q/2^n$.

(4) Backward C and forward X such that for a known random R: (4.1) $R \oplus K_1 = X$. This amounts to guessing K_1 with probability $q_1q/2^n$ over all C and X. (4.2) $\mathsf{P}_2^-(C_2 \oplus K_3) \oplus K_2 = \mathsf{P}_1(X)$. This amounts to guessing K_2 with probability $2q_1q/2^n$.

We now look at event Fg. This event can be triggered in the following ways.

(1) Forward inputs $\phi_1 \neq \phi_2$ such that $\mathsf{P}_1(\phi_1(K) \oplus K_1) \oplus K_2 = \mathsf{P}_1(\phi_2(K) \oplus K_1) \oplus K_2$. This violates claw-freeness.

(2) Backward inputs $C_1 \neq C_2$ such that $C_1 \oplus K_3 = C_2 \oplus K_3$. This is impossible.

Note that only P_2 in the forward direction and P_1^- in the backward direction are replaced. Hence these are all the collisions that need to be taken care of. The second term in the advantage bound follows.

Since $\mathcal{B}$ uses $\mathcal{O}$, we need to also consider Sp2. This event can be triggered in the following ways.

(1) Inputs ϕ_1 and C_2 such that ϕ_1 is chosen after seeing a random R and: (1.1) $\phi_1(K) \oplus K_1 = R \oplus K_1$. This is either a repeat query (when ϕ_1 is constant) or breaks claw-freeness. (1.2) $\mathsf{P}_1(\phi_1(K) \oplus K_1) = \mathsf{P}_2^-(C_2 \oplus K_3) \oplus K_2$. This amounts to guessing K_2 with probability $2q_1q/2^n$.

(2) Inputs ϕ_1 and C_2 such that C_2 is chosen after seeing random R and: (2.1) $\mathsf{P}_1(\phi_1(K) \oplus K_1) \oplus K_2 = \mathsf{P}_2^-(C_2 \oplus K_3)$. This amounts to guessing K_2 with probability $2q_1q/2^n$. (2.2) $C_2 \oplus K_3 = R \oplus K_3$. This event violates the KDM rule that an output ciphertext is not decrypted.

This concludes the proof of theorem. □

4.2.2.3 Two-round Even–Mansour with a single permutation

We now consider KDM security of two-round EM with permutation reuse.

Definition 4.2.4 (Offset-xor-freeness)**.** *We define the offset-xor-freeness advantage of $\mathcal{A}$ against a KDM set Φ consisting of functions $\phi : \{0,1\}^{rn} \longrightarrow \mathcal{M}$ as*

$$\mathbf{Adv}^{\mathrm{ox}}_{\Phi}(\mathcal{A}) := \Pr[\phi(K_1,.,K_r) = K_i \oplus K_j \oplus X : (i,j) \in \{1,.,r\}^2; i \neq j; \\ (K_1,.,K_r) \twoheadleftarrow \{0,1\}^{rn}; (\phi, X) \twoheadleftarrow \mathcal{A}]$$

Offset-xor-freeness and claw-freeness are sufficient for the KDM security of two-round EM with a single permutation.

Theorem 4.2.5. *Let Φ be a KDM set that is claw-free and offset-xor-free. Then $\mathsf{EM}^{\mathsf{P},\mathsf{P}}[K_1, K_2, K_3]$ (with a single permutation) is* Φ-KDM-CCA *secure. More precisely, for any adversary $\mathcal{A}$ against the* Φ-KDM-CCA *security of $\mathsf{EM}^{\mathsf{P},\mathsf{P}}[K_1, K_2, K_3]$, there is an adversary $\mathcal{C}_1$ against the claw-free property of Φ and an adversary $\mathcal{C}_2$ against the offset-xor-free property of Φ such that*

$$\mathbf{Adv}^{\text{kdm-cca}}_{\mathsf{EM}^{\mathsf{P},\mathsf{P}}[K_1,K_2,K_3],\Phi}(\mathcal{A}) \leq 9q_1q/2^{\mathsf{n}} + q^2(2 \cdot \mathbf{Adv}^{\text{cf}}_{\Phi}(\mathcal{C}_1) + \mathbf{Adv}^{\text{ox}}_{\Phi}(\mathcal{C}_2) + 9/2^{\mathsf{n}}) \ ,$$

where q_1 is the number of queries of $\mathcal{A}$ to $\mathsf{P}^{\pm}$ and q is the number of challenge queries of $\mathcal{A}$ in either direction.

Proof. The proof structure is analogous to that previous case and we only present the associated algorithm $\mathcal{B}$ in Figure 4.13 below.

Algo. $\mathcal{B}^{\mathrm{O},\textsc{Chal}}(K,(+,\phi))$	Algo. $\mathcal{B}^{\mathrm{O},\textsc{Chal}}(K,(-,zC))$
$(K_1,K_2,K_3) \leftarrow K$	$(K_1,K_2,K_3) \leftarrow K$
$X_1 \leftarrow \phi(K_1,K_2) \oplus K_1$	$X_3 \leftarrow C \oplus K_3$
$X_2 \leftarrow \mathrm{O}(+,X_1)$	$X_2 \leftarrow \mathrm{O}(-,X_3)$
$X_3 \leftarrow \textsc{Chal}(+,X_2 \oplus K_2)$	$X_1 \leftarrow \textsc{Chal}(-,X_2 \oplus K_2)$
$C \leftarrow X_3 \oplus K_3$	$M \leftarrow X_1 \oplus K_1$
Return C	Return M

Figure 4.13: Algorithm $\mathcal{B}$ used in the KDM analysis of two-round EM with a single permutation.

The oracle $\mathcal{O}$ implements ($\mathsf{P}^{\pm}$) and $\mathcal{O}'$implements ($\mathsf{P}^{\pm}, \mathsf{P'}^{\pm}$). The adversary can trigger Sp1 in a number of ways as described below.

(1) Two forward queries (ϕ, X) are such that $\mathsf{P}(\phi_1(K) \oplus K_1) \oplus K_2 = X$. This amounts to guessing K_2 with probability $2q_1q/2^{\mathsf{n}}$ (Recall that in the one-round construction we used offset-freeness at this stage.)

(2) Two backward queries (C, X) are such that $\mathsf{P}^{-}(C \oplus K_3) \oplus K_2 = X$. Again, this amounts to guessing K_2 with probability $2q_1q/2^{\mathsf{n}}$.

(3) A backward query C and a forward direct query X are such that: (3.1) $\mathsf{P}^{-}(C \oplus K_3) \oplus K_2 = \mathsf{P}(X)$. This amounts to guessing K_2 with probability $2q_1q/2^{\mathsf{n}}$. (3.2) For some random and known R we have $R \oplus K_1 = X$. This amounts to guessing K_1.

(4) A forward query ϕ and a backward direct query X are such that: (4.1) $\mathsf{P}(\phi(K) \oplus K_1) \oplus K_2 = \mathsf{P}^{-}(X)$. This amounts to guessing K_2 with probability $2q_1q/2^{\mathsf{n}}$. (4.2) $R \oplus K_3 = X$. This reduces to guessing K_3.

The adversary can trigger Fg in one of the ways described below.

(1) Two forward queries $\phi_1 \neq \phi_2$ are such that $\mathsf{P}(\phi_1(K) \oplus K_1) \oplus K_2 = \mathsf{P}(\phi_2(K) \oplus K_1) \oplus K_2$. This violates claw-freeness.

(2) Two backward queries $C_1 \neq C_2$ are such that $\mathsf{P}^-(C_1 \oplus K_3) \oplus K_2 = \mathsf{P}^-(C_2 \oplus K_3) \oplus K_2$. This is not possible.

(3) A forward ϕ and a backward C are such that C is chosen second and: (3.1) $\mathsf{P}(\phi(K) \oplus K_1) \oplus K_2 = R \oplus K_1$. Here R is a random value chosen after ϕ corresponding to the output of $\mathcal{B}$ on C. This happens with probability $1/2^{\mathsf{n}}$. (3.2) $\mathsf{P}^-(C \oplus K_3) \oplus K_2 = R \oplus K_1$. Here R is a random value corresponding to the output of $\mathcal{B}$ on ϕ. Here C can depend on R. This happens with probability $1/2^{\mathsf{n}}$: Even if K_3 is known, this event amounts to guessing $K_1 \oplus K_2$. This event happens with probability $2q^2/2^{\mathsf{n}}$. (Note that here we rely on round keys being different.)

(4) A forward ϕ and a backward C are such that ϕ is chosen second and: (4.1) $\mathsf{P}(\phi(K) \oplus K_1) \oplus K_2 = R \oplus K_1$. Here ϕ can depend on R. This happens with probability $1/2^n$: Even if we allow the value $\mathsf{P}(\phi(K) \oplus K_1)$ to be chosen, this amounts to guessing $K_1 \oplus K_2$. (4.2) $\mathsf{P}^-(C \oplus K_3) \oplus K_2 = R \oplus K_1$. Here R is chosen after C. This happens with probability $2q^2/2^{\mathsf{n}}$.

We also need to analyse event Sp2 here as $\mathcal{B}$ depends on the oracle. This event can be triggered as a result of a collision between two queries to $\mathcal{B}$ one of which is to the oracle and the other to the challenge. This can happen in one of the following ways.

(1) Inputs ϕ_1 and ϕ_2 such that: $\phi_1(K) \oplus K_1 = \mathsf{P}(\phi_2(K) \oplus K_1) \oplus K_2$. If the input to the permutation cannot be guessed, this happens with low probability. Else it violates xor-offset-freeness.

(2) Inputs C_1 and C_2 such that: $C_1 \oplus K_3 = \mathsf{P}^-(C_2 \oplus K_3) \oplus K_2$. Even given K_3, this amounts to guessing K_2.

(3) Inputs ϕ_1 and C_2 such that: (3.1) $\phi_1(K) \oplus K_1 = R \oplus K_1$. If R is chosen after ϕ_1 this happens with probability $1/2^{\mathsf{n}}$. Else, a constant ϕ_1 is not allowed and a non-constant ϕ_1 violates claw-freeness with the constant function mapping to R. (3.2) $C_2 \oplus K_3 = R \oplus K_3$. This violates the rules of the KDM game for a C_2 chosen after R and otherwise happens with probability $1/2^{\mathsf{n}}$. (3.3) $\mathsf{P}(\phi_1(K) \oplus K_1) = \mathsf{P}^-(C_2 \oplus K_3) \oplus K_2$. Even given K_3, this amounts to guessing K_2. This happens with probability $2q^2/2^{\mathsf{n}}$.

□

Remark 4.2.6 (Comments on Adversaries)**.** The adversary $\mathcal{A}^{\mathsf{P},\mathsf{EM}^{\mathsf{P},\mathsf{P}}[K,K,K]}$ described in Figure 4.10 triggers the following event that does not respect $\mathbf{fg}_{\mathrm{rp}}$ for $\mathcal{B}^{\mathsf{P},\textsc{Chal}}$: $(\sigma_1, x_1, y_1) = (\sigma_1, x_1, y_1)$ with $\sigma_1 = +$, $x_1 = \mathsf{P}(\Delta \oplus K) \oplus K$, $y_1 = \mathsf{P}(\mathsf{P}(\Delta \oplus K) \oplus K) \oplus K$. Similarly, the adversary against the scheme $\mathsf{EM}^{\mathsf{P},\mathsf{P}}[K_1, K_2, K_2]$ uses KDM functions without xor-offset-freeness. It is however not clear if xor-offset-freeness is indeed necessary for the general key schedule $\mathsf{EM}^{\mathsf{P},\mathsf{P}}[K_1, K_2, K_3]$.

Remark 4.2.7 (Open question and conjecture)**.** The simplest Even-Mansour configuration, where the internal permutations are the same and keys are equal, is not KDM-secure under a claw-free KDM set. An open question is what is the minimal configuration where the Even-Mansour Cipher is KDM secure under a KDM set that is only claw-free? We conjecture that the 3-round Even-Mansour cipher with the same permutations and independent keys is KDM-secure under a claw-free set.

4.3 KDM Security using the H-coefficient technique

KDM FUNCTIONS. In the following, we do not consider KDM functions $\phi^{\mathcal{O}}$ that depend on an oracle $\mathcal{O}$; we consider only oracle-free functions. Nonetheless, the adversary can have a direct access to an oracle $\mathcal{O}$ and he can use the answer from this oracle to build function ϕ. We(re)-define the KDM sets from a statistical point of view and without adversary which is useful when using H-coefficient. But the underlying properties (claw-freeness, offset-freeness, offset-xor-freeness), on the KDM set remains the same.

Definition 4.3.1 (Claw-freeness)**.** *Let Φ be a KDM set for key space $\mathcal{K}$ and message space $\mathcal{M}$. The claw-freeness of Φ is defined as*

$$\mathsf{cf}(\Phi) \coloneqq \max_{\phi_1 \neq \phi_2 \in \Phi} \Pr[\mathbf{k} \twoheadleftarrow \mathcal{K} : \phi_1(\mathbf{k}) = \phi_2(\mathbf{k})].$$

Definition 4.3.2 (Offset-Freeness)**.** *Fix integers $n, \ell > 0$. Let Φ be a KDM set for key space $\mathcal{K} = (\{0,1\}^n)^\ell$ and message space $\mathcal{M} = \{0,1\}^n$. The offset-freeness of Φ is defined as*

$$\mathsf{of}(\Phi) \coloneqq \max_{\substack{i \in \{1,\ldots,\ell\} \\ \phi \in \Phi,\, x \in \{0,1\}^n}} \Pr[\mathbf{k} = (K_1, \ldots, k_\ell) \twoheadleftarrow \mathcal{K} : \phi(\mathbf{k}) = K_i \oplus x].$$

Definition 4.3.3 (Offset-Xor-Freeness)**.** *Fix integers $\mathsf{n}, \ell > 0$. Let Φ be a KDM set for key space $\mathcal{K} = (\{0,1\}^{\mathsf{n}})^\ell$ and message space $\mathcal{M} = \{0,1\}^{\mathsf{n}}$. The offset-xor-freeness of Φ is defined as*

$$\mathsf{oxf}(\Phi) \coloneqq \max_{\substack{i \neq j \in \{1,\ldots,\ell\} \\ \phi \in \Phi,\, x \in \{0,1\}^{\mathsf{n}}}} \Pr[\mathbf{k} = (K_1, \ldots, K_\ell) \twoheadleftarrow \mathcal{K} : \phi(\mathbf{k}) = K_i \oplus K_j \oplus x].$$

4.3.1 H-coefficient and KDM Security

H-COEFFICIENTS. The H-coefficients technique [Pat09] was introduced by Patarin and is widely used to prove the security of block cipher constructions such that Even Mansour [CLL+14] or Feistel schemes [Pat04]. Consider a deterministic adversary $\mathcal{A}$ which takes no input, interacts with a set of oracles w (informally called a *world*), and returns a single bit b, which we write $\mathcal{A}^{\mathsf{w}} \Rightarrow b$. Given two worlds w_0 and w_1 (offering the same interfaces), the *advantage* of $\mathcal{A}$ in distinguishing w_0 and w_1 is defined as

$$\mathbf{Adv}_{\mathsf{w}_0,\mathsf{w}_1}(\mathcal{A}) \coloneqq |\Pr[\mathcal{A}^{\mathsf{w}_0} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{w}_1} \Rightarrow 1]|.$$

The interaction between the adversary $\mathcal{A}$ and the oracles is recorded in a transcript τ. The transcript is the list of all query/answer pairs respectively made by the adversary and returned by the oracles. Let $X_{\mathcal{A},\mathsf{w}}$ be the random variable distributed as the transcript resulting from the interaction of $\mathcal{A}$ with world w. A transcript τ is said to be *attainable for $\mathcal{A}$ and w* if this transcript can be the result of the interaction of $\mathcal{A}$ with world w (i.e., $\Pr[X_{\mathcal{A},\mathsf{w}} = \tau] > 0$).

Lemma 4.3.4 (H-Coefficients)**.** *Let w_0 and w_1 be two worlds and $\mathcal{A}$ be a distinguisher. Let $\mathcal{T}$ be the set of attainable transcripts for $\mathcal{A}$ and w_0, and let $\mathcal{T}_{good}$ and $\mathcal{T}_{bad}$ be two disjoint*

subsets of $\mathcal{T}$ *such that* $\mathcal{T} = \mathcal{T}_{good} \cup \mathcal{T}_{bad}$*. If there exist* ε_{bad} *such that*

$$\Pr[X_{\mathcal{A},\mathsf{w}_0} \in \mathcal{T}_{bad}] \leq \varepsilon_{bad}$$

and ε_{good} *such that for all* $\tau \in \mathcal{T}_{good}$*,*

$$\frac{Pr[X_{\mathcal{A},\mathsf{w}_1} = \tau]}{\Pr[X_{\mathcal{A},\mathsf{w}_0} = \tau]} \geq 1 - \varepsilon_{good},$$

then $\mathbf{Adv}_{\mathsf{w}_0,\mathsf{w}_1}(\mathcal{A}) \leq \varepsilon_{\mathrm{good}} + \varepsilon_{\mathrm{bad}}$*.*

The proof of this lemma is standard [CLL+14]. The adversary output is a deterministic function of the transcript, its distinguishing advantage is upper bounded by the statistical distance between $X_{\mathcal{A},\mathsf{w}_0}$ and $X_{\mathcal{A},\mathsf{w}_1}$.

For a given transcript $\mathcal{Q}_{\mathsf{F}}$ and a function F, we say that F extends $\mathcal{Q}_{\mathsf{F}}$, if $v = \mathsf{F}(u)$ for all $(u, v) \in \mathcal{Q}_{\mathsf{F}}$.

4.3.2 KDM Security with a Generic Lemma

Consider a blockcipher $\mathrm{BC}^{\mathcal{O}} := (\mathsf{K}, \mathsf{E}^{\mathcal{O}}, \mathsf{D}^{\mathcal{O}})$ with key space $\mathcal{K}$ and message space $\mathcal{M}$ based on some ideal primitive $\mathcal{O}$ sampled from some oracle space OSp. We formalize security under key-dependent message and chose-ciphertext attacks (KDM-CCA) as a distinguishing game between two worlds that we call the *real* and *ideal* worlds. Given a KDM set Φ, the adversary $\mathcal{A}$ has access to a KDM encryption oracle KDMEnc which takes as input a function $\phi \in \Phi$ and returns a ciphertext $y \in \mathcal{M}$, a decryption oracle Dec which takes as input a ciphertext $y \in \mathcal{M}$ and returns a plaintext $x \in \mathcal{M}$, and oracle $\mathcal{O}$. We do not allow the adversary to ask for decryption of key-dependent ciphertexts as we are not aware of any use cases where such an oracle is available. In the real world, a key K is drawn uniformly at random from $\mathcal{K}$ and $\mathrm{KDMEnc}(\phi)$ returns $\mathsf{E}^{\mathcal{O}}(K, \phi(K))$ while $\mathrm{Dec}(y)$ returns $\mathsf{D}^{\mathcal{O}}(K, y)$. The ideal world is similar to the real world except that $\mathsf{E}(K, \cdot)$ and $\mathsf{D}(K, \cdot)$ are replaced by a random permutation P and its inverse. To exclude trivial attacks, we do not allow decryption of ciphertexts that were obtained from the encryption oracle more precisely, such queries are answered with $\perp$ in both worlds), as otherwise the key can be recovered by decrypting the encryption of K. The real and ideal world are formally defined in Figure 4.14 (ignore the additional world pw for now). The KDM-CCA advantage of an adversary $\mathcal{A}$ against BC with respect to Φ is defined as

$$\mathbf{Adv}^{kdm-cca}_{\mathrm{BC}^{\mathcal{O}}}(\mathcal{A}, \Phi) := \mathbf{Adv}_{\mathsf{rw},\mathsf{iw}}(\mathcal{A}).$$

4.3.2.1 A Generic Lemma

In order to allow a convenient application of the H-coefficients technique when proving the KDM security of a block cipher $\mathrm{BC}^{\mathcal{O}}$, we introduce an intermediate world, called the perfect world (pw), defined in Figure 4.14. Note that this world does not involve any key. The encryption and decryption oracles lazily sample two independent random permutations stored respectively in tables $\mathsf{T}_{\mathsf{enc}}$ and $\mathsf{T}_{\mathsf{dec}}$, except that consistency is ensured for constant functions $\phi \in \Phi_{\mathcal{M}}$: when a decryption query $\mathrm{Dec}(y)$ is made with $y \notin \mathsf{Dom}(\mathsf{T}_{\mathsf{dec}})$, a plaintext x is

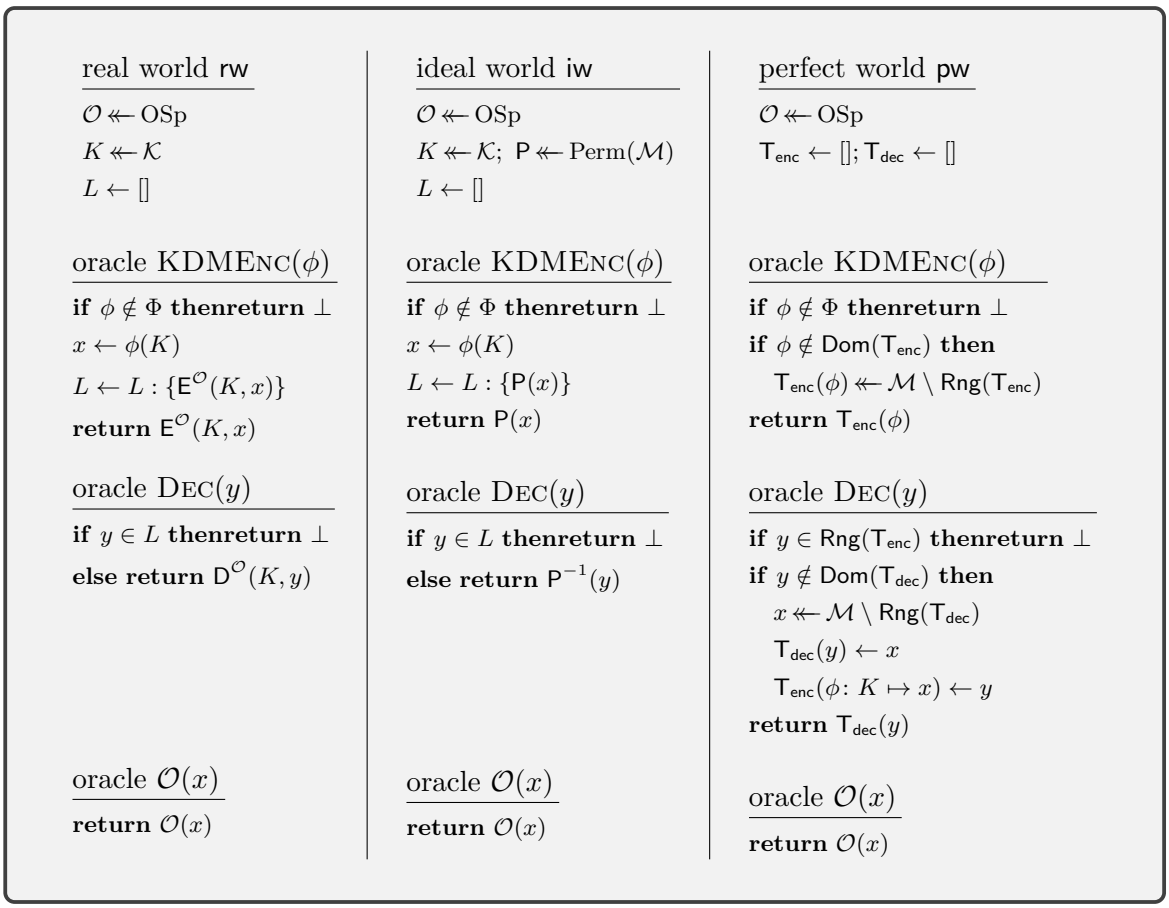

Figure 4.14: Real world rw and ideal world iw defining KDM-CCA-security and perfect world pw used in Theorem 4.3.5.

sampled from $\mathcal{M} \setminus \mathsf{Rng}(\mathsf{T}_{\mathsf{dec}})$ and the world assigns $\mathsf{T}_{\mathsf{dec}}(y) \coloneqq x$ and $\mathsf{T}_{\mathsf{enc}}(\phi) \coloneqq y$, where ϕ is the constant function $K \mapsto x$.

The following lemma upper-bounds the distinguishing advantage between the ideal and the perfect worlds. It does not depend on the block cipher at hand (neither the ideal nor the perfect world depends on it) nor on the oracle $\mathcal{O}$ (since neither in the ideal nor in the perfect world the encryption/decryption oracle depend on it). For specific block ciphers, it will allow to focus on the distinguishing advantage between the perfect and the real worlds, since by the triangular inequality,

$$\mathbf{Adv}_{\mathsf{iw},\mathsf{rw}}(\mathcal{A}) \leq \mathbf{Adv}_{\mathsf{iw},\mathsf{pw}}(\mathcal{A}) + \mathbf{Adv}_{\mathsf{pw},\mathsf{rw}}(\mathcal{A}). \tag{4.1}$$

Lemma 4.3.5. *Let Φ be a KDM set for key space $\mathcal{K}$ and message space $\mathcal{M}$. Let $\mathcal{A}$ be an adversary making at most q queries to* KDMEnc *or* Dec*. Then*

$$\mathbf{Adv}_{\mathsf{iw},\mathsf{pw}}(\mathcal{A}) \leq q^2 \cdot \mathsf{cf}(\Phi) + \frac{q^2}{\#\mathcal{M} - q}.$$

Proof. We apply the H-coefficients technique with $\mathsf{w}_0 = \mathsf{pw}$ and $\mathsf{w}_1 = \mathsf{iw}$. Fix a distinguisher

$\mathcal{A}$ making at most q encryption or decryption queries. We assume without loss of generality that

- the adversary never repeats a query;
- the adversary never queries $\phi : K \mapsto x$ to KDMEnc if it has received x as answer to some query Dec(y) before (since in both worlds such a query would be answered with y);
- the adversary never queries y to Dec if it has received y as answer to some query KDMEnc(ϕ) before (since in both worlds such a query would be answered with $\perp$).

We will refer to this as the *no-pointless-query* assumption.

We record the queries of the adversary to oracles KDMEnc or Dec in a list $\mathcal{Q}_{\mathrm{BC}}$: it contains all tuples $(+, \phi, y)$ such that $\mathcal{A}$ queried KDMEnc(ϕ) and received answer y, and all tuples $(-, x, y)$ such that $\mathcal{A}$ queried Dec(y) and received answer x. The queries of the adversary to oracle $\mathcal{O}$ are recorded in a list $\mathcal{Q}_{\mathcal{O}}$. After the adversary has finished querying the oracles, we reveal the key $\mathbf{k}$ in case the adversary interacts with the ideal world, while in the perfect world we reveal a uniformly random key independent from the oracles answers. Hence, a transcript is a triple $(\mathcal{Q}_{\mathrm{BC}}, \mathcal{Q}_{\mathcal{O}}, \mathbf{k})$. Let $\mathcal{T}$ be the set of attainable transcripts for $\mathcal{A}$ and pw. An attainable transcript $\tau = (\mathcal{Q}_{\mathrm{BC}}, \mathcal{Q}_{\mathcal{O}}, \mathbf{k})$ is said bad if one of the following conditions holds:

(C-1) there exists $(+, \phi, y) \neq (+, \phi', y') \in \mathcal{Q}_{\mathrm{BC}}$ such that $\phi(K) = \phi'(\mathbf{k})$;

(C-2) there exists $(+, \phi, y), (-, x, y') \in \mathcal{Q}_{\mathrm{BC}}$ such that $\phi(\mathbf{k}) = x$;

(C-3) there exists $(+, \phi, y), (-, x, y') \in \mathcal{Q}_{\mathrm{BC}}$ with $y = y'$.

Let $\mathcal{T}_{bad}$ denote the set of bad transcripts and let $\mathcal{T}_{good} \coloneqq \mathcal{T} \setminus \mathcal{T}_{bad}$. We first upper bound the probability to get a bad transcript in the perfect world.

Claim 1. *Let $\mathcal{A}$ be a distinguisher making at most $q \leq 2^{\mathsf{n}}$ queries to* KDMEnc *or* Dec*. With $\mathcal{T}_{bad}$ as defined above, one has*

$\Pr[X_{\mathcal{A},\mathsf{pw}} \in \mathcal{T}_{bad}] \leq q^2 \cdot \mathsf{cf}(\Phi) + \frac{3}{2} \cdot \frac{q^2}{\#\mathcal{M}-q}$.

Proof. We consider the probability of each condition in turn. Recall that in the perfect world, the key K is drawn at random independently from the oracle answers.

(C-1) Fix two queries $(+, \phi, y) \neq (+, \phi', y') \in \mathcal{Q}_{\mathrm{BC}}$. Then, by Theorem 4.3.1, $\phi(K) = \phi'(K)$ with probability at most $\mathsf{cf}(\Phi)$ over the draw of $\mathbf{k}$. By summing over all possible pairs, (C-1) happens with probability at most $q^2/2 \cdot \mathsf{cf}(\Phi)$.

(C-2) Fix two queries $(+, \phi, y), (-, x, y') \in \mathcal{Q}_{\mathrm{BC}}$. If query $(+, \phi, y)$ came first, then x is uniformly random in a set of size at least $\#\mathcal{M} - q$ and independent from $\phi(\mathbf{k})$, and hence $\phi(\mathbf{k}) = x$ with probability at most $1/(\#\mathcal{M}-q)$. If query $(-, x, y')$ came first, then by the no-pointless-query assumption, $\phi \neq (\mathbf{k} \mapsto x)$, so that $\phi(\mathbf{k}) = x$ with probability at most $\mathrm{cf}(\Phi)$ (since it constitutes a claw with the constant function). All in all, $\phi(\mathbf{k}) = x$ with probability at most $(1/(\#\mathcal{M} - q) + \mathrm{cf}(\Phi)$. By summing over all possible pairs of queries, (C-2) happens with probability at most $q^2/2 \cdot (1/(\#\mathcal{M} - q) + \mathrm{cf}(\Phi))$.

(C-3) Fix two queries $(+,\phi,y),(-,x,y') \in \mathcal{Q}_{\mathrm{BC}}$. If query $(+,\phi,y)$ came first, then, by the no-pointless-query assumption, $y' \neq y$. If query $(-,x,y')$ came first, then y is uniformly random in a set of size at least $\#\mathcal{M}-q$ and independent from y', hence $y=y'$ with probability at most $1/(\#\mathcal{M}-q)$. By summing over all possible pairs of queries, (C-3) happens with probability at most $q^2/(\#\mathcal{M}-q)$.

The result follows by the union bound. ■

Claim 2. *Fix a good transcript* $\tau=(\mathcal{Q}_{\mathrm{BC}},\mathcal{Q}_{\mathcal{O}},\mathbf{k})$. *Then*

$$\frac{\Pr[X_{\mathcal{A},\mathsf{iw}}=\tau]}{\Pr[X_{\mathcal{A},\mathsf{pw}}=\tau]} \geq 1.$$

Proof. Let q_{enc}, resp. q_{dec} denote the number of queries to KDMEnc, resp. Dec in $\mathcal{Q}_{\mathrm{BC}}$, with $q_{enc}+q_{\mathrm{dec}}=q$.

In the perfect world, queries to KDMEnc and Dec are answered by lazily sampling two independent injections $I_{enc}\colon[q_{enc}]\to\mathcal{M}$ and $I_{\mathrm{dec}}\colon[q_{\mathrm{dec}}]\to\mathcal{M}$ (this follows from the no-pointless-queries assumption which implies that for any query KDMEnc(ϕ) we have $\phi\notin\mathsf{Dom}(\mathsf{T}_{\mathsf{enc}})$ and for any query Dec(y) we have $y\notin\mathsf{Dom}(\mathsf{T}_{\mathsf{dec}})$). Hence, letting $\mathcal{Q}_{enc}$, resp. $\mathcal{Q}_{\mathrm{dec}}$ denote the set of encryption, resp. decryption queries in $\mathcal{Q}_{\mathrm{BC}}$, one has

$$\begin{aligned}\Pr[X_{\mathcal{A},\mathsf{pw}}=\tau] &= \Pr_{K'\twoheadleftarrow\mathcal{K}}[K'=K]\cdot\Pr_{\mathcal{O}\twoheadleftarrow\mathrm{OSp}}[\mathcal{O}\vdash\mathcal{Q}_{\mathcal{O}}]\cdot\Pr_{I_{enc}}[I_{enc}\vdash\mathcal{Q}_{enc}]\cdot\Pr_{I_{\mathrm{dec}}}[I_{\mathrm{dec}}\vdash\mathcal{Q}_{\mathrm{dec}}]\\ &= \Pr_{K'\twoheadleftarrow\mathcal{K}}[K'=K]\cdot\Pr_{\mathcal{O}\twoheadleftarrow\mathrm{OSp}}[\mathcal{O}\vdash\mathcal{Q}_{\mathcal{O}}]\cdot\frac{1}{(\#\mathcal{M})_{q_{enc}}\cdot(\#\mathcal{M})_{q_{\mathrm{dec}}}}.\end{aligned}$$

We now compute the probability to obtain τ in the ideal world. Consider the modified transcript $\mathcal{Q}'_{\mathrm{BC}}$ containing pairs $(x,y)\in\mathcal{M}^2$ constructed from $\mathcal{Q}_{\mathrm{BC}}$ as follows: for each triplet $(+,\phi,y)\in\mathcal{Q}_{\mathrm{BC}}$, append $(\phi(x),y)$ to $\mathcal{Q}'_{\mathrm{BC}}$ and for each $(-,x,y)\in\mathcal{Q}_{\mathrm{BC}}$, append (x,y) to $\mathcal{Q}'_{\mathrm{BC}}$. Then, for any $(x,y)\neq(x',y')\in\mathcal{Q}'_{\mathrm{BC}}$, we have $x\neq x'$ (as otherwise condition (C-1a), (C-2a), or (C-3a) would be met) and $y\neq y'$ (as otherwise condition (C-1b), (C-2b), or (C-3b) would be met). Hence,

$$\begin{aligned}\Pr[X_{\mathcal{A},\mathsf{iw}}=\tau] &= \Pr_{k'\twoheadleftarrow\mathcal{K}}[k'=k]\cdot\Pr_{\mathcal{O}\twoheadleftarrow\mathrm{OSp}}[\mathcal{O}\vdash\mathcal{Q}_{\mathcal{O}}]\cdot\Pr_{\mathsf{P}\twoheadleftarrow\mathrm{Perm}(\mathcal{M})}[\mathsf{P}\vdash\mathcal{Q}'_{\mathrm{BC}}]\\ &= \Pr_{k'\twoheadleftarrow\mathcal{K}}[k'=k]\cdot\Pr_{\mathcal{O}\twoheadleftarrow\mathrm{OSp}}[\mathcal{O}\vdash\mathcal{Q}_{\mathcal{O}}]\cdot\frac{1}{(\#\mathcal{M})_q}.\end{aligned}$$

Thus,

$$\frac{\Pr[X_{\mathcal{A},\mathsf{iw}}=\tau]}{\Pr[X_{\mathcal{A},\mathsf{pw}}=\tau]}=\frac{(\#\mathcal{M})_{q_{enc}}\cdot(\#\mathcal{M})_{q_{\mathrm{dec}}}}{(\#\mathcal{M})_q}\geq 1,$$

where the inequality follows from $q_{enc}+q_{\mathrm{dec}}=q$. □

The result follows by combining the two claims with Theorem 4.3.4.

□

4.4 Security of Key Alternating Feistel Ciphers

Key-Alternating Feistel (KAF). For a given public function $\mathsf{F}\in\mathcal{F}^*(\{0,1\}^{\mathsf{n}},\{0,1\}^{\mathsf{n}})$

and a key $K \in \{0,1\}^{\mathsf{n}}$, the one-round KAF is the permutation $\mathsf{KAF}^{\mathsf{F}}_K \in \mathcal{P}^*(\{0,1\}^{2\mathsf{n}})$ defined as:

$$\mathsf{KAF}^{\mathsf{F}}[K](LR) = R||\mathsf{F}(K \oplus R) \oplus L.$$

The value L and R are respectively the left and right halves n-bits of the input. And the left and right halves n-bits of the output are usually denoted S and T such that $S = R$ and $T = \mathsf{F}(K \oplus R) \oplus L$. The r-rounds version of KAF is specified by r public functions $\mathsf{F}_1, \mathsf{F}_2, \ldots, \mathsf{F}_r$ and r keys $K_1, K_2, \ldots, K_r$ such that:

$$\mathsf{KAF}^{\mathsf{F}_1,\mathsf{F}_2,\ldots,\mathsf{F}_r}[K_1, K_2, \ldots, K_r](LR) = \mathsf{P}^{\mathsf{F}_r}_{K_r} \circ \cdots \circ \mathsf{P}^{\mathsf{F}_1}_{K_1}(LR).$$

Usually, the round keys $K_1, K_2, \ldots, K_r$ are written as a key vector $\mathbf{k} = (K_1, K_2, \ldots, K_r)$.

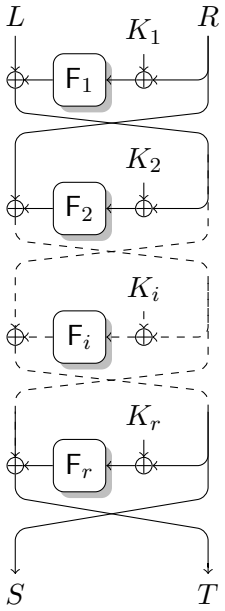

Figure 4.15: The r-round Key-Alternating Feistel (KAF).

4.4.1 KDM Security of Four-Round Key-Alternating Feistel

We study the 4-round KAF with a single round function F and key vector $\mathbf{k}$ such that $\mathbf{k} = (K_1, K_2, K_3, K_4)$. Given a function ϕ with range $\{0,1\}^{2\mathsf{n}}$, we let ϕ_L, resp. ϕ_R denotes the function which returns the n leftmost, resp. rightmost bits of ϕ. Given a KDM set Φ for message space $\mathcal{M} = \{0,1\}^{2\mathsf{n}}$, we define $\Phi_L \coloneqq \{\phi_L : \phi \in \Phi\}$ and $\Phi_R \coloneqq \{\phi_R : \phi \in \Phi\}$.

The 4-rounds KAF with the same round function $\mathsf{F} : \{0,1\}^{\mathsf{n}} \to \{0,1\}^{\mathsf{n}}$ and independent keys $K_i \in \{0,1\}^{\mathsf{n}}$ (such that $\mathbf{k} = (K_1, K_2, K_3, K_4)$), denoted $\mathsf{KAF}^{\mathsf{F},\mathsf{F},\mathsf{F},\mathsf{F}}[K_1, K_2, K_3, K_4]$ can be denoted $\mathsf{KAF}^{\mathsf{F}}_{\mathbf{k}}$ in the following. For this analysis, the oracles of the real world is configured as follows:

$$\mathcal{O} \coloneqq \mathsf{F} \text{ and } \mathrm{BC} \coloneqq \mathsf{KAF}^{\mathsf{F}}_{\mathbf{k}}.$$

A transcript τ is composed by the transcripts $\mathcal{Q}_{\mathrm{BC}}$ and $\mathcal{Q}_{\mathsf{F}}$, and the key vector $\mathbf{k}$ such

that: $\tau := (\mathbf{k}, \mathcal{Q}_{\mathrm{BC}}, \mathcal{Q}_{\mathsf{F}})$. $\mathcal{Q}_{\mathrm{BC}}$ is the list of forward queries $(+, \phi, ST)$ (ST is the answer of KDMEnc when called with the input ϕ) and backward queries $(-, L'R', S'T')$ ($L'R'$ is the answer of the oracle Dec when called with $S'T'$ as input) such that $\phi \in \Phi_R \times \Phi_L$, $(L'R', ST, S'T') \in (\{0,1\}^{2\mathsf{n}})^3$. $\mathcal{Q}_{\mathsf{F}}$ is the list of the queries (u, v) to the public function F (v is the answer of the oracle $\mathcal{O}$ when called with u as input) where $(u, v) \in (\{0,1\}^{\mathsf{n}})^2$.

Theorem 4.4.1 shows that the 4-rounds KAF, with the same rounds functions and independent keys, is CCA-KDM secure if the set of key-dependent functions $\Phi := \Phi_L \times \Phi_R$ has a Φ_R that is Offset-free and Offset-xor-free as defined respectively in Theorem 4.3.2 and Theorem 4.3.3.

Theorem 4.4.1. *For a 4-rounds Key-Alternating Cipher* $\mathsf{KAF}^{\mathsf{F}}_{\mathbf{k}}$ *where the key vector* $\mathbf{k} = (K_1, K_2, K_3, K_4)$ *is sampled uniformly at random, we have*

$$\mathbf{Adv}^{\text{kdm-cca}}_{\mathsf{KAF}^{\mathsf{F}}}(\mathcal{A}) \leq q^2(2 \cdot \mathsf{cf}(\Phi) + 3/2 \cdot \mathsf{oxf}(\Phi_R) + 14/2^{\mathsf{n}}) + qq_{\mathrm{f}}(\mathsf{of}(\Phi_R) + 7/2^{\mathsf{n}})$$

where q is the number of challenge queries of $\mathcal{A}$ in either direction and q_f is the number of queries of $\mathcal{A}$ to F.

Proof. Fix a distinguisher $\mathcal{A}$ making at most q queries to KDMEnc or Dec and q_{f} queries to F. By Equation (4.1) and Theorem 4.3.5, we have:

$$\begin{aligned}\mathbf{Adv}^{\text{kdm-cca}}_{\mathsf{KAF}^{\mathsf{F}}}(\mathcal{A}) \leq & q^2 \cdot \mathrm{cf}(\Phi) + q^2/(\#\mathcal{M} - q) + \mathbf{Adv}_{\mathsf{pw,rw}}(\mathcal{A}) \\ & q^2 \cdot \mathrm{cf}(\Phi) + q^2/2^{\mathsf{n}} + \mathbf{Adv}_{\mathsf{pw,rw}}(\mathcal{A}).\end{aligned}$$

Due to the fact that $\#\mathcal{M} = 2^{2\mathsf{n}}$ and $1/(2^{2\mathsf{n}} - q) \leq 1/2^{\mathsf{n}}$.

Hence, it remains to upper bound $\mathbf{Adv}_{\mathsf{pw,rw}}(\mathcal{A})$. For this, we use the H-coefficients technique.

Sketch of proof. We first give the definition of a bad transcript in Theorem 4.4.3 which is a transcript that enables to distinguish easily the perfect world from the real world. A bad transcript is defined according to conditions "C-i": these conditions correspond to possible collisions between the inputs of the first and the last functions of the KAF construction in the transcript $\mathcal{Q}_{\mathrm{BC}}$ and the input of the public function F in the transcript $\mathcal{Q}_{\mathsf{F}}$. These inputs are represented in red in left part of Figure 4.16. The adversary is able to check the presence of these collisions because the vector key is given in the transcript τ. The probability that $\mathcal{A}$ produces a bad transcript in the perfect world is upper-bounded in Theorem 4.4.4 and to make this probability low, note that the KDM set has to be offset-free, offset-xor-free and unsurprisingly claw-free. Then, conditioning in the fact that the transcript is good (condition C-i do not happen), the probability to pick a function F that leaks information is low (upper bounded in Theorem 4.4.8). Conditions "C'-i", as defined in Theorem 4.4.5, correspond to collisions that can happen between the inputs w_i, z_i and other inputs of F (inputs of the second and the third internal functions as shown in the left part of Figure 4.16) in case τ is a good transcript.

To be more formal, combining Theorem 4.3.4, Theorem 4.4.4, and Theorem 4.4.8, we obtain

$$\mathbf{Adv}_{\mathsf{pw,rw}}(\mathcal{A}) \leq q^2(\mathsf{cf}(\Phi) + 3/2 \cdot \mathsf{oxf}(\Phi_R) + 13/2^{\mathsf{n}}) + qq_{\mathrm{f}}(\mathsf{of}(\Phi_R) + 7/2^{\mathsf{n}})$$

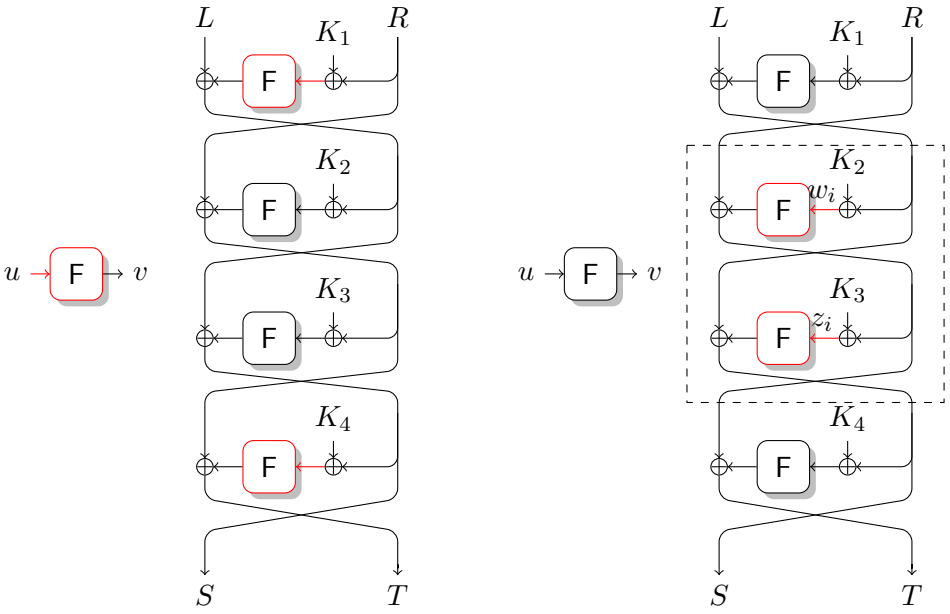

Figure 4.16: (left) In red, collisions giving a bad transcript (Condition C-i). (Right) In red, for a good transcript τ, F is bad if the inputs w_i and z_i collide with any other input of F (conditions C'-i). For a good transcript τ, F is good if the values w_i,z_i are all distinct;

from which the result follows. □

Remark 4.4.2 (Security does not rely on K_2 and K_3)**.** The proof of Theorem 4.4.1 does not rely on the keys K_2 and K_3 but only K_1 and K_4.

We first define *bad transcripts* and upper bound their probability in the perfect world. Informally, a transcript is said to be bad if some unexpected collision occurs in the set of all inputs to the first and the fourth round functions (note that the adversary can let some inputs collide with probability 1, for example by querying $\textsc{Dec}(ST)$ and $\textsc{Dec}(ST')$; bad transcripts only capture collisions happening by chance).

Definition 4.4.3. *A transcript* $\tau = (\mathcal{Q}_{\mathrm{BC}}, \mathcal{Q}_{\mathsf{F}}, \mathbf{k})$ *with* $\mathbf{k} = (K_1, K_2, K_3, K_4)$ *is said* bad *if one of the following conditions hold:*

(C-1) there exists $(+, \phi, ST) \in \mathcal{Q}_{\mathrm{BC}}$ *and* $(u, v) \in \mathcal{Q}_{\mathsf{F}}$ *such that*

- *(a)* $\phi_R(\mathbf{k}) \oplus K_1 = u$ *or*
- *(b)* $S \oplus K_4 = u$*;*

(C-2) there exists $(-, LR, ST) \in \mathcal{Q}_{\mathrm{BC}}$ *and* $(u, v) \in \mathcal{Q}_F$ *such that*

- *(a)* $R \oplus K_1 = u$ *or*
- *(b)* $S \oplus K_4 = u$*;*

(C-3) there exists $(+, \phi, ST) \neq (+, \phi', S'T') \in \mathcal{Q}_{\mathrm{BC}}$ *such that*

(a) $\phi(\mathbf{k}) = \phi'(\mathbf{k})$ *or*

(b) $S = S'$*;*

(C-4) there exists $(+, \phi, ST), (+, \phi', S'T') \in \mathcal{Q}_{\mathrm{BC}}$ *(not necessarily distinct) such that* $\phi_R(\mathbf{k}) \oplus K_1 = S' \oplus K_4$*;*

(C-5) there exists $(-, LR, ST) \neq (-, L'R', S'T') \in \mathcal{Q}_{\mathrm{BC}}$ *such that* $R = R'$*;*

(C-6) there exists $(-, LR, ST), (-, L'R', S'T') \in \mathcal{Q}_{\mathrm{BC}}$ *(not necessarily distinct) such that* $R \oplus K_1 = S' \oplus K_4$*;*

(C-7) there exists $(+, \phi, ST), (-, L'R', S'T') \in \mathcal{Q}_{\mathrm{BC}}$ *such that*

(a) $\phi(\mathbf{k}) = L'R'$ *or*

(b) $ST = S'T'$ *or*

(c) $\phi_R(\mathbf{k}) \oplus K_1 = S' \oplus K_4$ *or*

(d) $S \oplus K_4 = R' \oplus K_1$*.*

Let $\mathcal{T}_{bad}$ *denote the set of bad transcripts and let* $\mathcal{T}_{good} \coloneqq \mathcal{T} \setminus \mathcal{T}_{bad}$*.*

Lemma 4.4.4. *Let* $\mathcal{A}$ *be a distinguisher making at most* $q \leq 2^{\mathsf{n}}$ *queries to* KDMEnc *or* Dec *and* q_f *queries to* F*. With* $\mathcal{T}_{bad}$ *as defined above, one has*

$$\Pr[X_{\mathcal{A},\mathsf{pw}} \in \mathcal{T}_{bad}] \leq q^2(\mathsf{cf}(\Phi) + 3/2 \cdot \mathsf{oxf}(\Phi_R) + 6/2^{\mathsf{n}}) + qq_f(\mathsf{of}(\Phi_R) + 3/2^{\mathsf{n}}).$$

Proof. We compute the probability of each condition in turn. Remember that in pw, the key vector $\mathbf{k} = (K_1, K_2, K_3, K_4)$ is drawn at random independently of the oracles answers at the end.

(C-1) Fix an encryption query $(+, \phi, ST) \in \mathcal{Q}_{\mathrm{BC}}$ and a query to the public function $(u, v) \in \mathcal{Q}_{\mathsf{F}}$.

(a) By Theorem 4.3.2, $\phi_R(\mathbf{k}) = K_1 \oplus u$ with probability at most $\mathsf{of}(\Phi_R)$;

(b) Since k_4 is uniformly random and independent from the query transcript, $K_4 = S \oplus u$ with probability at most $1/2^{\mathsf{n}}$.

By summing over all possible pairs, condition (C-1) happens with probability at most $qq_{\mathrm{f}}(\mathsf{of}(\Phi_R) + 1/2^{\mathsf{n}})$.

(C-2) Fix a decryption query $(-, LR, ST) \in \mathcal{Q}_{\mathrm{BC}}$ and a query to F $(u, v) \in \mathcal{Q}_{\mathsf{F}}$.

(a) Since k_1 is uniformly random and independent from the query transcript, $R \oplus u = K_1$ with probability at most $1/2^{\mathsf{n}}$ (probability of guessing K_1).

(b) Since k_4 is uniformly random and independent from the query transcript, $S \oplus u = K_4$ with probability at most $1/2^{\mathsf{n}}$.

By summing over all possible pairs, condition (C-2) happens with probability at most $2qq_{\mathrm{f}}/2^{\mathsf{n}}$.

(C-3) Fix two queries $(+, \phi, ST) \neq (+, \phi', S'T') \in \mathcal{Q}_{\mathrm{BC}}$. Since the adversary never repeats queries, we have $\phi \neq \phi'$.

(a) By Theorem 4.3.1, $\phi(\mathbf{k}) = \phi'(\mathbf{k})$ with probability at most $\mathsf{cf}(\Phi)$ over the draw of $\mathbf{k}$.

(b) Since $\phi \neq \phi'$, the output $S'T'$ are uniformly sampled at random in a set a of size at least $2^{2\mathsf{n}} - q$ (possible values after q queries) and independently from ST. Thus, $S = S'$ with probability at most $2^{\mathsf{n}}/(2^{2\mathsf{n}} - q) \leq 1/(2^{\mathsf{n}} - 1) \leq 2/2^{\mathsf{n}}$.

By summing over all possible distinct pairs, condition (C-3) happens with probability at most $q^2/2 \cdot (\mathsf{cf}(\Phi) + 4/2^{\mathsf{n}})$.

(C-4) Fix two queries $(+, \phi, ST), (+, \phi', S'T') \in \mathcal{Q}_{\mathrm{BC}}$. By Theorem 4.3.3, $\phi_R(\mathbf{k}) = S' \oplus K_1 \oplus K_4$ with probability at most $\mathsf{oxf}(\Phi_R)$ over the draw of $\mathbf{k}$. By summing over all possible pairs, condition (C-4) happens with probability at most $q^2 \cdot \mathsf{oxf}(\Phi_R)$.

(C-5) Fix two decryption queries $(-, LR, ST) \neq (-, L'R', S'T') \in \mathcal{Q}_{\mathrm{BC}}$. The value $L'R'$ is sampled uniformly at random in a set of size at least $2^{2\mathsf{n}} - q$ and independently from LR. Thus, $R = R'$ with probability at most $2^{\mathsf{n}}/(2^{2\mathsf{n}} - q) \leq 1/(2^{\mathsf{n}} - 1) \leq 2/2^{\mathsf{n}}$. By summing over all possible distinct pairs, condition (C-5) happens with probability at most $q^2/2^{\mathsf{n}}$.

(C-6) Fix two decryption queries $(-, LR, ST), (-, L'R', S'T') \in \mathcal{Q}_{\mathrm{BC}}$. As K_1 and K_4 are randomly sampled, $R \oplus S' = K_1 \oplus K_4$ with probability at most $1/2^{\mathsf{n}}$. By summing over all possible distinct pairs, condition (C-6) happens with probability at most $q^2/2^{\mathsf{n}}$.

(C-7) Fix an encryption query $(+, \phi, ST) \in \mathcal{Q}_{\mathrm{BC}}$ and a decryption query $(-, L'R', S'T') \in \mathcal{Q}_{\mathrm{BC}}$.

(a) By Theorem 4.3.1, $\phi(\mathbf{k}) = L'R'$ with probability at most $\mathsf{cf}(\Phi)$.

(b) We distinguish two cases. If the encryption query occurs before the decryption query, then necessarily $ST \neq S'T'$ according to the no-pointless-query assumption (the adversary cannot ask to oracle Dec to decrypt a value that was received as an answer to the KDMEnc oracle). If the decryption query occurs before the encryption query, then ST is uniformly random in a set of size at least $2^{2\mathsf{n}} - q$ and independent from $S'T'$. Hence, the condition occurs with probability at most $1/(2^{2\mathsf{n}} - q) \leq 1/2^{\mathsf{n}}$.

(c) By Theorem 4.3.3, $\phi_R(\mathbf{k}) = S' \oplus K_1 \oplus K_4$ with probability at most $\mathsf{oxf}(\Phi_R)$ over the draw of $\mathbf{k}$.

(d) Since k_1 and k_4 are drawn uniformly at random and independently from the query transcript, the probability that $K_1 \oplus K_4 = S \oplus R'$ is at most $1/2^{\mathsf{n}}$.

By summing over all possible distinct pairs, condition (C-7) happens with probability at most $q^2/2 \cdot (\mathsf{cf}(\Phi) + \mathsf{oxf}(\Phi_R) + 4/2^{\mathsf{n}})$.

The result follow by the union bound over conditions (C-1) to (C-7). □

We now settle to lower bound $\Pr[X_{\mathcal{A},\mathsf{rw}} = \tau] / \Pr[X_{\mathcal{A},\mathsf{pw}} = \tau]$ for a good transcript τ. For this, we introduce the following definition. Informally, given a good transcript τ, a function F is bad with respect to τ is there is a collision in the set of all inputs to the second and third round functions.

Definition 4.4.5. *Fix a good transcript $\tau = (\mathcal{Q}_{\mathrm{BC}}, \mathcal{Q}_{\mathsf{F}}, \mathbf{k})$. Let*

$$\mathsf{Domain}(F) \coloneqq \{u \in \{0,1\}^{\mathsf{n}} : \exists (u,v) \in \mathcal{Q}_F\}$$

$$\mathsf{Domain}'(F) \coloneqq \left\{ u \in \{0,1\}^{\mathsf{n}} : \begin{array}{l} \exists (+,\phi,ST) \in \mathcal{Q}_{\mathrm{BC}}, u = \phi_R(\mathbf{k}) \oplus K_1 \vee \\ \exists (+,\phi,ST) \in \mathcal{Q}_{\mathrm{BC}}, u = S \oplus K_4 \vee \\ \exists (-,LR,ST) \in \mathcal{Q}_{\mathrm{BC}}, u = R \oplus K_1 \vee \\ \exists (-,LR,ST) \in \mathcal{Q}_{\mathrm{BC}}, u = S \oplus K_4 \end{array} \right\}.$$

A function F is said bad with respect to τ, *denoted* $\mathsf{bad}(\mathsf{F},\tau)$, *if one of the following conditions is fulfilled:*

(C'-1) there exists $(+,\phi,ST) \in \mathcal{Q}_{\mathrm{BC}}$ such that

(a) $\phi_L(\mathbf{k}) \oplus \mathsf{F}(\phi_R(\mathbf{k}) \oplus K_1) \oplus K_2 \in \mathsf{Domain}(\mathsf{F}) \cup \mathsf{Domain}'(\mathsf{F})$ or

(b) $T \oplus \mathsf{F}(S \oplus K_4) \oplus K_3 \in \mathsf{Domain}(\mathsf{F}) \cup \mathsf{Domain}'(\mathsf{F})$;

(C'-2) there exists $(-,LR,ST) \in \mathcal{Q}_{\mathrm{BC}}$ such that

(a) $L \oplus \mathsf{F}(R \oplus K_1) \oplus K_2 \in \mathsf{Domain}(\mathsf{F}) \cup \mathsf{Domain}'(\mathsf{F})$ or

(b) $T \oplus \mathsf{F}(S \oplus K_4) \oplus K_3 \in \mathsf{Domain}(\mathsf{F}) \cup \mathsf{Domain}'(\mathsf{F})$;

(C'-3) there exists $(+,\phi,ST) \neq (+,\phi',S'T') \in \mathcal{Q}_{\mathrm{BC}}$ such that

(a) $\phi_L(\mathbf{k}) \oplus F(\phi_R(\mathbf{k}) \oplus K_1) = \phi'_L(\mathbf{k}) \oplus \mathsf{F}(\phi'_R(\mathbf{k}) \oplus K_1)$ or

(b) $T \oplus \mathsf{F}(S \oplus K_4) = T' \oplus \mathsf{F}(S' \oplus K_4)$;

(C'-4) there exists $(+,\phi,ST),(+,\phi',S'T') \in \mathcal{Q}_{\mathrm{BC}}$ (not necessarily distinct) such that
$\phi_L(\mathbf{k}) \oplus \mathsf{F}(\phi_R(\mathbf{k}) \oplus K_1) \oplus K_2 = T' \oplus \mathsf{F}(S' \oplus k_4) \oplus K_3$;

(C'-5) there exists $(-,LR,ST) \neq (-,L'R',S'T') \in \mathcal{Q}_{\mathrm{BC}}$ such that

(a) $L \oplus \mathsf{F}(R \oplus K_1) = L' \oplus \mathsf{F}(R' \oplus K_1)$ or

(b) $T \oplus \mathsf{F}(S \oplus K_4) = T' \oplus \mathsf{F}(S' \oplus K_4)$;

(C'-6) there exists $(-,LR,ST),(-,L'R',S'T') \in \mathcal{Q}_{\mathrm{BC}}$ (not necessarily distinct) such that
$L \oplus \mathsf{F}(R \oplus K_1) \oplus K_2 = T' \oplus \mathsf{F}(S' \oplus K_4) \oplus K_3$;

(C'-7) there exists $(+,\phi,ST),(-,L'R',S'T') \in \mathcal{Q}_{\mathrm{BC}}$ such that

(a) $\phi_L(\mathbf{k}) \oplus F(\phi_R(\mathbf{k}) \oplus K_1) = L' \oplus F(R' \oplus K_1)$ or

(b) $T \oplus \mathsf{F}(S \oplus K_4) = T' \oplus \mathsf{F}(S' \oplus K_4)$ or

(c) $\phi_L(\mathbf{k}) \oplus F(\phi_R(\mathbf{k}) \oplus K_1) \oplus K_2 = T' \oplus F(S' \oplus k_4) \oplus k_3$ or

(d) $T \oplus F(S \oplus K_4) \oplus K_3 = L' \oplus F(R' \oplus K_1) \oplus K_2$.

Lemma 4.4.6. *Fix a good transcript $\tau = (\mathcal{Q}_{\mathrm{BC}}, \mathcal{Q}_{\mathsf{F}}, \mathbf{k})$. Then*

$$\Pr[\mathsf{F} \twoheadleftarrow \mathcal{F}^*(\{0,1\}^{\mathsf{n}},\{0,1\}^{\mathsf{n}}) : \mathsf{KAF}^{\mathsf{F}}_{\mathbf{k}} \vdash \mathcal{Q}_{\mathrm{BC}} \,|\, \mathsf{F} \vdash \mathcal{Q}_{\mathsf{F}} \wedge \neg \mathsf{bad}(\mathsf{F},\tau)] = \frac{1}{(2^{\mathsf{n}})^{2q}}.$$

Proof. Let q_{enc}, resp. q_{dec} denote the number of queries to KDMEnc, resp. Dec in $\mathcal{Q}_{\text{BC}}$, with $q_{enc} + q_{\text{dec}} = q$. Using an arbitrary order, let

$$\mathcal{Q}_{\text{BC}} = ((+, \phi_1, S_1T_1), \ldots, (+, \phi_{q_{\text{enc}}}, S_{q_{\text{enc}}}T_{q_{\text{enc}}}), \\ (-, L_{q_{\text{enc}}+1}R_{q_{\text{enc}}+1}, S_{q_{\text{enc}}+1}T_{q_{\text{enc}}+1}), \ldots, (-, L_qR_q, S_qT_q)).$$

For a given function F, let

$$\begin{aligned} w_i &= \phi_{i,L}(\mathbf{k}) \oplus \mathsf{F}(\phi_{i,R}(\mathbf{k}) \oplus K_1) \oplus K_2 && \text{for } 1 \le i \le q_{enc} \\ &= L_i \oplus \mathsf{F}(R_i \oplus K_1) \oplus K_2 && \text{for } q_{enc} + 1 \le i \le q \\ z_i &= T_i \oplus \mathsf{F}(S_i \oplus K_4) \oplus K_3 && \text{for } 1 \le i \le q. \end{aligned}$$

Distinct input values. In the following, we will show that for any good transcript $\tau = (\mathcal{Q}_{\text{BC}}, \mathcal{Q}_{\mathsf{F}}, \mathbf{k})$, with $\mathbf{k} = (K_1, K_2, K_3, K_4)$ where the function F is a good function, all the values w_i, z_i are distinct for all queries in the transcript $\mathcal{Q}_{\text{BC}}$.

Since the transcript τ is good and F is a good function then

→ all the inputs w_i for $1 \le i \le q_{enc}$ (e.g. forward queries) are pairwise distinct otherwise τ would fulfil condition (C'-3a);

→ all the values w_i for $q_{enc} + 1 \le i \le q_{enc}$ (e.g. backward queries) are pairwise distinct otherwise τ would fulfil condition (C'-5a);

→ all the values w_i for $1 \le i \le q_{enc}$ and all the values w_i for $1 \le i \le q_{enc}$ are distinct otherwise τ would fulfil condition (C'-7a).

Then all the values w_i are pairwise distinct in a transcript $\mathcal{Q}_{\text{BC}}$.

Similarly, since the transcript τ is good and F is a good function then

→ all the values z_i for $1 \le i \le q_{enc}$ are distinct from all values u for $u \in \mathcal{Q}_{\mathsf{F}}$ otherwise τ would fulfil condition (C'-1b);

→ all the values z_i for $q_{enc} + 1 \le i \le q_{enc}$ are distinct from all values u for $u \in \mathcal{Q}_{\mathsf{F}}$ otherwise τ would fulfil condition (C'-2b);

→ all the values z_i for $1 \le i \le q_{enc}$ are pairwise distinct otherwise τ would fulfil conditions (C'-3b);

→ all the values z_i for $q_{enc} + 1 \le i \le q$ are pairwise distinct otherwise τ would fulfil condition (C'-5b);

→ all the values z_i for $1 \le i \le q$ and all the value z_i for $q_{enc} + 1 \le i \le q$ are distinct otherwise τ would fulfil condition (C'-7b).

Then all the values z_i are pairwise distinct in a transcript $\mathcal{Q}_{\text{BC}}$.

Similarly, since the transcript τ is good and F is a good function then

$\rightarrow$ all the w_i for $1 \leq i \leq q_{enc}$ and all the values z_i for $1 \leq i \leq q_{enc}$ are distinct otherwise[$\rightarrow$] τ would fulfil condition (C'-4);

$\rightarrow$ all the w_i for $q_{enc}+1 \leq i \leq q$ and all the values z_i for $q_{enc}+1 \leq i \leq q$ are distinct otherwise τ would fulfil condition (C'-6);

$\rightarrow$ all the w_i for $1 \leq i \leq q_{enc}$ and all the values z_i for $q_{enc}+1 \leq i \leq q$ are distinct otherwise τ would fulfil condition (C'-7c);

$\rightarrow$ all the w_i for $q_{enc}+1 \leq i \leq q$ and all the values z_i for $1 \leq i \leq q_{enc}$ are distinct otherwise τ would fulfil condition (C'-7d).

Then all the values w_i and z_i are distinct.

Random output values. For all the $2q$ queries in $\mathcal{Q}_{\mathrm{BC}}$, we will show that the images $\mathsf{F}(w_i)$ and $\mathsf{F}(z_i)$ are undetermined and then uniformly random.
For each forward query $(+, \phi_i, S_iT_i) \in \mathcal{Q}_{\mathrm{BC}}$, we have

$$(+, \phi_i, S_iT_i) \in \mathcal{Q}_{\mathrm{BC}} \iff \left\{ \begin{array}{lll} \mathsf{F}(z_i) & = \phi_{i,L}(\mathbf{k}) \oplus S_i \oplus \mathsf{F}(\phi_{i,R}(\mathbf{k}) \oplus K_1) & \text{for } 1 \leq i \leq q_{enc} \\ \mathsf{F}(w_i) & = \phi_{i,R}(\mathbf{k}) \oplus T_i \oplus \mathsf{F}(S_i \oplus K_4) & \text{for } 1 \leq i \leq q_{enc} \end{array} \right\}$$

For any forward query:

The value z_i is different from $\phi_{i,R}(\mathbf{k}) \oplus K_1$ and different from the values u for $(u, v) \in \mathcal{Q}_{\mathsf{F}}$ otherwise the condition (C'-1a) would be fulfilled. Then the image $\mathsf{F}(z_i)$ is undetermined.

The value w_i is different from $S_i \oplus K_4$ and different from the values u for $(u, v) \in \mathcal{Q}_{\mathsf{F}}$ otherwise the condition (C'-1b) would be fulfilled. Then the image $\mathsf{F}(w_i)$ is undetermined.

This reasoning is true for all the possible good transcripts τ with a good function F. Consequently, for any $i \in \{1, \ldots, q_{enc}\}$,
$\Pr[\mathsf{KAF}^{\mathsf{F}}_{\mathbf{k}}(\phi_{i,L}(\mathbf{k})\phi_{i,R}(\mathbf{k})) = S_iT_i] = \Pr[\mathsf{F}(z_i) = \phi_{i,L}(\mathbf{k}) \oplus S_i \oplus X_i \oplus K_1) \wedge \mathsf{F}(w_i) = \phi_{i,R}(\mathbf{k}) \oplus T_i \oplus Y_i]$ where X_i a random value (for all transcripts). Thus, $\Pr[\mathsf{KAF}^{\mathsf{F}}_{\mathbf{k}}(\phi_{i,L}(\mathbf{k})\phi_{i,R}(\mathbf{k})) = S_iT_i] = 1/(2^{\mathsf{n}})^2$.

For each backward query $(-, L_iR_i, S_iT_i)$, we have

$$(-, L_iR_i, S_iT_i) \in \mathcal{Q}_{\mathrm{BC}} \iff \left\{ \begin{array}{lll} \mathsf{F}(z_i) & = S_i \oplus L_i \oplus \mathsf{F}(R_i \oplus K_1) & \text{for } q_{enc}+1 \leq i \leq q \\ \mathsf{F}(w_i) & = R_i \oplus T_i \oplus \mathsf{F}(S_i \oplus K_4) & \text{for } q_{enc}+1 \leq i \leq q \end{array} \right\}$$

Similarly, for any backward query:

The value z_i is different from $R_i \oplus K_1$ and different from the values u for $(u, v) \in \mathcal{Q}_{\mathsf{F}}$ otherwise the condition (C'-2a) would be fulfilled. Then the image $\mathsf{F}(z_i)$ is undetermined.

The value w_i is different from $S_i \oplus K_4$ and different from the values u for $(u, v) \in \mathcal{Q}_{\mathsf{F}}$ otherwise the condition (C'-2b) would be fulfilled. Then the image $\mathsf{F}(w_i)$ is undetermined.

This reasoning is true for all the possible good transcripts τ with a good function F. Consequently, for any $i \in \{q_{enc}+1, q\}$, $\Pr[\mathsf{KAF}^{\mathsf{F}}_{\mathbf{k}}(L_iR_i) = S_iT_i] = 1/(2^{\mathsf{n}})^2$.

The probability over the $2q$ queries gives the result. □

Lemma 4.4.7. *Fix a good transcript $\tau = (\mathcal{Q}_{\mathrm{BC}}, \mathcal{Q}_{\mathsf{F}}, \mathbf{k})$. Then*

$$\Pr[\mathsf{F} \twoheadleftarrow \mathcal{F}^*(\{0,1\}^{\mathsf{n}}, \{0,1\}^{\mathsf{n}}) : \mathsf{bad}(\mathsf{F}, \tau) \mid \mathsf{F} \vdash \mathcal{Q}_{\mathsf{F}}] \leq 4 \cdot qq_f/2^{\mathsf{n}} + 7 \cdot q^2/2^{\mathsf{n}}.$$

Proof. First, note that $\#\mathsf{Domain}(\mathsf{F}) = q_{\mathrm{f}}$ and $\#\mathsf{Domain}'(\mathsf{F}) \leq 2q$. We consider each condition in turn. (In all the following, we will argue using the fact that the transcript is good by referring to which specific condition defining a bad transcript would hold, saying e.g. "By ¬(C-ix), ...").

(C'-1) Fix an encryption query $(+, \phi, ST) \in \mathcal{Q}_{\mathrm{BC}}$.

(a) By ¬(C-1a), $\phi_R(\mathbf{k}) \oplus K_1$ is a fresh input for the function F and hence $\mathsf{F}(\phi_R(\mathbf{k}) \oplus K_1)$ is uniformly random. Thus, $\phi_L(\mathbf{k}) \oplus \mathsf{F}(\phi_R(\mathbf{k}) \oplus K_1) \oplus K_2 \in \mathsf{Domain}(\mathsf{F}) \cup \mathsf{Domain}'(\mathsf{F})$ with probability at most $(q_{\mathrm{f}} + 2q)/2^n$.

(b) By ¬(C-1b), $S \oplus K_4$ is a fresh input for the function F and hence $\mathsf{F}(S \oplus K_4)$ is uniformly random. Thus, $T \oplus \mathsf{F}(S \oplus K_4) \oplus K_3 \in \mathsf{Domain}(\mathsf{F}) \cup \mathsf{Domain}'(\mathsf{F})$ with probability at most $(q_{\mathrm{f}} + 2q)/2^n$.

By summing over all encryption queries, condition (C'-1) happens with probability at most $2q(q_{\mathrm{f}} + 2q)/2^{\mathsf{n}}$.

(C'-2) Fix a decryption query $(-, LR, ST) \in \mathcal{Q}_{\mathrm{BC}}$.

(a) By ¬(C-2a), $R \oplus K_1$ is a fresh input for the function F and hence $\mathsf{F}(R \oplus K_1)$ is uniformly random. Thus, $L \oplus \mathsf{F}(R \oplus K_1) \oplus K_2 \in \mathsf{Domain}(\mathsf{F}) \cup \mathsf{Domain}'(\mathsf{F})$ with probability at most $(q_{\mathrm{f}} + 2q)/2^n$.

(b) By ¬(C-2b), $S \oplus K_4$ is a fresh value for the function F and hence $T \oplus \mathsf{F}(S \oplus K_4) \oplus K_3 \in \mathsf{Domain}(\mathsf{F}) \cup \mathsf{Domain}'(\mathsf{F})$ with probability at most $(q_{\mathrm{f}} + 2q)/2^{\mathsf{n}}$.

By summing over all encryption queries, condition (C'-2) happens with probability at most $2q(q_{\mathrm{f}} + 2q)/2^{\mathsf{n}}$.

(C'-3) Fix $(+, \phi, ST) \neq (+, \phi', S'T') \in \mathcal{Q}_{\mathrm{BC}}$.

(a) By ¬(C-1a), we have $\phi_R(\mathbf{k}) \oplus K_1 \notin \mathsf{Domain}(\mathsf{F})$ and $\phi'_R(\mathbf{k}) \oplus K_1 \notin \mathsf{Domain}(\mathsf{F})$. Moreover, by ¬(C-3a), we have that $\phi(\mathbf{k}) \neq \phi'(\mathbf{k})$. We distinguish two cases. If $\phi_R(\mathbf{k}) \neq \phi'_R(\mathbf{k})$, then $\mathsf{F}(\phi_R(\mathbf{k}) \oplus K_1)$ and $\mathsf{F}(\phi'_R(\mathbf{k}) \oplus K_1)$ are uniformly random and independent, so that $\phi_L(\mathbf{k}) \oplus \mathsf{F}(\phi_R(\mathbf{k}) \oplus K_1) = \phi'_L(\mathbf{k}) \oplus \mathsf{F}(\phi'_R(\mathbf{k}) \oplus K_1)$ with probability $1/2^{\mathsf{n}}$. If $\phi_R(\mathbf{k}) = \phi'_R(\mathbf{k})$, then necessarily $\phi_L(\mathbf{k}) \neq \phi'_L(\mathbf{k})$, so that the condition cannot hold. Hence, this condition holds with probability at most $1/2^{\mathsf{n}}$.

(b) By ¬(C-1b), $S \oplus k_4 \notin \mathsf{Domain}(\mathsf{F})$ and $S' \oplus k_4 \notin \mathsf{Domain}(\mathsf{F})$; moreover, by ¬(C-3b), $S \neq S'$, so that $\mathsf{F}(S \oplus K_4)$ and $\mathsf{F}(S' \oplus K_4)$ are uniformly random and independent; hence, $T \oplus \mathsf{F}(S \oplus k_4) = T' \oplus \mathsf{F}(S' \oplus k_4)$ with probability at most $1/2^{\mathsf{n}}$.

By summing over all possible pairs, condition (C'-3) happens with probability at most $q^2/2^{\mathsf{n}}$.

(C'-4) Fix two (possibly equal) encryption queries $(+, \phi, ST), (+, \phi', S'T') \in \mathcal{Q}_{\mathrm{BC}}$. By ¬(C-1a), $\phi_R(\mathbf{k}) \oplus K_1 \notin \mathsf{Domain}(\mathsf{F})$ and by ¬(C-1b), $S' \oplus K_4 \notin \mathsf{Domain}(\mathsf{F})$; moreover, by ¬(C-4), we have $\phi_R(\mathbf{k}) \oplus K_1 \neq S' \oplus K_4$, so that $\mathsf{F}(\phi_R(\mathbf{k}) \oplus K_1)$ and $\mathsf{F}(S' \oplus K_4)$ are uniformly random and independent; hence, $\phi_L(\mathbf{k}) \oplus \mathsf{F}(\phi_R(\mathbf{k}) \oplus K_1) \oplus K_2 = T' \oplus \mathsf{F}(S' \oplus k_4) \oplus K_3$ with probability at most $1/2^{\mathsf{n}}$. By summing over all possible pairs, condition (C'-4) happens with probability at most $q^2/2^{\mathsf{n}}$.

(C'-5) Fix two decryption queries $(-, LR, ST) \neq (-, L'R', S'T') \in \mathcal{Q}_{\mathrm{BC}}$.

(a) By ¬(C-2a), $R \oplus k_1 \notin \mathsf{Domain}(\mathsf{F})$ and $R' \oplus k_1 \notin \mathsf{Domain}(\mathsf{F})$; moreover, by ¬(C-5a), $R \neq R'$ so that $\mathsf{F}(R \oplus K_1)$ and $\mathsf{F}(R' \oplus K_1)$ are uniformly random and independent. Hence, $L \oplus \mathsf{F}(R \oplus K_1) = L' \oplus \mathsf{F}(R' \oplus K_1)$ with probability at most $1/2^{\mathsf{n}}$.

(b) By ¬(C-2b), $S \oplus k_4 \notin \mathsf{Domain}(\mathsf{F})$ and $S' \oplus k_4 \notin \mathsf{Domain}(F)$. We distinguish two cases. If $S \neq S'$ then $\mathsf{F}(S \oplus K_4)$ and $\mathsf{F}(S' \oplus K_4)$ are uniformly random and independent and hence $T \oplus F(S \oplus K_4) = T' \oplus F(S' \oplus K_4)$ with probability at most $1/2^{\mathsf{n}}$. If $S = S'$ then necessarily $T \neq T'$ since the adversary does not repeat queries and hence the condition cannot hold. In all cases, the conditions holds with probability at most $1/2^n$.

By summing over all possible distinct pairs, condition (C'-5) happens with probability at most $q^2/2^{\mathsf{n}}$.

(C'-6) Fix two (possibly equal) decryption queries $(-, LR, ST), (-, L'R', S'T') \in \mathcal{Q}_{\mathrm{BC}}$. By ¬(C-2a), $R \oplus k_1 \notin \mathsf{Domain}(\mathsf{F})$ and by ¬(C-2b), $S' \oplus k_4 \notin \mathsf{Domain}(\mathsf{F})$; moreover, by ¬(C-6), $R \oplus K_1 \neq S' \oplus K_4$ so that $\mathsf{F}(R \oplus K_1)$ and $\mathsf{F}(S' \oplus K_4)$ are uniformly random and independent; hence, $L \oplus \mathsf{F}(R \oplus K_1) \oplus K_2 = T' \oplus \mathsf{F}(S' \oplus K_4) \oplus K_3$ with probability at most $1/2^{\mathsf{n}}$. By summing over all possible pairs, condition (C'-6) happens with probability at most $q^2/2^{\mathsf{n}}$.

(C'-7) Fix an encryption query $(+, \phi, ST) \in \mathcal{Q}_{\mathrm{BC}}$ and a decryption query $(-, L'R', S'T') \in \mathcal{Q}_{\mathrm{BC}}$. By respectively ¬(C-1a), ¬(C-1b), ¬(C-2a), and ¬(C-2b), $\phi_R(\mathbf{k}) \oplus K_1$, $S \oplus K_4$, $R' \oplus K_1$, and $S' \oplus K_4$ are all fresh input values to F.

(a) By ¬(C-7a), $\phi(\mathbf{k}) \neq L'R'$. We distinguish two cases. If $\phi_R(\mathbf{k}) \neq R'$, then $\mathsf{F}(\phi_R(\mathbf{k}) \oplus K_1)$ and $\mathsf{F}(R' \oplus K_1)$ are uniformly random and independent and thus $\phi_L(\mathbf{k}) \oplus \mathsf{F}(\phi_R(\mathbf{k}) \oplus K_1) = L' \oplus \mathsf{F}(R' \oplus K_1)$ with probability at most $1/2^{\mathsf{n}}$. If $\phi_R(\mathbf{k}) = R'$, then necessarily $\phi_L(\mathbf{k}) \neq L'$ and hence the condition cannot hold. In all cases, the condition holds with probability at most $1/2^n$.

(b) By ¬(C-7b), $ST \neq S'T'$. If $S \neq S'$, then $\mathsf{F}(S \oplus K_4)$ and $\mathsf{F}(S' \oplus K_4)$ are uniformly random and independent and hence $T \oplus \mathsf{F}(S \oplus K_4) = T' \oplus \mathsf{F}(S' \oplus K_4)$ with probability at most $1/2^{\mathsf{n}}$. If $S = S'$, then necessarily $T \neq T'$ and the condition cannot hold. In all cases, the condition holds with probability at most $1/2^n$.

(c) By ¬(C-7c), $\phi_R(\mathbf{k}) \oplus K_1 \neq S' \oplus k_4$ so that $\mathsf{F}(\phi_R(\mathbf{k}) \oplus K_1)$ and $\mathsf{F}(S' \oplus k_4)$ are uniformly random and independent and thus $\phi_L(\mathbf{k}) \oplus F(\phi_R(\mathbf{k}) \oplus K_1) \oplus K_2 = T' \oplus F(S' \oplus k_4) \oplus k_3$ with probability at most $1/2^{\mathsf{n}}$.

(d) By ¬(C-7d), $S \oplus K_4 \neq R' \oplus K_1$ so that $\mathsf{F}(S \oplus K_4)$ and $\mathsf{F}(R' \oplus K_1)$ are uniformly random and independent and thus $T \oplus \mathsf{F}(S \oplus K_4) \oplus K_3 = L' \oplus \mathsf{F}(R' \oplus K_1) \oplus K_2$ with probability at most $1/2^{\mathsf{n}}$.

By summing over all possible pairs, condition (C'-7) happens with probability at most $3q^2/2^{\mathsf{n}}$.

The result follows by the union bound over all conditions. □

Lemma 4.4.8. *Fix a good transcript $\tau = (\mathcal{Q}_{\mathrm{BC}}, \mathcal{Q}_F, \mathbf{k})$. Then*

$$\frac{\Pr[X_{\mathcal{A},\mathsf{rw}} = \tau]}{\Pr[X_{\mathcal{A},\mathsf{iw}} = \tau]} \geq 1 - 4 \cdot qq_f/2^{\mathsf{n}} + 7 \cdot q^2/2^{\mathsf{n}}.$$

Proof. Let $\tau = (\mathcal{Q}_{\mathrm{BC}}, \mathcal{Q}_{\mathsf{F}}, \mathbf{k})$ with $\mathbf{k} = (k_1, k_2, k_3, k_4)$ be a good transcript, and let q_{enc}, resp. q_{dec} denote the number of queries to KDMEnc, resp. Dec in $\mathcal{Q}_{\mathrm{BC}}$, with $q_{enc} + q_{\mathrm{dec}} = q$.

Recall that one has

$$\Pr[X_{\mathcal{A},\mathsf{pw}} = \tau] = \frac{1}{\#\mathcal{K}} \cdot \frac{1}{(2^{2\mathsf{n}})_{q_{enc}}} \cdot \frac{1}{(2^{2\mathsf{n}})_{q_{\mathrm{dec}}}} \cdot \frac{1}{(2^{\mathsf{n}})^{q_{\mathrm{f}}}}, \tag{4.2}$$

where $\mathcal{K} \coloneqq (\{0,1\}^{\mathsf{k}})^4$ denote the key space of the construction.

We must now lower bound the probability that $X_{\mathcal{A},\mathsf{rw}} = \tau$. One has

$$\begin{aligned}
\Pr[X_{\mathcal{A},\mathsf{rw}} = \tau] &= \frac{1}{\#\mathcal{K}} \cdot \Pr[\mathsf{KAF}^{\mathsf{F}}_{\mathbf{k}} \vdash \mathcal{Q}_{\mathrm{BC}} \wedge \mathsf{F} \vdash \mathcal{Q}_{\mathsf{F}}] \\
&= \frac{1}{\#\mathcal{K}} \cdot \frac{1}{(2^{\mathsf{n}})^{q_{\mathrm{f}}}} \cdot \Pr[\mathsf{KAF}^{F}_{\mathbf{k}} \vdash \mathcal{Q}_{\mathrm{BC}} \,|\, \mathsf{F} \vdash \mathcal{Q}_{\mathsf{F}}] \\
&= \frac{1}{\#\mathcal{K}} \cdot \frac{1}{(2^{\mathsf{n}})^{q_{\mathrm{f}}}} \cdot \Pr[\mathsf{KAF}^{\mathsf{F}}_{\mathbf{k}} \vdash \mathcal{Q}_{\mathrm{BC}} \,|\, \mathsf{F} \vdash \mathcal{Q}_{\mathsf{F}} \wedge \neg\mathsf{bad}(\mathsf{F}, \tau)] \\
&\qquad \cdot (1 - \Pr[\mathsf{bad}(\mathsf{F}, \tau) \,|\, \mathsf{F} \vdash \mathcal{Q}_{\mathsf{F}}]),
\end{aligned} \tag{4.3}$$

where all probabilities are over $\mathsf{F} \twoheadleftarrow \mathcal{F}^*(\{0,1\}^{\mathsf{n}}, \{0,1\}^{\mathsf{n}})$.

Combining Equation (4.2) and Equation (4.3), one has

$$\begin{aligned}
\frac{\Pr[X_{\mathcal{A},\mathsf{rw}} = \tau]}{\Pr[X_{\mathcal{A},\mathsf{pw}} = \tau]} &= (2^{2\mathsf{n}})_{q_{enc}} \cdot (2^{2n})_{q_{\mathrm{dec}}} \cdot \Pr[\mathsf{KAF}^{\mathsf{F}}_{\mathbf{k}} \vdash \mathcal{Q}_{\mathrm{BC}} \,|\, \mathsf{F} \vdash \mathcal{Q}_{\mathsf{F}} \wedge \neg\mathsf{bad}(\mathsf{F}, \tau)] \\
&\qquad \cdot (1 - \Pr[\mathsf{bad}(\mathsf{F}, \tau) \,|\, \mathsf{F} \vdash \mathcal{Q}_{\mathsf{F}}]).
\end{aligned}$$

Using Theorem 4.4.6 and Theorem 4.4.7, we obtain

$$\begin{aligned}
\frac{\Pr[X_{\mathcal{A},\mathsf{rw}} = \tau]}{\Pr[X_{\mathcal{A},\mathsf{pw}} = \tau]} &\geq \frac{(2^{2\mathsf{n}})_{q_{enc}} \cdot (2^{2\mathsf{n}})_{q_{\mathrm{dec}}}}{(2^{\mathsf{n}})^{2q}} \cdot (1 - 4 \cdot qq_{\mathrm{f}}/2^{\mathsf{n}} + 7 \cdot q^2/2^{\mathsf{n}}) \\
&\geq 1 - 4 \cdot qq_{\mathrm{f}}/2^{\mathsf{n}} + 7 \cdot q^2/2^{\mathsf{n}}.
\end{aligned}$$

□

Attack on 4 rounds KAF. If the KDM set Φ is not Offset-free then the following adversary breaks the KDM security of a 4-rounds $\mathsf{KAF}^{\mathsf{F}}_{\mathbf{k}}$ with the same round functions F and independent keys such that $\mathbf{k} = (K_1, K_2, K_3, K_4)$. This attack take advantage of a collision at the input if the third function F with two different encryption queries.

- $\mathcal{A}$ chooses two values x and x' then it uses the oracle $\mathcal{O}$ (it implements the function F) to obtain the values $\mathsf{F}(x)$, $\mathsf{F}(x')$, $\mathsf{F}^2(x)$ and $\mathsf{F}^2(x')$.

- $\mathcal{A}$ builds the values $\Delta_L = \mathsf{F}^2(x) \oplus x$; $\Delta_R = \mathsf{F}(x)$, $\Delta'_L = \mathsf{F}^2(x') \oplus x'$ and $\Delta'_R = \mathsf{F}(x')$ and calls the KDMEnc oracle twice with the inputs ϕ and ϕ' such that $\phi_L(\mathbf{k}) = K_2 \oplus \Delta_L$, $\phi_R(\mathbf{k}) = K_1 \oplus \Delta_R$, $\phi'_L(\mathbf{k}) = K_2 \oplus \Delta'_L$ and $\phi'_R(\mathbf{k}) = K_1 \oplus \Delta'_R$. It receives the values ST and $S'T'$.

- If $S \oplus S' = x \oplus x'$, $\mathcal{A}$ returns 1 and it returns 0 otherwise.

4.4.2 Sliding attack for r-rounds

In this section, we analyse the simplest KAF configuration where all functions are the same and all keys are identical. This construction, for any number of rounds, is already insecure in the CPA model: using two encryption queries, we have $\mathsf{KAF}(LR) = ST$ and $\mathsf{KAF}(TS) = LR$ (the adversary can recover the plaintext LR). In the KDM security model, we give a stronger attack, namely a key-recovery attack, using only one query. Four the 4-rounds KAF, the adversary $\mathcal{A}$ is the following:

- $\mathcal{A}$ chooses a value $\Delta \in \{0,1\}^{\mathsf{n}}$ and calls the encryption oracle KDMEnc giving as input the function ϕ such that

  $\phi_R(\mathbf{k}) = K \oplus \mathsf{F}(\Delta) \oplus \mathsf{F}[\mathsf{F}^2(\Delta) \oplus \Delta]$ and $\phi_L(\mathbf{k}) = K \oplus \mathsf{F}^2(\Delta) \oplus \Delta \oplus \mathsf{F}[\mathsf{F}(\Delta) \oplus \mathsf{F}(\mathsf{F}^2(\Delta) \oplus \Delta)]$.

  It receives the value ST.

- If $S \oplus T = \Delta$ then $\mathcal{A}$ returns the value T (which is the key K in the real world).

For this attack, one encryption query is sufficient to extract the key K. The details of this attack is shown in Figure 4.17. This attack can be generalized for any number of rounds. We can give a generic expression for the function ϕ for any number of round r but we believe that it will be hard to read. Instead, we give the following method. The idea is that, for a r-rounds KAF, the outputs ST has to be equal to $S = L_0$ and $T = R_0$ where $L_0 = K \oplus \Delta$ and $R_0 = K$. We define the following recursive sequences

$$\forall i \in \mathbb{N},\ (LR)_i := \begin{cases} L_0 = K \oplus \Delta \text{ and } R_0 = K \\ L_{i+1} = \mathsf{F}(L_i \oplus K) \oplus R_i \\ R_{i+1} = L_i \end{cases}$$

$\forall i \in \mathbb{N}$, the values L_i and R_i can be written such that $L_i = K \oplus \Delta_L$ and $R_i = K \oplus \Delta_R$ where Δ_L and Δ_R are constants independent from K then choosing such functions (L_i and R_i) as inputs is allowed. Then the previous adversary $\mathcal{A}$ can be used to recover the key in the CPA-KDM model of the r-round KAF (same round functions and same keys) with $\phi_L(K) = L_r$ and $\phi_R(K) = R_r$. The outputs ST will always be the same (equal to L_0R_0 as shown in Figure 4.18).

Even in the r round KAF, where the keys of all odd rounds are equal to the same key K_1 and all the keys of even rounds are equal to the same K_2 the keys K_1 and K_2 can be extracted using one query in the KDM model. The previous attack can be adapted by choosing $L_0 = K_2 \oplus \Delta$ and $R_0 = K_1$ when r is even and $L_0 = K_1 \oplus \Delta$ and $R_0 = K_2$ when r is odd.

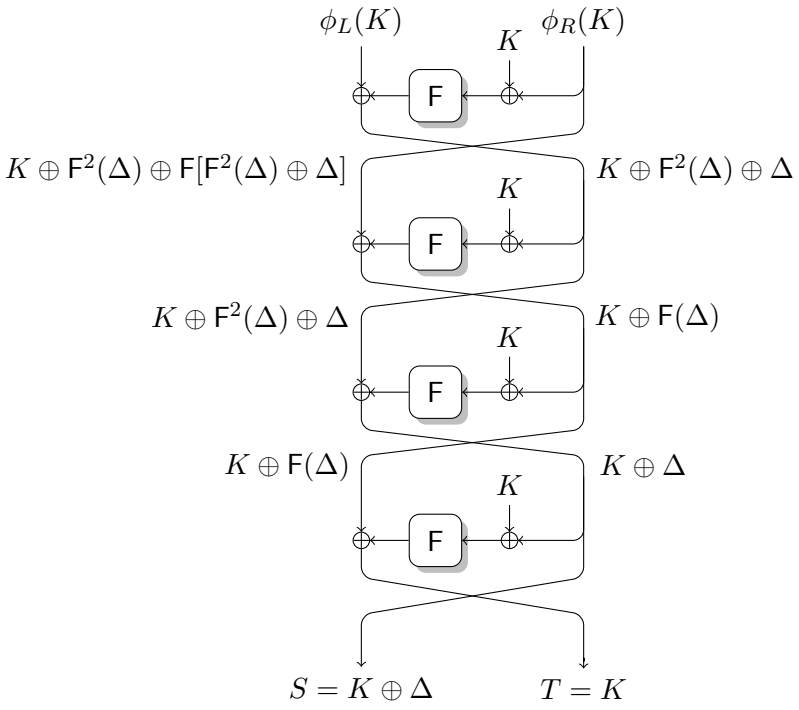

Figure 4.17: Details of the Sliding attack on 4-rounds KAF (same keys and same round function).

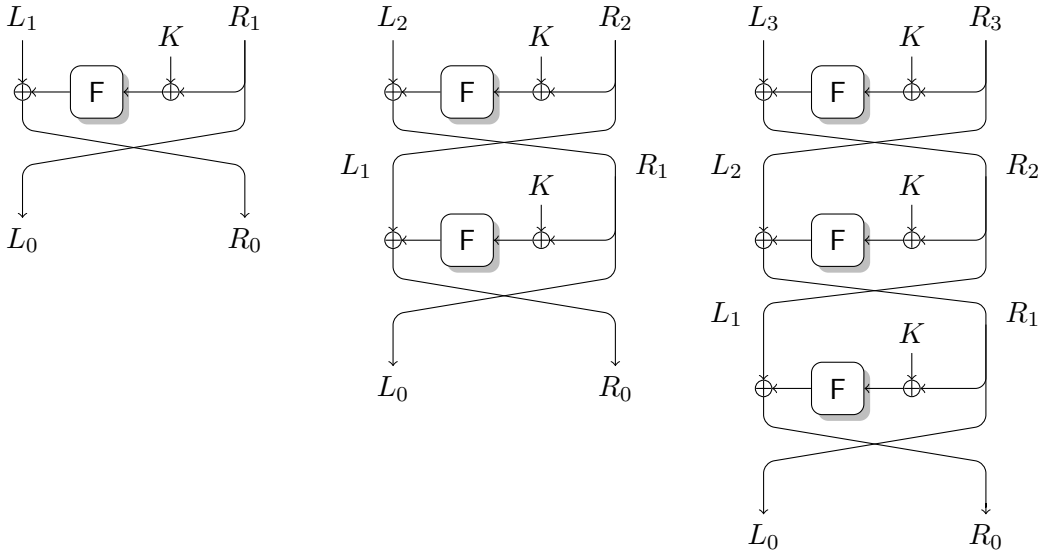

Figure 4.18: Notations for Sliding attack.

To be more formal, we have the following recursive sequences:

$$\forall i = 2j \text{ with } j \in \mathbb{N},\ (LR)_i \coloneqq \begin{cases} L_0 = K_1 \oplus \Delta \text{ and } R_0 = K_2 \\ L_{i+1} = \mathsf{F}(L_i \oplus K^*) \oplus R_i \\ R_{i+1} = L_i \end{cases}$$

When computing the internal value L_{i+1}, the key K^* is equal to K_1 if i is odd and it is equal to K_2 otherwise.

$$\forall i = 2j+1 \text{ with } j \in \mathbb{N},\ (LR)_i \coloneqq \begin{cases} L_0 = K_2 \oplus \Delta \text{ and } R_0 = K_1 \\ L_{i+1} = \mathsf{F}(L_i \oplus K^*) \oplus R_i \\ R_{i+1} = L_i \end{cases}$$

When computing the internal value L_{i+1}, the key K^* is equal to K_1 if i is even and it is equal to K_2 otherwise.

To recover the two keys, K_1 and K_2, the adversary computes $S \oplus \Delta$, which corresponds to one key (K_1 or K_2 depending of the number of rounds) and the other key is the output T.

4.5 Even-Mansour KDM security with H-coefficients

The KDM-security of the one-round Even-Mansour cipher is analysed in a first place in section 4.2.2.1 with the splitting and forgetting framework. To compare the two KDM security analysis methods, it was analysed also with the H-coefficient technique.

The r-round Even–Mansour cipher in a model of computation with r permutations $\mathsf{P}_1^{\pm}, \ldots, \mathsf{P}_r^{\pm}$ with domain $\mathcal{M} = \{0,1\}^n$ is a block cipher with key space $\mathcal{K} = \{0,1\}^{(r+1)n}$ and enciphering and deciphering algorithms

$$\mathsf{E}^{\mathsf{P}_1,\ldots,\mathsf{P}_r}((K_1,\ldots,K_{r+1}),M) := \mathsf{P}_r(\cdots \mathsf{P}_2(\mathsf{P}_1(M \oplus K_1) \oplus K_2)\cdots) \oplus K_{r+1}\ ,$$
$$\mathsf{D}^{\mathsf{P}_1^-,\ldots,\mathsf{P}_r^-}((K_1,\ldots,K_{r+1}),M) := \mathsf{P}_1^-(\cdots \mathsf{P}_{r-1}^-(\mathsf{P}_r^-(M \oplus K_{r+1}) \oplus K_r)\cdots) \oplus K_1\ .$$

The EM ciphers can be also considered in configurations where (some of the) keys and/or (some of the) permutations are reused in different rounds. We denote the EM cipher where P_i and K_{i+1} are used in round i by $\mathsf{EM}^{\mathsf{P}_1,\ldots,\mathsf{P}_r}[K_1, K_2, \ldots, K_{r+1}]$.

4.5.1 Security of 1-round Even-Mansour

Theorem 4.5.1. *Let Φ be a KDM set that is claw-free and offset-xor-free. Then $\mathsf{EM}^{\mathsf{P}}[\mathbf{k}]$, with $\mathbf{k} = (K_1, K_2)$ is Φ-KDM-secure.*

$$\mathbf{Adv}^{kdm-cca}_{\mathsf{EM}^{\mathsf{P}}_{K_1,K_2}}(\mathcal{A}) \leq 2q^2 \cdot \mathsf{cf}(\Phi) + qq_{\mathrm{f}} \cdot \mathsf{of}(\Phi) + 3/2 \cdot q^2/(2^{\mathsf{n}} - 1) + 3qq_{\mathrm{f}} \cdot 1/2^{\mathsf{n}}.$$

where q is the number of challenge queries of $\mathcal{A}$ in either direction and q_f is the number of queries of $\mathcal{A}$ to P.

Proof. By Equation (4.1) and Theorem 4.3.5, we have

$$\mathbf{Adv}^{kdm-cca}_{\mathsf{EM}^{\mathsf{P}}_{K_1,K_2}}(\mathcal{A}) \leq q^2 \mathrm{cf}(\Phi) + \frac{q^2}{2^n - q} + \mathbf{Adv}_{\mathsf{pw},\mathsf{rw}}(\mathcal{A}).$$

Applying H-coefficient technique between the perfect world and the real world with Theorem 4.5.3 and Theorem 4.5.4 gives

$$\mathbf{Adv}_{\mathsf{pw},\mathsf{rw}}(\mathcal{A}) \leq q^2 \cdot \mathsf{cf}(\Phi) + qq_{\mathrm{f}} \cdot \mathsf{of}(\Phi) + q^2/2 \cdot 1/(2^{\mathsf{n}} - q) + 3qq_{\mathrm{f}}/2^{\mathsf{n}}.$$

Then combining the two above in-equation gives the result. This analysis gives the same result with $K_1 = K_2$. □

Real *vs* Perfect. In the following, we give the analysis that gives the advantage of an adversary distinguishing the perfect world pw and the real world rw.

We define the bad transcripts for 1-round Even-Mansour as transcripts that contains queries where trivial collisions exist. By trivial collisions, we mean collisions that enable the adversary to distinguish the perfect world and the ideal world with high probability.

Definition 4.5.2. *A transcript* $\tau = (\mathcal{Q}_{\mathsf{EM}}, \mathcal{Q}_{\mathsf{F}}, \mathbf{k})$ *with* $\mathbf{k} = (K_1, K_2)$ *is said bad if one of the following condition hold*

(C-1) there exist $(+, \phi, y) \neq (+, \phi', y') \in \mathcal{Q}_{\mathsf{EM}}$ *such that* $\phi(\mathbf{k}) \neq \phi'(\mathbf{k})$.

(C-2) there exist $(+, \phi, y) \neq (-, x', y') \in \mathcal{Q}_{\mathsf{EM}}$ *such that*

- *(a)* $y = y'$ *or*
- *(b)* $\phi(\mathbf{k}) = x'$ *where* ϕ *different from the constant function* $k \mapsto x$.

(C-3) there exist $(+, \phi, y) \in \mathcal{Q}_{\mathsf{EM}}$ *and* $(u, v) \in \mathcal{Q}_{\mathsf{P}}$ *such that*

- *(a)* $\phi(\mathbf{k}) \oplus K_1 = u$ *or*
- *(b)* $y \oplus K_2 = v$.

(C-4) there exist $(-, x, y) \in \mathcal{Q}_{\mathsf{EM}}$ *and* $(u, v) \in \mathcal{Q}_{\mathsf{P}}$ *such that*

- *(a)* $x \oplus K_1 = u$ *or*
- *(b)* $y \oplus K_2 = v$.

Lemma 4.5.3. *For the settings* $\mathrm{BC} := \mathsf{EM}_{K_1,K_2}$ *and* $\mathcal{O} := \mathsf{P}$*, let* $\mathcal{A}$ *be a distinguisher making at most* $q \leq 2^n$ *queries to* KDMEnc *or* Dec *and* q_f *queries to* P*. With* $\mathcal{T}_{bad}$ *as defined above, one has*

$$\Pr[X_{\mathcal{A},\mathsf{pw}} \in \mathcal{T}_{bad}] \leq q^2 \cdot \mathsf{cf}(\Phi) + q^2/2 \cdot 1/(2^{\mathsf{n}} - q) + qq_f \cdot \mathsf{of}(\Phi) + 3qq_f/2^{\mathsf{n}}.$$

Proof. We analyse the probability of each condition given in Theorem 4.5.2 in the perfect world pw where the keys K_1 and K_2 are sampled at random independently from the oracle answers.

(C-1) For two encryption queries in $\mathcal{Q}_{\mathsf{EM}}$, by Theorem 4.3.1, the probability of condition (C-1) is at most $\mathsf{cf}(\Phi)$. Summing over all the possible distinct pairs, condition (C-1) happens wit probability at most $q^2/2 \cdot \mathsf{cf}(\Phi)$.

(C-2) Fix an encryption query $(+, \phi, y) \in \mathcal{Q}_{\mathsf{EM}}$ and a decryption query $(-, x', y') \in \mathcal{Q}_{\mathsf{EM}}$,

(a) The decryption query has to before the encryption query in the transcript otherwise the encryption query is a pointless query. Then the value y is sampled in a set of size $2^n - q$ independently from the value y'. Then the probability to have $\phi(\mathbf{k}) = x'$ is at most $1/(2^n - q)$.

(b) Here the function ϕ is different from the constant function $\mathbf{k} \mapsto x'$ otherwise, $(+, \phi, y) = (-, x', y') \in \mathcal{Q}_{\mathsf{EM}}$. As the key vector $\mathbf{k}$ is sampled uniformly at random after all encryption/decryption query, the probability to have $\phi(\mathbf{k}) = x'$ is $\mathsf{cf}(\Phi)$ by Theorem 4.3.1.

Summing over all the possible distinct pairs, condition (C-2) happens with probability at most $q^2/2 \cdot 1/(2^{\mathsf{n}} - q) + q^2/2 \cdot \mathsf{cf}(\Phi)$.

(C-3) Fix a query to the encryption oracle $(+, \phi, y) \in \mathcal{Q}_{\mathsf{EM}}$ and a query to the public permutation $(u, v) \in \mathcal{Q}_{\mathsf{P}}$.

(a) By Theorem 4.3.2, $\phi(\mathbf{k}) \oplus K_1 = u$ with probability at most $\mathsf{of}(\Phi)$.

(b) As the key K_2 is sampled uniformly at random and independently from all the queries, the collision $y \oplus K_2 = v$ happens with probability at most $1/2^n$. Summing over all the pairs gives $qq_{\mathrm{f}}/2^n$.

Summing over all the possible pairs, condition (C-3) happens with at most probability $qq_{\mathrm{f}} \cdot \mathsf{of}(\Phi) + qq_{\mathrm{f}}/2^{\mathsf{n}}$.

(C-4) Fix a query to the decryption oracle $(-, x, y) \in \mathcal{Q}_{\mathsf{EM}}$ and a query to the public permutation $(u, v) \in \mathcal{Q}_{\mathsf{P}}$. The keys K_1 and K_2 are uniformly at random and are independent of all the queries, each collision happens with probability at most $1/2^{\mathsf{n}}$. By summing over all the possible pairs, condition (C-4) happens with probability at most $2qq_{\mathrm{f}}/2^{\mathsf{n}}$.

The result follows by the union bound. □

Lemma 4.5.4. *Let τ be a good transcript. Then*

$$\frac{\Pr[X_{\mathcal{A},\mathsf{rw}} = \tau]}{\Pr[X_{\mathcal{A},\mathsf{pw}} = \tau]} \geq 1.$$

Proof. Let $\tau = (\mathcal{Q}_{\mathsf{EM}}, \mathcal{Q}_{\mathsf{P}}, \mathbf{k})$ with $\mathbf{k} = (K_1, K_2)$ be a good transcript, and let q_{enc}, resp. q_{dec} denote the number of queries to KDMEnc, resp. Dec in $\mathcal{Q}_{\mathsf{EM}}$, with $q_{enc} + q_{\mathrm{dec}} = q$.
Recall that one has

$$\Pr[X_{\mathcal{A},\mathsf{pw}} = \tau] = \frac{1}{|\mathcal{K}|} \cdot \frac{1}{(2^{\mathsf{n}})_{q_{enc}}} \cdot \frac{1}{(2^{\mathsf{n}})_{q_{\mathrm{dec}}}} \cdot \frac{1}{(2^{\mathsf{n}})_{q_{\mathrm{f}}}}, \tag{4.4}$$

where $\mathcal{K} \coloneqq (\{0,1\}^n)^2$ denotes the key space of the 1-round Even-Mansour scheme.
In the real world, one obtains the queries transcript $(\mathcal{Q}_{\mathsf{EM}}, \mathcal{Q}_{\mathsf{P}})$ *iff* P satisfies $q + q_{\mathrm{f}}$ distinct and "compatible" equations and

$$\Pr[X_{\mathcal{A},\mathsf{rw}} = \tau] = \frac{1}{|\mathcal{K}|} \cdot \frac{1}{(2^{\mathsf{n}})_{q+q_{\mathrm{f}}}}, \tag{4.5}$$

where all probabilities are over $\mathsf{P} \twoheadleftarrow \mathcal{P}^*(\{0,1\}^n)$.
Combining Equation (4.4) and Equation (4.5) gives the result. □

Conclusion. This analysis of the one-round Even-Mansour cipher, using the H-coefficient technique, gives the same security result than the the splitting and forgetting framework (the advantages are very close). Moreover, this analysis, using H-coefficient technique, is shorter and more readable but the intuition of the security proof is more hidden. We think that if the reader is familiar with the H-coefficient technique, this second technique is easier to apply. It can be interesting to prove the KDM-security for different schemes and configurations (EM, KAF or others) with both techniques to see if for some cases, one of these methods gives a better result.

Generally, this analysis shows that KDM security under a claw-free KDM set is not achievable for the simple EM and KAF configurations, where the keys are equal and the internal primitives (functions or permutations) are the same; and no matter the number of rounds due to the slide attacks. But some level of KDM security, for a more constrained KDM set (offset-free and/or offset-xor-free), is achievable when the underlying primitives of the EM/KAF constructions are different or the keys are independent. The most interesting configurations, from an implementation point of view, is where the internal primitives are the same; it is implemented once and can be used many times. We noticed that, when the keys are independent, the KDM security level increases: the one-round Even-Mansour is is proved secure under a KDM set that is offset-free, offset-xor-free and claw-free; the KDM set for the two-round Even-Mansour is expanded by removing the offset-freeness restriction and we believe that adding one round will give KDM security with a claw-free set (for 3-round EM). For the KAF, with the same configuration, giving a conjecture is less straightforward but we believe that 6-round KAF cipher (same functions and independent keys) should be secure under a claw-free KDM set.

Chapter 5

Incremental Authentication Schemes

Contents

5.1 Incremental MACs and Security notions 111
5.1.1 Incremental Authentication Scheme Framework 111
5.1.2 Security Notions for Incremental MACs 115
5.1.3 Relations among Security Notions for Incremental MACs 118
5.1.4 From Single-Document to Multi-Document Security 122
5.2 Incremental MACs with IUF1 Security 127
5.2.1 XMAC Constructions 127
5.2.2 XS Construction 128
5.2.3 MXS Construction 135
5.3 Incremental MACs with IUF2 Security 142
5.3.1 XMAC Constructions 142
5.3.2 MXS Construction 143

Bellare, Goldreich and Goldwasser initiated the study on *incremental cryptography* in [BGG94] and then refined it in [BGG95]. Cryptographic incremental constructions are meant to provide efficient updates compared to classical algorithms. Usually, the result of a cryptographic algorithm (such as encryption or authentication) over a document has to be re-computed entirely if any change is applied to the document (and this regardless of the modification size). Incremental cryptography enables to update a signature, a message authentication code (MAC) or a ciphertext in time proportional to the number of modifications applied to the corresponding document. This attractive feature leads to build many incremental cryptographic primitives such as encryption schemes [BGG95, AM13, Ati14], signatures [BGG94, Fis97b, MPRS12], MACs [BGG95, Fis97b, Mic97], hash functions [GSC01, BM97] and authenticated encryption constructions [BKY02, AM13, SY16].

An algorithm is incremental regarding specific *update* operations such as inserting, deleting or replacing a data block inside a document. A desirable incremental algorithm should support all these operations for *any* position: it should be possible to insert, delete or replace a data block of the document for all positions without breaking the security of the cryptographic algorithm. Most known algorithms only support replacement of data blocks and the algorithms that support insertion, deletion and replacement[1] are deemed *strongly incremental*.

Protection against virus is the first application of incremental cryptography quoted in the seminal paper [BGG94]. They consider the usage scenario where a processor accesses files on a remote host and a virus can alter these files. A simple idea is to compute authentication tags for all files with a key stored securely by the processor and any modification by a virus will be detected by verifying the corresponding tag. Knowing that these files will be updated often enough, using an incremental authentication algorithm preserves the processor by performing a lighter computation.

Bellare *et al.* also introduced in [BGG94] the corresponding security notions. In the *basic security* model, the adversary can obtain a valid authentication tag for any message he wanted (as in classical MAC security) and it can also update (with the supported update operations) valid message/tag pairs. This is a first security level but it is reasonable to consider a stronger adversary that can alter files *and* tags before applying update operations; it corresponds to the *tamper-proof security* notion introduced in [BGG94].

Nowadays, this use case can be extended to the "digital world". Large amount of data [GR12, MGS15] are processed every day by different services like cloud services, distributed networks and distributed storage. It is clear that all these data require integrity and/or privacy at a low computational cost otherwise going through gigabytes of data for minor changes without incremental primitives is really demanding in term of time and energy. A concrete example is the Cloud Bigtable by Google [BM06] that stores petabytes of data across thousands of commodity servers. This Bigtable has a particular data structure that links a unique index number to each block. In this case an incremental hashing that supports replacement and insertion operations is suitable as mentioned in [MGS15]. A more critical usage is storage services in mobile cloud computing where a mobile client device is in addition limited in term of energy consumption. To solve this issue, Itani, Kayssi and Chehab provide an energy-efficient protocol in [IKC10] that guarantees data integrity based on incremental MACs. Another use case is sensor networks and more specifically environmental sensors [HM06, MGS15]: several

[1] Actually supporting insertion and deletion is sufficient as replacement can be obtain by combining these two update operations.

sensors are deployed at different physical positions and they record continuously data. At some point, all the data end up in a big public database that has to be publicly checkable. The database is updated (insertion operation mainly) at a high frequency and if the hash value over all the database is entirely re-computed for each insertion it will be very consuming. All these use cases are examples among many others. Incremental cryptography is clearly an area to explore to solve practical issues. For now incrementality is mainly investigated for hashing and signing even if it was also considered for encryption in [BGG94, AM13]. It is not surprising regarding all the practical use cases that need incremental authenticated constructions. Recently, the CAESAR[2] competition stimulated research on authenticated encryption algorithms. Sasaki and Yasuda analysed several candidates and found that none of them performed incrementality. That is why they designed their own authenticated encryption mode with associated data [SY16] based on existing constructions. This new mode is incremental for the replace, insert and delete operations, but the insert and delete operations of this mode concern only the last block of the authenticated data or the last block of the message (and it remains open to design a strongly incremental authenticated encryption algorithm).

In chapter 2, we define a strong model where the authenticity of the disk sectors is guaranteed. Incremental MAC can be a possibility to enhance the disk performance by reducing the read and write operation delays as the update and verify operations should be done in constant time. A first scenario is to use a regular MAC scheme to compute the local tags and an incremental MAC to ensure the authenticity of the local tags. Then for each data sector modification (e.g. each write operation), the corresponding local tag is recomputed from scratch and the global tag is updated *quickly* in the secure memory according to the local tag. And each time, a sector is read, the authenticity of the local tag has to be verified *quickly* (using the incremental MAC) before the verification of the data sector integrity (using the MAC). Another possibility is to use the incremental MAC for the data sector authenticity with a key K_1 and for the local tag integrity with a different key K_2. But for each sector modification, even only one bit, the sector is re-written entirely then if the local tag computation with a regular MAC takes less time than a read/write operation (without data authenticity) then using an incremental MAC for the sector data does not enhance the read or write operations. As the number of sectors is fixed and it is better to guarantee the whole disk integrity, the incremental MAC set of operations can be reduced to the *replace* operation. The initialisation of the disk will take some time due to the encryption and the local tag computation of each sector and the global tag in the FADE model (described in Section 2.3.2) but this is only the case for the first disk usage.

One can also consider encryption "on the fly" where each sector are encrypted and tagged only when it is used for the first time. In this chapter, we introduce a specific framework to analyse incremental MACs for different security notions. Then we give an attack against the the basic security of the Xor-Scheme which is a strongly incremental MAC. We also give different ways to patch this construction and a security proof for one of them. Then we introduce a new incremental MAC scheme that achieves the strong security notion IUF2 and the corresponding security proofs.

[2]Competition for Authenticated Encryption: Security, Applicability, and Robustness.

5.1 Incremental MACs and Security notions

In this section, we give a formal definition for an incremental MAC scheme and the corresponding security notions using security games. We recall that the game rules and the game notations are described in Section 1.2. We follow the line of papers oriented on incremental MACs (like in [Fis97a]) by using the word "document" instead of message but both can be used.

5.1.1 Incremental Authentication Scheme Framework

Document editing systems. A document editing system DES [3] specified as follows $\mathsf{DES} := (\mathsf{BS}, \mathsf{DS}, \mathsf{OpCodes}, \mathsf{Edit})$. There is a block space $\mathsf{DES.BS} = \{0,1\}^{\mathsf{DES.n}}$ where $\mathsf{DES.n} \geq 1$ is the block length. The document space $\mathsf{DES.DS}$ is then defined as the set of all vectors over $\mathsf{DES.BS}$, meaning a document has the form $D = D[1]||\ldots||D[\mathsf{nb}]$ with $\mathsf{nb} = |D|_{\mathsf{DES.n}}$ (number of $\mathsf{DES.n}$-bit blocks in D) and $D[i] \in \mathsf{DES.BS}$ for all $i \in [1..\mathsf{nb}]$. There is a set $\mathsf{DES.OpCodes}$ of operation codes, which are names or formal symbols to indicate edit operations on documents. There is a deterministic algorithm $\mathsf{DES.Edit}$ which takes, as inputs, a document $D \in \mathsf{DES.DS}$, an operation code $op \in \mathsf{DES.OpCodes}$ and arguments arg and returns an updated document $D' \in \mathsf{DES.DS}$.

op	arg	$\mathsf{DES.Edit}(D, op, arg)$
replace	i, x	$(D[1], \ldots, D[i-1], x, D[i+1], \ldots, D[\lvert D\rvert])$
insert	i, x	$(D[1], \ldots, D[i], x, D[i+1], \ldots, D[\lvert D\rvert])$
delete	i	$(D[1], \ldots, D[i-1], D[i+1], \ldots, D[\lvert D\rvert])$

Figure 5.1: Examples of edit operations.

Examples of operations are given in Figure 5.1. The figure shows three common edit operations, namely insert, replace and delete, whose operation codes are denoted respectively by *insert*, *replace* and *delete*. The *insert* operation allows inserting a block x at position i in the document D, the *delete* operation allows deletion of the i-th block of D, and the *replace* operation allows replacement of the i-th block of D by the block x. Note that a scheme which is incremental for the *insert* and *delete* operations is also incremental for the *replace* operation: *replace* can be implemented by using *insert* and *delete*. But in some cases we will consider only *replace*.

Incremental message authentication schemes. In the following, we define two types of incremental schemes: probabilistic or randomized and nonce-based incremental schemes.

A nonce-based incremental message authentication scheme iMA for a document editing system DES is specified as follows:

$$\mathsf{iMA} = (\mathsf{KS}, \mathsf{BS}, \mathsf{DS}, \mathsf{NS}, \mathsf{IS}, \mathsf{kg}, \mathsf{init}, \mathsf{tag}, \mathsf{upd}, \mathsf{ver}).$$

There is a block space $\mathsf{iMA.BS} = \{0,1\}^{\mathsf{iMA.n}}$ where $\mathsf{iMA.n} \geq 1$ is the block length; the document space $\mathsf{iMA.DS}$ is then defined as the set of all vectors over $\mathsf{iMA.BS}$ and $\mathsf{iMA.DS} \supseteq \mathsf{DES.DS}$,

[3] In the following DES refers to this notion and not to the Data Encryption Standard.

so that all documents of the document editing system may be tagged. There are a document identity space iMA.IS, a nonce space iMA.NS and a key space iMA.KS defined for the scheme.

- The iMA scheme has to be initialized by calling the probabilistic *key generation algorithm* iMA.kg that takes no inputs and returns a key $K \in$ iMA.KS.
- The *initialization algorithm* iMA.init takes an identifier $id \in$ iMA.IS and returns an initial state st_{id} for id. It can be probabilistic.
- The *tagging algorithm* iMA.tag takes a key $K \in$ iMA.KS, a nonce $N \in$ iMA.NS, an identifier $id \in$ iMA.IS, a document $D \in$ iMA.DS and current state st_{id}, and deterministically returns a tag t and updated state st_{id}.
- There is an *update algorithm* iMA.upd that takes the key $K \in$ iMA.KS, a nonce $N \in$ NS, an identifier $id \in$ iMA.IS, a document $D \in$ iMA.DS, an operation code $op \in$ DES.OpCodes, arguments arg for the operation, a tag t and the current state st_{id} and deterministically returns an updated tag t and updated state st_{id}.
- The *verification algorithm* iMA.ver takes a key $K \in$ iMA.KS, an identifier $id \in$ iMA.IS, a document $D \in$ iMA.DS, the state st_{id} and a candidate tag t to deterministically return either true or false.
- An iMA scheme may take other parameters and if it is the case, it will be explicitly underlined in its specifications.

For nonce-based incremental schemes, the tagging, update and verify algorithms are deterministic.

A probabilistic incremental scheme iMA such that

$$\mathsf{iMA} = (\mathsf{KS}, \mathsf{BS}, \mathsf{DS}, \mathsf{RS}, \mathsf{IS}, \mathsf{kg}, \mathsf{init}, \mathsf{tag}, \mathsf{upd}, \mathsf{ver}),$$

has similar specifications except that

1. Instead of a nonce space NS, it has as a parameter a random value space RS,
2. The tagging and update algorithms are probabilistic and do not take a nonce as input.

Update time. The update time of an incremental MAC should be proportional to the modification. This property makes an incremental MAC more efficient that a standard MAC where for each modification, the tag is recomputed from scratch. More formally, for a couple document/tag where the document is composed by nb blocks, if the size of the modification in the document is m blocks, the update operation should be done in time $O(m)$. For the Xor-MAC, the replace operation for one document block costs 2 evaluations of the underlying PRF which makes it incremental but the replace operation of one document block for the Merkle tree costs $O(m \log_2 \mathsf{nb})$ evaluations of the underlying PRF. However, the incrementality of the tag verification is not required.

In the following, the security games are written regarding nonce-based incremental authentication schemes. The corresponding games for probabilistic schemes are similar except that (1) for all procedures, the nonces are ignored (the corresponding nonce-space NS to); (2) the internal iMA.tag and iMA.upd are probabilistic. For instance, the

code line $(t_{id}, st_{id}) \leftarrow \mathsf{iMA.tag}(K, N, id, D, st_{id})$ in the nonce-based setting is replaced by $(t_{id}, st_{id}) \twoheadleftarrow \mathsf{iMA.tag}(K, id, D, st_{id})$ in the probabilistic setting.

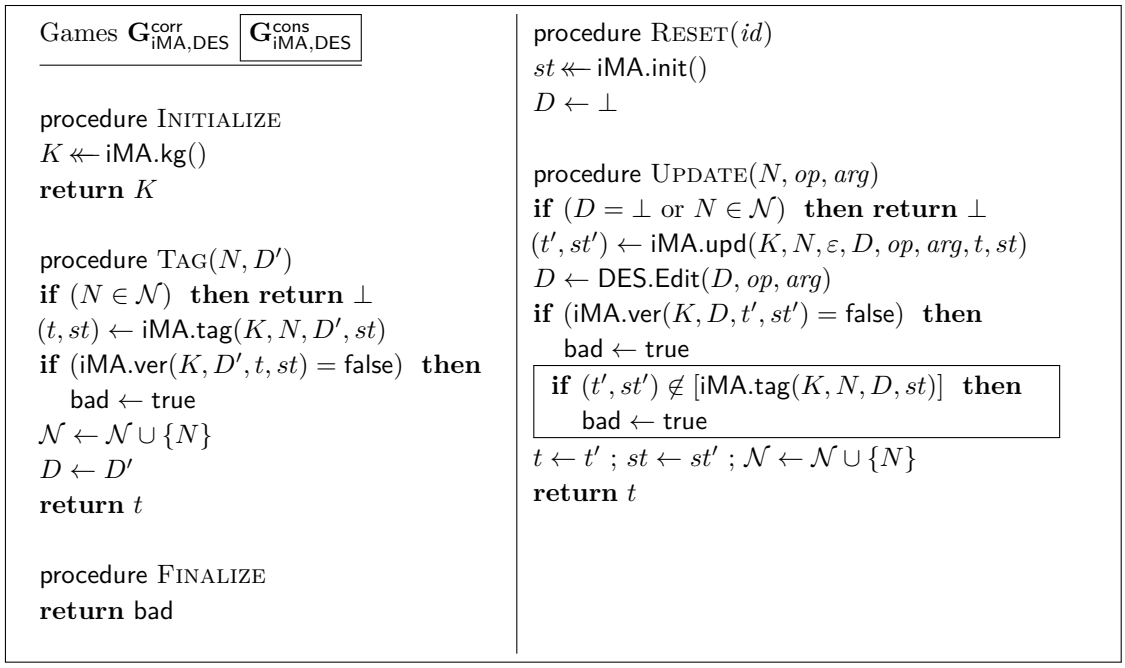

Figure 5.2: Games defining Correctness and Consistency for nonce-based incremental authentication scheme iMA for the single-document setting.

Stateless incremental MACs. We say that the scheme iMA is stateless if iMA.init always returns $st = \varepsilon$, and the tagging and update algorithms also always return ε as the updated state. In this scenario, we may omit the state st as an input or output to all algorithms.

Single-Document setting. We say that iMA is a Single-Document (SD) incremental MAC scheme if the identity space $\mathsf{iMA.IS} = \{\varepsilon\}$ consists of a single identity namely the empty string ε. As for stateless schemes, in this setting we may omit the document identity id as an input or output to all algorithms. When we say that iMA is a Multi-Document (MD) incremental MAC scheme, we are simply emphasizing that, as per the definition of an incremental MAC scheme, the identity space could have any non-zero size. If a scheme iMA achieves a security notion in the multi-document setting, it implies that it readily achieves the same security notion in the single-document setting (see Figure 5.9).

Correctness. Correctness requires that tags generated under a certain key and state, for a certain document, are correct, meaning accepted by the verification algorithm, for the same key, the updated state and the same document. It also requires that update of correct tags results in correct tags. A formal correctness condition is simple to state for stateless schemes, but the stateful case requires more care. We do it via a game, formally asking that $\Pr[\mathbf{G}^{\mathsf{corr}}_{\mathsf{iMA,DES}}(\mathcal{A})] = 0$ for all adversaries $\mathcal{A}$ regardless of their running time, where game $\mathbf{G}^{\mathsf{corr}}_{\mathsf{iMA,DES}}$ is shown in Figure 5.2 and Figure 5.3, the boxed code excluded.

For the correctness and the consistency games (below), the adversary must query once the Initialize procedure and then the Reset procedure in this order. The Reset procedure

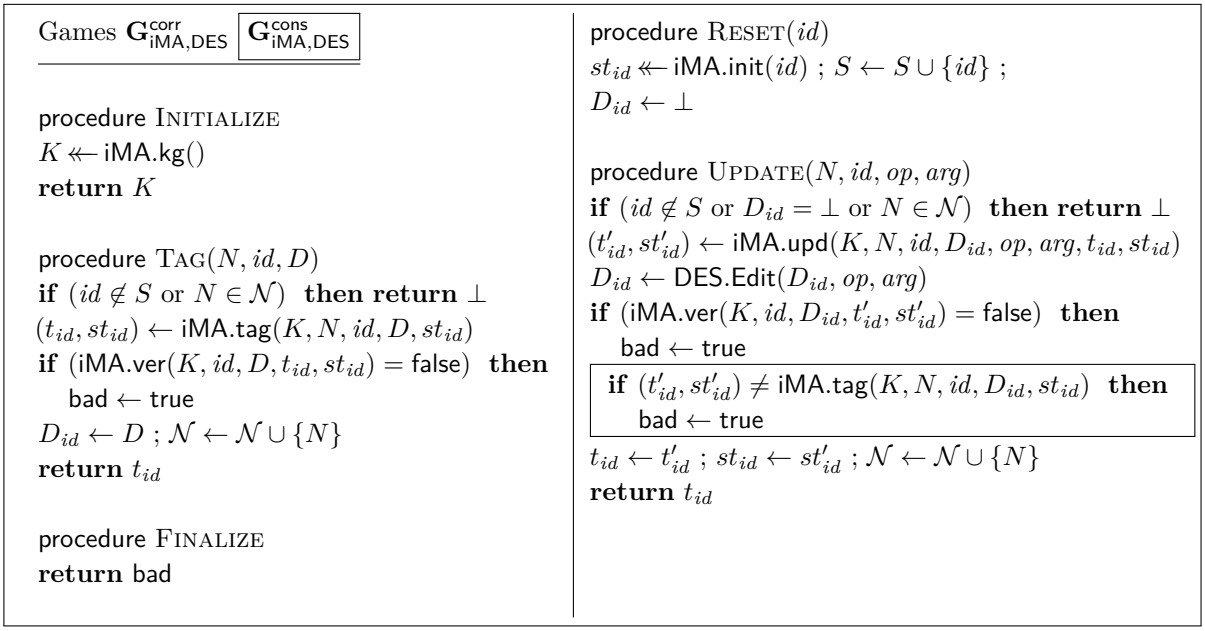

Figure 5.3: Games defining Correctness and Consistency for nonce-based incremental authentication scheme iMA for the multi-document setting.

initializes the state for a specific identification number. Then the adversary is allowed to query the Tag procedure once per document identification number, which implies once in the single-document setting; many times the Update and Verify procedures and he finishes with a query to the Finalize procedure.

Consistency. We introduce a notion called **consistency**. It asks that tags returned by the update algorithm are the same as if the updated document had been tagged directly[4] (i.e. all pairs (t'_{id}, st'_{id}) obtained by running $\mathsf{iMA.upd}(K, N, id, D_{id}, op, arg, t_{id}, st_{id})$ belong to the set $[\mathsf{iMA.tag}(K, N, id, D'_{id}, st_{id})]$ of tags generated by the algorithm iMA.tag on input $(K, N, id, D'_{id}, st_{id})$ where $D'_{id} = \mathsf{DES.Edit}(D_{id}, op, arg)$). The formalization must consider that the algorithms may be randomized and stateful, so that the actual condition is more general. Namely we say that the update algorithm has the *consistency* property if $\Pr[\mathbf{G}^{\mathsf{corr}}_{\mathsf{iMA,DES}}(\mathcal{A})] = 0$ for all adversaries $\mathcal{A}$ regardless of their running time, where game $\mathbf{G}^{\mathsf{corr}}_{\mathsf{iMA,DES}}$ is as in Figure 5.2 and Figure 5.3 but with the boxed code now included. We do not mandate consistency, but it is a strong and nice property that will be possessed by some of our schemes. For greater clarity, we gave the games for the single-document setting in Figure 5.2 and for the multi-document setting in Figure 5.2 and Figure 5.3.

[4] We consider a simple definition where the tag of the updated document belongs to the set of tags produced by the algorithm iMA.tag using the same key, the same nonce and the same state. For probabilistic incremental MACs, we could have considered a stronger consistency definition where the tag of the updated document is uniformly distributed over this set of tags.

5.1.2 Security Notions for Incremental MACs

In the following, we will present several security notions for incremental MACs that correspond to different types of *Incremental UnForgeability* (IUF). Two security notions, namely the *basic security* and the *tamper-proof security*, were introduced in previous works [BGG94, Fis97a] but the formal security definition of the latter was unclear. As mentioned above, in the basic security scenario, the adversary can obtain authentication tags for any messages of its choice and it can also update valid message/tag pairs. The corresponding security game can be easily described (see Figure 5.4). The tamper-proof security considers much stronger adversaries that can modify documents and tags arbitrarily before applying update operations. In this setting, it is quite complex to define the notion of *forgery* since the list of messages that are actually authenticated is difficult to define. We define several new notions that try to capture this tamper-proof security notion. In order to do so, for each security notion, we will consider both Single-Document (SD) and Multi-Document (MD) settings, except for basic security as it was defined without it.

In the multi-document setting, the adversary can get signatures to multiple documents of its choice by using the Tag procedure with different document identities. The single document setting is the special case where the document identity id is fixed as the empty string ε then each value X_{id} can be read as X in the single document setting. After calling the Initialize procedure, the Reset procedure has to be called to initialize the state; after that it can be called at any moment. Then the Tag procedure has to be called at most once for each document (i.e. with a fixed identity document). After that, the adversary can make an arbitrary number of queries to the Update and the Verify procedures. The adversary $\mathcal{A}$ has access to a Reset procedure that erases the document and the state st in the SD setting or a specific document in the MD setting if $\mathcal{A}$ gives the document identification number id as input to the oracle. Finally, the last call of $\mathcal{A}$ is a call to Finalize. The security game of IUF-BS is defined for any document: there is no distinction between single and multiple document settings. The adversary $\mathcal{A}$ playing this security game has to follow the rules defined above except that he can call many times the Tag oracle (otherwise only one document can be updated).

Basic Security (IUF-BS). This security notion is the basic security as defined in [BGG94]. In this setting, the documents are not identified (and as in the single-document setting we omit the document identity id as an input or output to all algorithms) and we assume that the documents and tags are stored securely. Hence, the update operations have to be done on authentic documents D and tags t that is why the update operations are done on pairs (D, t) where $\mathsf{iMA.ver}(., D, t, .)$ returns true. The corresponding security game is given in Figure 5.4. Another specificity is that here the adversary can call many times the Tag oracle as mentioned previously. Let iMA be a message authentication scheme that is incremental for the document editing system DES. The advantages of an adversary $\mathcal{A}$ playing the IUF-BS security game is defined as $\mathbf{Adv}^{\mathsf{IUF\text{-}BS}}_{\mathsf{iMA}} = \Pr[\mathbf{G}^{\mathsf{IUF\text{-}BS}}_{\mathsf{iMA}}]$.

Incremental UnForgeability 1 (IUF1). IUF1 security notion is close to the IUF-BS, the main difference is that in IUF1 the adversary is allowed to call the Tag oracle only once per identification number. In other words, a document, identified with the number id, can be tagged only once. Then he can update all of them as he wants to. Each update operation has to be applied to the previous versions of the document and the tag: $\mathcal{A}$ cannot ask to

Games $\mathbf{G}^{\mathsf{IUF\text{-}BS}}_{\mathsf{iMA}}$

procedure INITIALIZE
$K \twoheadleftarrow \mathsf{iMA.kg}()$

procedure RESET()
$st \twoheadleftarrow \mathsf{iMA.init}()$

procedure TAG(N, D)
if $(N \in \mathcal{N})$ **then return** $\perp$
$(t, st) \leftarrow \mathsf{iMA.tag}(K, N, D, st)$
$\mathrm{DL} \leftarrow \mathrm{DL} \cup \{D\}$
return t

procedure UPDATE(N, D, op, arg, t)
if $(N \in \mathcal{N})$ **then return** $\perp$
if $\mathsf{iMA.ver}(K, D, t, st) = \mathsf{false}$ **then return** $\perp$
$(t, st) \leftarrow \mathsf{iMA.upd}(K, N, D, op, arg, t, st)$
$D \leftarrow \mathsf{DES.Edit}(D, op, arg)$
$\mathrm{DL} \leftarrow \mathrm{DL} \cup \{D\}$
$\mathcal{N} \leftarrow \mathcal{N} \cup \{N\}$
return t

procedure VERIFY(D, t)
return $\mathsf{iMA.ver}(K, D, t, st)$

procedure FINALIZE(D, t)
if $(D) \in \mathrm{DL}$ **then return** false
return $\mathsf{iMA.ver}(K, D, t, st)$

Figure 5.4: Games defining IUF-BS security of an incremental authentication scheme iMA.

update a specific version of the document but only the latest one that will be called in the following the *current* document and the *current* tag. Hence, the update oracle does not take as input a document D and a tag t, these values are global variables and are updated in the game. It also means that for each update query the document D and a tag t are authentic as $\mathcal{A}$ cannot tamper with them. Let iMA be a message authentication scheme that is incremental for the document editing system DES. The advantages of an adversary $\mathcal{A}$ playing against the IUF1 security game in the single-document setting IUF1-SD and the multi-document setting IUF1-MD are defined respectively as $\mathbf{Adv}^{\mathsf{IUF1\text{-}SD}}_{\mathsf{iMA}} = \Pr[\mathbf{G}^{\mathsf{IUF1\text{-}SD}}_{\mathsf{iMA}}]$ and $\mathbf{Adv}^{\mathsf{IUF1\text{-}MD}}_{\mathsf{iMA}} = \Pr[\mathbf{G}^{\mathsf{IUF1\text{-}MD}}_{\mathsf{iMA}}]$. The corresponding security games are given in Figure 5.5.

As mentioned in chapter 2, for disk authentication a specific tampering attack is what we can call a "downgrade attack" or a "replay attack": the adversary snapshots the entire disk at some specific time and waits for a specific moment to restore it. A desired property to thwart this is a so-called **temporal data authentication**: at any moment, a user wants to be sure that the data he is manipulating now is the data stored during the previous legitimate manipulation. The security notions IUF1-SD and IUF1-MD does not provide such security guarantee since the list DL contains all documents/tags generated by the procedures TAG and UPDATE. We thus define two stronger security notions where the document list DL contains only the last version of the document authenticated (for each document identification number). These security notions are described in Figure 5.6 and are called IUF1R-SD and IUF1R-MD (where the letter R stands for "Replay attack").

INCREMENTAL UNFORGEABILITY 2 (IUF2). This security notion tries to capture what was called *tamper-proof security* in [BGG95]. The adversary is given the power to tamper with documents and authentication tags stored in the system, and thus obtain the result of the incremental scheme on document-tag pairs which need not be valid. This is modeled by allowing the adversary to choose the document and tag given as input to the update

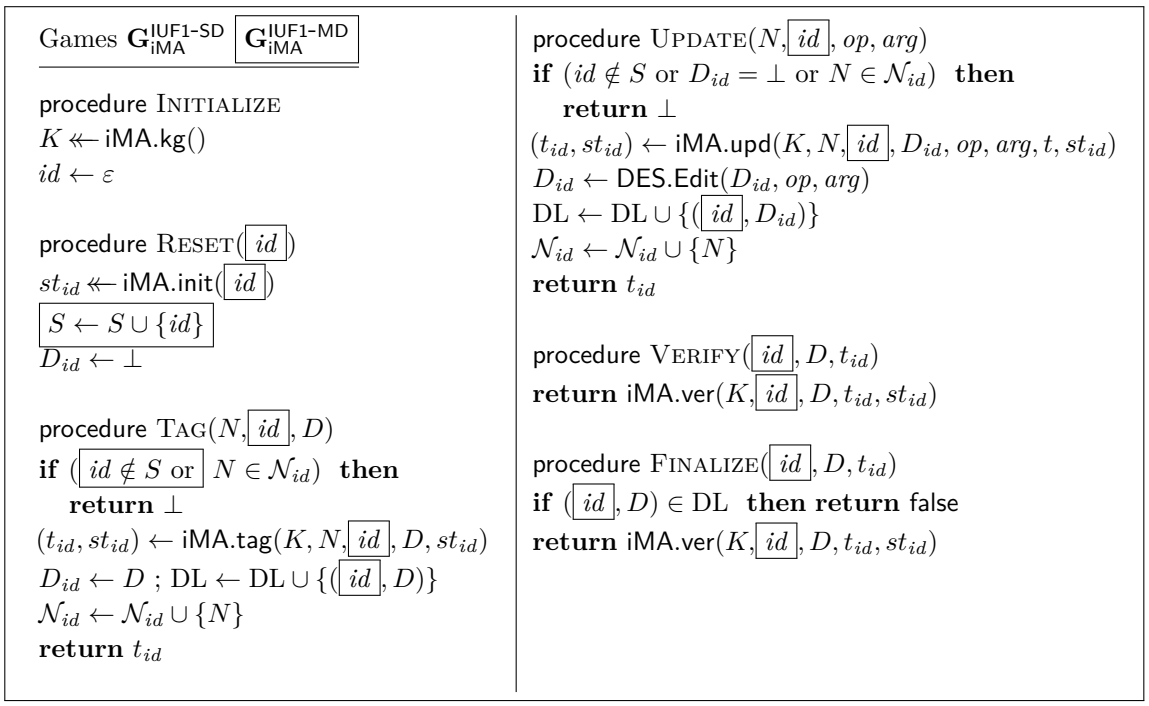

Figure 5.5: Games defining IUF1 security of an incremental authentication scheme iMA for both the Single-Document (SD) and the Multi-Document (MD) settings. The framed boxes are excluded from the SD security game.

algorithm (as seen in the games Figure 5.7). In order to formally define what is a forgery, in IUF2, the adversary is allowed to apply update operations on any tags and any documents; and the list of authenticated documents DL is constructed by performing the same update operation to the current document. The adversary succeeds in producing a forgery if it is able to output a valid pair tag/document where the document does not belong to DL. Let iMA be a message authentication scheme that is incremental for the document editing system DES. The advantages of an adversary $\mathcal{A}$ playing against the IUF2 security game in in the single-document setting IUF2-SD and the multi-document setting IUF2-MD are defined respectively as $\mathbf{Adv}^{\mathsf{IUF2\text{-}MD}}_{\mathsf{iMA}} = \Pr[\mathbf{G}^{\mathsf{IUF2\text{-}MD}}_{\mathsf{iMA}}]$ and $\mathbf{Adv}^{\mathsf{IUF2\text{-}SD}}_{\mathsf{iMA}} = \Pr[\mathbf{G}^{\mathsf{IUF2\text{-}SD}}_{\mathsf{iMA}}]$. The corresponding security games are given in Figure 5.7.

In order to achieve temporal data authentication and prevent "replay attacks", we also introduce the security notions described in Figure 5.8 which are called IUF2R-SD and IUF2R-MD (where the letter R stands for "Replay attack") which differs only to IUF2-SD and IUF2-MD by the fact that the document list DL contains only the last version of the document authenticated (for each document identification number).

Remark 5.1.1. (Relations with other incremental security notions) In [Fis97a], Fischlin mentioned three security models where the single and multi-document settings are not defined (e.g. the adversary is not restricted to one TAG query):

- *basic attack*: the adversary plays in the basic security game IUF-BS as defined above;
- *message substitution attack*: the adversary $\mathcal{A}$ is allowed to tamper the document but

Games $\mathbf{G}_{\mathsf{iMA}}^{\mathsf{IUF1R\text{-}SD}}$ $\boxed{\mathbf{G}_{\mathsf{iMA}}^{\mathsf{IUF1R\text{-}MD}}}$

procedure INITIALIZE
$K \twoheadleftarrow \mathsf{iMA.kg}()$
$id \leftarrow \varepsilon$

procedure RESET($\boxed{id}$)
$st_{id} \twoheadleftarrow \mathsf{iMA.init}(\boxed{id})$
$\boxed{S \leftarrow S \cup \{id\}}$
$D_{id} \leftarrow \perp$

procedure TAG($N, \boxed{id}, D$)
if ($\boxed{id \notin S \text{ or }} N \in \mathcal{N}_{id}$) **then**
  return $\perp$
$(t_{id}, st_{id}) \leftarrow \mathsf{iMA.tag}(K, N, \boxed{id}, D, st_{id})$
$D_{id} \leftarrow D$; $\mathrm{DL} \leftarrow \{(\boxed{id}, D)\}$
$\mathcal{N}_{id} \leftarrow \mathcal{N}_{id} \cup \{N\}$
return t_{id}

procedure UPDATE($N, \boxed{id}, op, arg$)
if ($id \notin S$ or $D_{id} = \perp$ or $N \in \mathcal{N}_{id}$) **then**
  return $\perp$
$(t_{id}, st_{id}) \leftarrow \mathsf{iMA.upd}(K, N, \boxed{id}, D_{id}, op, arg, t, st_{id})$
$D_{id} \leftarrow \mathsf{DES.Edit}(D_{id}, op, arg)$
$\mathrm{DL} \leftarrow \{(\boxed{id}, D_{id})\}$
$\mathcal{N}_{id} \leftarrow \mathcal{N}_{id} \cup \{N\}$
return t_{id}

procedure VERIFY($\boxed{id}, D, t_{id}$)
return $\mathsf{iMA.ver}(K, \boxed{id}, D, t_{id}, st_{id})$

procedure FINALIZE($\boxed{id}, D, t_{id}$)
if ($\boxed{id}, D) \in \mathrm{DL}$ **then return** false
return $\mathsf{iMA.ver}(K, \boxed{id}, D, t_{id}, st_{id})$

Figure 5.6: Games defining IUF1R security of an incremental authentication scheme iMA for both the Single-Document (SD) and the Multi-Document (MD) settings. The framed boxes are excluded from the SD security game.

not the tag. This model is close to IUF2-MD but weaker: the tag has to be kept in the state (then $\mathcal{A}$ cannot tamper with it). Moreover, the UPDATE oracle has to use the non tampered tag then either the tag given as input by the adversary is compared to the one stored in state or either the TAG oracle does not take as input a tag and the tag from state is directly used.

- *total substitution attack*: the adversary is allowed to tamper with the documents and the tags but the notion of forgery is not formally defined.

5.1.3 Relations among Security Notions for Incremental MACs

In this subsection, we analyse the relations among the different security models defined in the previous subsection.

If one restricts, an incremental MAC secure in the multi-document setting, to authenticate a single document, one gets trivially the following proposition:

Proposition 5.1.2. *Let $x \in \{1, 2\}$ and $y \in \{\mathsf{R}, \epsilon\}$. Let* iMA *be an incremental authentication scheme for the document editing system* DES *in the multi-document setting. If we have an adversary $\mathcal{A}$ against the* IUFxy-SD *security of* iMA *(restricted to one document identification number) which makes one query to the* TAG *procedure and which runs in time t, then we can construct an adversary $\mathcal{B}$ which makes one query to the* TAG *procedure, that runs in time $t' \leq t$ against the* IUFxy-MD *security of* iMA *such that* $\mathbf{Adv}_{\mathsf{iMA}}^{\mathsf{IUFxy\text{-}SD}}(\mathcal{A}) \leq \mathbf{Adv}_{\mathsf{iMA}}^{\mathsf{IUFxy\text{-}MD}}(\mathcal{B})$.

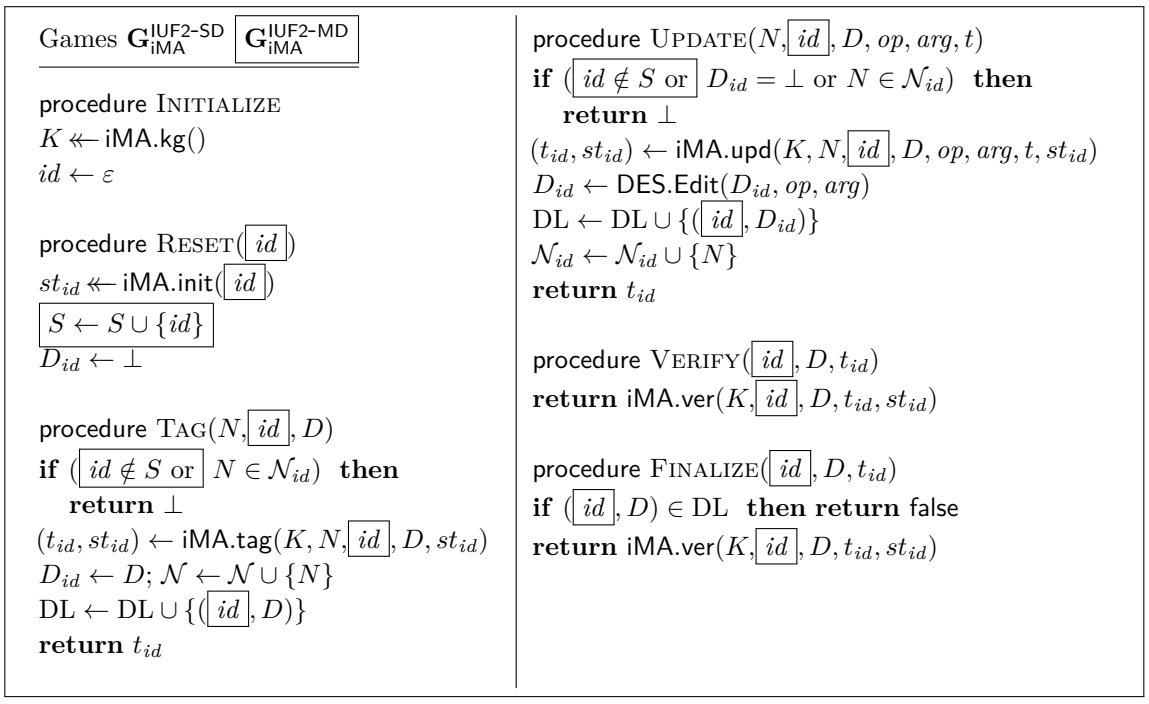

Figure 5.7: Game defining IUF2 security of an incremental authentication scheme iMA in the Single-Document (SD) and in the Multi-Document (MD) settings. The framed boxes are excluded from the SD game.

□

However, assuming the existence of a secure MAC, there exist incremental MAC secure in the IUFxy-SD model which are not secure in the IUFxy-MD (for $x \in \{1, 2\}$ and $y \in \{\mathsf{R}, \epsilon\}$). It suffices to construct an incremental MAC in the multi-document setting from the IUFxy-SD-secure incremental MAC in the single-document setting which produces MAC independent of the document identification numbers. This scheme still achieves IUFxy-SD security but do not achieves IUFxy-MD (since a valid tag for a document with identification number id is a valid forgery for the same document with an identification number $id' \neq id$).

Let $y \in \{\mathsf{R}, \epsilon\}$ and $z \in \{\mathsf{SD}, \mathsf{MD}\}$. Since the capabilities of an adversary in game $\mathbf{G}^{\mathsf{IUF2y\text{-}z}}_{\mathsf{iMA}}$ are stronger than those of an adversary in game $\mathbf{G}^{\mathsf{IUF1y\text{-}z}}_{\mathsf{iMA}}$, we get readily the following proposition:

Proposition 5.1.3. *Let $y \in \{\mathsf{R}, \epsilon\}$ and $z \in \{\mathsf{SD}, \mathsf{MD}\}$. Let* iMA *be an incremental authentication scheme for the document editing system* DES *in the multi-document setting. If we have an adversary $\mathcal{A}$ against the* IUF1y-z *security of* iMA *which makes no more than q queries to the* Tag *procedure and runs in time t, then we can construct an adversary $\mathcal{B}$ which makes no more than $q' \leq q$ queries to the* Tag *procedure and runs in time $t' \leq t$ against the* IUF2y-z *security of* iMA *such that* $\mathbf{Adv}^{\mathsf{IUF1y\text{-}z}}_{\mathsf{iMA}}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{IUF2y\text{-}z}}_{\mathsf{iMA}}(\mathcal{B})$.

□

Let $y \in \{\mathsf{R}, \epsilon\}$ and $z \in \{\mathsf{SD}, \mathsf{MD}\}$. Assuming the existence of a secure MAC, there exist incremental MAC secure in the IUF1y-z model which are not secure in the IUF2y-z. It suffices

Games $\mathbf{G}^{\mathsf{IUF2R\text{-}SD}}_{\mathsf{iMA}}$ $\boxed{\mathbf{G}^{\mathsf{IUF2R\text{-}MD}}_{\mathsf{iMA}}}$

procedure INITIALIZE
$K \twoheadleftarrow \mathsf{iMA.kg}()$
$id \leftarrow \varepsilon$

procedure RESET($\boxed{id}$)
$st_{id} \twoheadleftarrow \mathsf{iMA.init}(\boxed{id})$
$\boxed{S \leftarrow S \cup \{id\}}$
$D_{id} \leftarrow \perp$

procedure TAG($N, \boxed{id}, D$)
if ($\boxed{id \notin S \text{ or }} N \in \mathcal{N}_{id}$) **then**
  return $\perp$
$(t_{id}, st_{id}) \leftarrow \mathsf{iMA.tag}(K, N, \boxed{id}, D, st_{id})$
$D_{id} \leftarrow D$; $\mathcal{N} \leftarrow \mathcal{N} \cup \{N\}$
$\mathrm{DL} \leftarrow \{(\boxed{id}, D)\}$
return t_{id}

procedure UPDATE($N, \boxed{id}, D, op, arg, t$)
if ($\boxed{id \notin S \text{ or }} D_{id} = \perp$ or $N \in \mathcal{N}_{id}$) **then**
  return $\perp$
$(t_{id}, st_{id}) \leftarrow \mathsf{iMA.upd}(K, N, \boxed{id}, D, op, arg, t, st_{id})$
$D_{id} \leftarrow \mathsf{DES.Edit}(D_{id}, op, arg)$
$\mathrm{DL} \leftarrow \{(\boxed{id}, D_{id})\}$
$\mathcal{N}_{id} \leftarrow \mathcal{N}_{id} \cup \{N\}$
return t_{id}

procedure VERIFY($\boxed{id}, D, t_{id}$)
return $\mathsf{iMA.ver}(K, \boxed{id}, D, t_{id}, st_{id})$

procedure FINALIZE($\boxed{id}, D, t_{id}$)
if ($\boxed{id}, D$) $\in$ DL **then return** false
return $\mathsf{iMA.ver}(K, \boxed{id}, D, t_{id}, st_{id})$

Figure 5.8: Game defining IUF2R security of an incremental authentication scheme iMA in the Single-Document (SD) and in the Multi-Document (MD) settings. The framed boxes are excluded from the SD game.

to construct an incremental MAC from the IUF1y-z-secure incremental MAC by modifying only the update algorithm such that it outputs the key when executed on a public specific tag τ^* (for any nonce, document and operation). Since the original scheme is IUF1y-z-secure, this tag is unlikely to be output by a tag or update algorithm and the modified scheme remains IUF1y-z-secure (since an adversary cannot query τ^* to the update algorithm in the IUF1y-z security game). However, the modified scheme is not IUF2y-z-secure since the query of τ^* (for any nonce, document and operation) to the update algorithm (legitimate in this game) reveals immediately the authenticating key.

Let $x \in \{1, 2\}$ and $z \in \{\mathsf{SD}, \mathsf{MD}\}$. Since the games $\mathbf{G}^{\mathsf{IUFxR\text{-}z}}_{\mathsf{iMA}}$ and $\mathbf{G}^{\mathsf{IUFx\text{-}z}}_{\mathsf{iMA}}$ are identical except that the document list DL in the former game is a subset of the document list in the latter game, we get readily the following proposition:

Proposition 5.1.4. *Let $x \in \{1, 2\}$ and $z \in \{\mathsf{SD}, \mathsf{MD}\}$. Let* iMA *be an incremental authentication scheme for the document editing system* DES *in the multi-document setting. If we have an adversary $\mathcal{A}$ against the* IUFx-z *security of* iMA *which makes no more than q queries to the* TAG *procedure and runs in time t, then we can construct an adversary $\mathcal{B}$ which makes no more than $q' \leq q$ queries to the* TAG *procedure and runs in time $t' \leq t$ against the* IUFxR-z *security of* iMA *such that* $\mathbf{Adv}^{\mathsf{IUFx\text{-}z}}_{\mathsf{iMA}}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{IUFxR\text{-}z}}_{\mathsf{iMA}}(\mathcal{B})$.

□

However, a stateless incremental MAC cannot achieve the security notions IUFxR-z (since an adversary can simply output a previous tag as its forgery that would be accepted as

valid). This gives a simple separation with the security notions IUFx-z and IUFxR-z that can be achieved with stateless schemes (assuming the existence of classical MACs or one-way functions).

Remark 5.1.5 (From IUFx-z security to IUFxR-z security)**.** If an iMA_1 achieves IUFx-z security then an IUFxR-z secure scheme iMA_2 can be built as follows: let t be the tag produced by iMA_1, with the tag or update queries, it is sufficient to xor the value $\mathrm{MAC}(ctr)$, where MAC is a standard authentication scheme and ctr is a counter incremented for each tag and update queries, with the tag t. This resulting construction iMA_2 is stateful and the state contains the counter value ctr.

The following proposition states that if an incremental MAC is IUF-BS then it is also IUF1-SD:

Proposition 5.1.6. *Let* iMA *be an incremental authentication scheme for the document editing system* DES *in the multi-document setting. If we have an adversary $\mathcal{A}$ against the* IUF1-SD *security of* iMA *which makes no more than q queries to the* Tag *procedure and which runs in time t, then we can construct an adversary $\mathcal{B}$ which makes no more than $q' \leq q$ queries to the* Tag *procedure that runs in time $t' \leq t$ against the* IUF-BS *security of* iMA *such that* $\mathbf{Adv}^{\mathsf{IUF1\text{-}SD}}_{\mathsf{iMA}}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{IUF\text{-}BS}}_{\mathsf{iMA}}(\mathcal{B})$.

Proof. This proposition follows readily from the fact that an adversary $\mathcal{A}$ for the IUF1-SD security notion is also a valid adversary for the IUF-BS security (with possibly several document identifiers *id*) which succeeds in the game $\mathbf{G}^{\mathsf{IUF\text{-}BS}}_{\mathsf{iMA}}$ if $\mathcal{A}$ succeeds in game $\mathbf{G}^{\mathsf{IUF1\text{-}SD}}_{\mathsf{iMA}}$ (within the same time bound). We can therefore simply consider $\mathcal{B} = \mathcal{A}$ and the result follows. □

Finally, generalizing the previous examples, we can see that there exist an incremental MAC which is IUF-BS-secure but not IUF2-SD-secure (with a construction similar to the counter-example following Proposition 5.1.3) and an an incremental MAC which is IUF-BS-secure but not IUF1R-SD-secure (with a construction similar to the counter-example following Proposition 5.1.4).

Figure 5.9 presents a synthetic view of the relations among security notions for incremental MACs.

5.1.4 From Single-Document to Multi-Document Security

In the previous subsection, after Proposition 5.1.2, we explain that there exist incremental MAC secure in the IUFxy-SD model which are not secure in the IUFxy-MD (for $x \in \{1, 2\}$ and $y \in \{\mathsf{R}, \epsilon\}$). In this section, we show a generic construction of a scheme in the multi-document setting with the latter security property from a scheme in the single document setting with the former security property.

Suppose we have an incremental message authentication scheme iMA1 for the single-document setting for the editing system DES. We show how to transform this into an incremental message authentication scheme iMA2, $\mathsf{iMA2} = \mathbf{SDtoMD_1}(\mathsf{iMA1}, \mathsf{F})$, for the multi-document setting (also for the same document editing system DES) such that: if iMA1 is IUFxy-SD secure and F is a pseudorandom function, then iMA2 is IUFxy-MD secure, for $x \in \{1, 2\}$ and $y \in \{\mathsf{R}, \epsilon\}$. The description is given in Figure 5.10 and is denoted as $\mathsf{iMA2} = \mathbf{SDtoMD_1}(\mathsf{iMA1}, \mathsf{F})$.

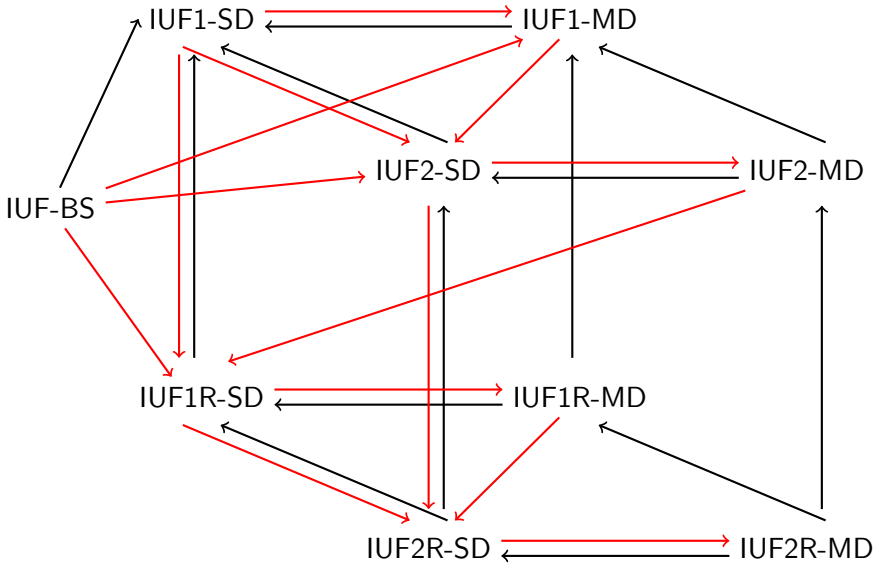

Figure 5.9: Relations among security notions for incremental MACs.
A black arrow is an implication, and in the directed graph given by the black arrows, there is a path from security notion A to security notion B if and only A implies B. The red arrows represent separations (i.e. if there is a red arrow from security notion A to security notion B, then – assuming the existence of one-way functions – there exist a scheme which achieves A but does not achieve B). All other relations follow automatically.

Theorem 5.1.7. *Let* $x \in \{1,2\}$ *and* $y \in \{\mathsf{R}, \epsilon\}$. *Let* iMA1 *be an incremental authentication scheme for the document editing system* DES *in the single-document setting and let* F *denote a pseudorandom function* $\mathsf{F} : \mathsf{F.KS} \times \mathsf{iMA2.IS} \to \mathsf{iMA1.KS}$. *Let* iMA2 *be the incremental authentication scheme for the same document editing system in the multi-document setting that is generated by the transform given in Figure 5.10, as* $\mathsf{iMA2} = \mathbf{SDtoMD_1}(\mathsf{iMA1}, \mathsf{F})$. *If we have an adversary* $\mathcal{A}$ *against the* IUFxy-SD *security of* iMA2 *which makes no more than* q *queries to the* Tag *procedure and which runs in time* t, *then we can construct an adversary* $\mathcal{B}$ *that runs in time* $t' \leq t + q \cdot T(\mathsf{iMA.init})$ *(where* $T(\mathsf{iMA.kg})$ *denotes the running time of the algorithm* iMA.kg*) against the* IUFxy-SD *security of* iMA1 *and a distinguisher* $\mathcal{D}$ *that runs in time* $t'' \leq t$ *for the PRF* F *such that*

$$\mathbf{Adv}^{\mathsf{IUFxy\text{-}MD}}_{\mathsf{iMA2}}(\mathcal{A}) \leq q \cdot \mathbf{Adv}^{\mathsf{IUFxy\text{-}SD}}_{\mathsf{iMA1}}(\mathcal{B}) + \mathbf{Adv}^{\mathsf{prf}}_{\mathsf{F}}(\mathcal{D})$$

Proof. We proceed via a series of games. The game G_0 is exactly the IUFxy-MD game for the transform $\mathsf{iMA2} = \mathbf{SDtoMD_1}(\mathsf{iMA1}, \mathsf{F})$ defined in Figure 5.12. We define the game G_1 in Figure 5.12 by replacing the PRF used to generate a key for each document in the tagging, update and verification oracles with a random function.

The advantage of an adversary $\mathcal{A}$ distinguishing G_0 from G_1 is reduced to an adversary $\mathcal{D}$ against the underlying PRF such that $\Pr[\mathrm{G}_0] - \Pr[\mathrm{G}_1] \leq \mathbf{Adv}^{\mathsf{prf}}_{\mathsf{F}}(\mathcal{D})$. We built an adversary

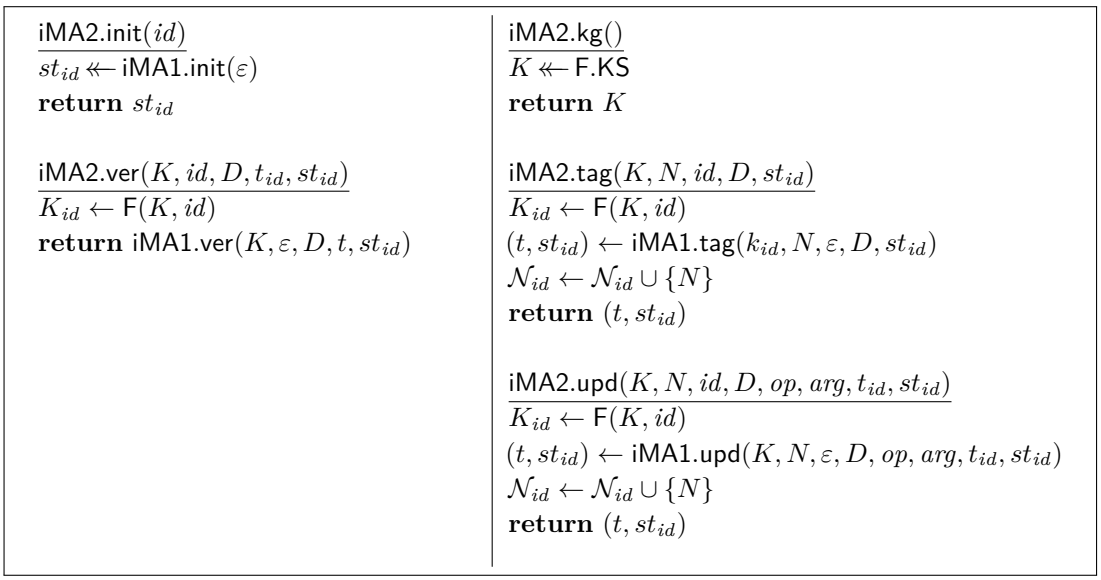

Figure 5.10: Transform $\mathbf{SDtoMD_1}$ from a single-document scheme to a multi-document scheme, denoted as $\mathsf{iMA2} = \mathbf{SDtoMD_1}(\mathsf{iMA1}, \mathsf{F})$.

$\mathcal{D}$ using the adversary $\mathcal{A}$ as shown in Figure 5.11. $\mathcal{D}$ simulates all the oracles of $\mathrm{G}_0/\mathrm{G}_1$ for $\mathcal{A}$ as follows: in all the simulated oracles, to compute K_{id}, $\mathcal{D}$ uses it own oracle F from PRF game (see Figure 1.1 for more details).

The game G_2 is exactly the IUFxy-SD game for iMA1. We construct an adversary $\mathcal{B}$ for the IUFxy-MD game in Figure 5.13. This gives us $\Pr[\,\mathrm{G}_2\,] \geq \frac{1}{q}\Pr[\,\mathrm{G}_1\,]$. The adversary $\mathcal{A}$ has no information about which id_i was chosen by $\mathcal{B}$. This means the view of $\mathcal{A}$ must be independent of the choice of i. Further, since $\mathcal{B}$ picks i at random from $\{1, 2, \ldots q\}$, we get the factor of q.

Putting together these equations, along with the facts that $\Pr[\,\mathrm{G}_0\,] = \Pr\left[\,\mathbf{G}^{\mathsf{IUFxy\text{-}MD}}_{\mathsf{iMA2}}\,\right]$ and $\Pr[\,\mathrm{G}_2\,] = \Pr\left[\,\mathbf{G}^{\mathsf{IUFxy\text{-}MD}}_{\mathsf{iMA1}}\,\right]$, we get the required inequality.

□

Adversary $\mathcal{D}^{\mathrm{F}}$
INITIALIZE (from PRF game)
$(id, D, t) \leftarrow \mathcal{A}^{\textsc{Reset},\textsc{Tag},\textsc{Update},\textsc{Verify}}$
$K_{id} \leftarrow \mathrm{F}(id)$
if $(id, D) \in \mathrm{DL}$ **then return** false
FINALIZE($\mathsf{iMA1.ver}(K, \varepsilon, D, t, st_{id})$)

UPDATE(N, id, D, op, arg, t)
if ($id \notin S$ or $D_{id} = \perp$ or $N \in \mathcal{N}_{id}$) **then**
  return $\perp$
$K_{id} \leftarrow \mathrm{F}(id)$
$(t'_{id}, st'_{id}) \leftarrow \mathsf{iMA1.upd}(K_{id}, N, \varepsilon, D, op, arg, t, st_{id})$
$D_{id} \leftarrow \mathsf{DES.Edit}(D_{id}, op, arg)$
$\mathrm{DL} \leftarrow \mathrm{DL} \cup \{(id, D_{id})\}$
$t_{id} \leftarrow t'_{id}$; $st_{id} \leftarrow st'_{id}$; $\mathcal{N}_{id} \leftarrow \mathcal{N}_{id} \cup \{N\}$
return t_{id}

RESET(id)
$st_{id} \twoheadleftarrow \mathsf{iMA1.init}(\varepsilon)$
$S \leftarrow S \cup \{id\}$; $D_{id} \leftarrow \perp$

TAG(N, id, D)
if ($id \notin S$ or $N \in \mathcal{N}_{id}$) **then**
  return $\perp$
$K_{id} \leftarrow \mathrm{F}(id)$
$(t_{id}, st_{id}) \leftarrow \mathsf{iMA1.tag}(K_{id}, N, \varepsilon, D, st_{id})$
$D_{id} \leftarrow D$; $\mathrm{DL} \leftarrow \mathrm{DL} \cup \{(id, D)\}$
$\mathcal{N}_{id} \leftarrow \mathcal{N}_{id} \cup \{N\}$
return t_{id}

VERIFY(id, D, t_{id})
$K_{id} \leftarrow \mathrm{F}(id)$
return $\mathsf{iMA1.ver}(K_{id}, \varepsilon, D, t_{id}, st_{id})$

Figure 5.11: Adversary $\mathcal{D}$ against the PRF game using the adversary $\mathcal{A}$ playing the indistinguishability game between G_0 and G_1 (see Figure 5.12) used in in the proof of Theorem 5.1.7.

procedure INITIALIZE — G_0
$K \twoheadleftarrow \mathsf{F.KS}$; $f \leftarrow \mathsf{F}_K$

procedure RESET(id) — $\mathrm{G}_0, \mathrm{G}_1$
$st_{id} \twoheadleftarrow \mathsf{iMA1.init}(\varepsilon)$
$S \leftarrow S \cup \{id\}$; $D_{id} \leftarrow \perp$

procedure UPDATE(N, id, D, op, arg, t) — $\mathrm{G}_0, \mathrm{G}_1$
if ($id \notin S$ or $D_{id} = \perp$ or $N \in \mathcal{N}_{id}$) **then**
  return $\perp$
$K_{id} \leftarrow f(id)$
$(t_{id}, st_{id}) \leftarrow \mathsf{iMA1.upd}(K_{id}, \varepsilon, D, op, arg, t, st_{id})$
$D_{id} \leftarrow \mathsf{DES.Edit}(D_{id}, op, arg)$
$\mathrm{DL} \leftarrow \mathrm{DL} \cup \{(id, D_{id})\}$; $\mathcal{N}_{id} \leftarrow \mathcal{N}_{id} \cup \{N\}$
return t_{id}

procedure VERIFY(id, D, t_{id}) — $\mathrm{G}_0, \mathrm{G}_1$
$K_{id} \leftarrow f(id)$
return $\mathsf{iMA1.ver}(K_{id}, \varepsilon, D, t_{id}, st_{id})$

procedure INITIALIZE — G_1
$f \twoheadleftarrow \mathcal{F}^*(\mathsf{F.Dom}, \mathsf{F.Rng})$

procedure TAG(N, id, D) — $\mathrm{G}_0, \mathrm{G}_1$
if ($id \notin S$ or $N \in \mathcal{N}_{id}$) **then return** $\perp$
$K_{id} \leftarrow f(id)$
$(t_{id}, st_{id}) \leftarrow \mathsf{iMA1.tag}(K_{id}, N, \varepsilon, D, st_{id})$
$D_{id} \leftarrow D$; $\mathrm{DL} \leftarrow \mathrm{DL} \cup \{(id, D)\}$
$\mathcal{N}_{id} \leftarrow \mathcal{N}_{id} \cup \{N\}$
return t_{id}

procedure FINALIZE(id, D, t_{id}) — $\mathrm{G}_0, \mathrm{G}_1$
$K_{id} \leftarrow f(id)$
if $(id, D) \in \mathrm{DL}$ **then return** false
return $\mathsf{iMA1.ver}(K_{id}, \varepsilon, D, t_{id}, st_{id})$

Figure 5.12: Games for proof of security of the **SDtoMD$_1$** transform (Theorem 5.1.7).

Adversary $\mathcal{B}^{\text{TAG,UPDATE,VERIFY}}$

INITIALIZE
$i \twoheadleftarrow \{1, 2, \ldots, q\}$
$f \twoheadleftarrow \mathcal{F}^*(\mathsf{F.Dom}, \mathsf{F.Rng})$
$(id, D, t) \leftarrow \mathcal{A}^{\text{RESET}', \text{TAG}', \text{UPDATE}', \text{VERIFY}'}$
if $(id = id_i)$ **then** FINALIZE(D, t)

UPDATE$'(N, id, D, op, arg, t)$
if $(id \notin S$ or $D_{id} = \perp$ or $N \in \mathcal{N}_{id})$ **then**
  return $\perp$
if $(id = id_i)$ **then**
  return UPDATE(N, D, op, arg, t)
$K_{id} \leftarrow f(id)$
$(t_{id}, st_{id}) \leftarrow \mathsf{iMA1.upd}(K_{id}, N, \varepsilon, D, op, arg, t, st_{id})$
$D_{id} \leftarrow \mathsf{DES.Edit}(D_{id}, op, arg)$
$\text{DL} \leftarrow \text{DL} \cup \{(id, D_{id})\}$
$\mathcal{N}_{id} \leftarrow \mathcal{N}_{id} \cup \{N\}$
return t_{id}

RESET$'(id)$
if $(id = id_i)$ **then** RESET(ε)
$st_{id} \twoheadleftarrow \mathsf{iMA1.init}(\varepsilon)$
$S \leftarrow S \cup \{id\}$; $D_{id} \leftarrow \perp$

TAG$'(N, id, D)$
if $(id \notin S$ or $N \in \mathcal{N}_{id})$ **then**
  return $\perp$
if $(id = id_i)$ **then**
  return TAG(N, D)
$K_{id} \twoheadleftarrow f(id)$
$(t_{id}, st_{id}) \leftarrow \mathsf{iMA1.tag}(K_{id}, N, \varepsilon, D, st_{id})$
$D_{id} \leftarrow D$; $\text{DL} \leftarrow \text{DL} \cup \{(id, D)\}$
$\mathcal{N}_{id} \leftarrow \mathcal{N}_{id} \cup \{N\}$
return t_{id}

VERIFY$'(id, D, t_{id})$
if $(id = id_i)$ **then return** VERIFY(D, t_{id})
$K_{id} \leftarrow f(id)$
return $\mathsf{iMA1.ver}(K_{id}, \varepsilon, D, t_{id}, st_{id})$

Figure 5.13: Construction of an adversary $\mathcal{B}$ against the IUFxy-MD security game using $\mathcal{A}$ playing the indistinguishability G_1 and G_2 in the proof of Theorem 5.1.7.

5.2 Incremental MACs with IUF1 Security

In this section, we give the description of two incremental MACs, the Xor-MAC [BGR95a] and the chaining Xor-Scheme [BGG94] constructions, and their analysis in the IUF1 and IUF-BS security models.

5.2.1 XMAC Constructions

The Xor-MAC scheme XMAC is an incremental scheme regarding the replace operation and regarding the insert and delete operations for the last message block only. A randomized version XMACR and a "counter-based" (or nonce-based) version XMACC of this scheme was given in [BGR95b] by Bellare, Guérin and Rogaway (see Figure 5.14). The Xor-MAC structure inspires different scheme designs such as the incremental signature [Fis97a] given by Fischlin, the PMAC constructions [BR02, Rog04a, Yas11].

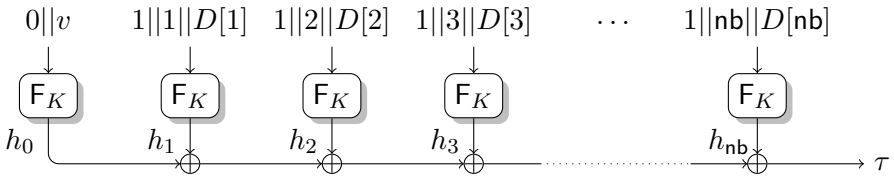

Figure 5.14: Description of the XMAC where v is a random value for XMACR and v is a nonce for XMACC.

Then XMAC is a stateless scheme defined as follows

$$\mathsf{XMAC\text{-}C/R} = (\mathsf{KS}, \mathsf{BS}, \mathsf{DS}, \mathsf{NS/RS}, \mathsf{kg}, \mathsf{tag}, \mathsf{upd}, \mathsf{ver}).$$

- XMAC is based on a pseudorandom function family $\mathsf{F} = (\mathsf{KS}, \mathsf{Dom}, \mathsf{Rng}, \mathsf{eval})$ such that $\mathsf{F} : \mathsf{F.KS} \times \mathsf{F.Dom} \to \mathsf{F.Rng}$. It follows that $\mathsf{F.Dom} = \{1\} \times \{0,1\}^p \times \mathsf{XMAC.BS}$ where $\{0,1\}^p$ is the position space and $\mathsf{F.Rng} = \{0,1\}^{t\ell}$ where $t\ell$ is the tag size.
- The **key generation algorithm** XMAC.kg is a (probabilistic) algorithm that takes no input and returns a key $K \in \mathsf{XMAC.KS}$ such that $\mathsf{XMAC.KS} = \mathsf{F.KS}$.
- The **tagging algorithm** $\mathsf{XMAC.tag}^{\mathbf{1}}$ takes as inputs the key $K \in \mathsf{XMAC.KS}$, a document $D \in \mathsf{XMAC.DS}$ and outputs a tag τ. A value v prepended by the bit 0, such that $v \in \mathsf{XMACR.RS}$ for the randomized version and such that $v \in \mathsf{XMAC.NS}$ for the counter-based version, is given as input to F and each document block $D[i] \in \mathsf{XMAC.BS}$ prepended by $1||i$ is given as input to the pseudorandom function F_K then the sum of the corresponding outputs τ and the value v is returned as the tag $t = (v, \tau)$.
- The **Verification algorithm** XMAC.ver takes as inputs the key $K \in \mathsf{XMAC.KS}$, the document D and the tag $t = (v, \tau)$. It re-computes the value τ from the inputs D and v then it returns true if this value is equal to the input τ and false otherwise.
- The **Update algorithm** $\mathsf{XMAC.upd}^{\mathbf{1}}$ takes as inputs the key $K \in \mathsf{XMAC.KS}$, the document D, the operation $op \in \mathsf{OpCodes}$ where $\mathsf{OpCodes} = \{\mathtt{R}, \mathtt{I}, \mathtt{D}\}$, the set of argument arg and the tag t. The argument arg is composed by the position i where

the block value has to be inserted, deleted or replaced and the new document block $x \in \mathsf{XMAC.BS}$ for the insert and delete operations or ε if it is a delete operation: $\arg = \langle i, x \rangle$. The *insert* and *delete* operations can be performed only for the last position: as each document block is processed with it block position, it is not possible to insert or delete a block efficiently.

In [BGR95b], Bellare *et al.* showed that XMACC (Theorem 5.2.1) and XMACR (Theorem 5.2.2) are IUF-BS secure.

Theorem 5.2.1. *Let $\mathcal{A}$ be any (computationally unbounded) an* IUF-BS*-adversary making a (q_t, q_v, q_{upd})-attack against the* MACC *with a function picked uniformly at random from $\mathcal{F}^*(\mathcal{D}, \mathcal{R})$ such that $\mathcal{D} = \{0,1\}^{2\ell}$ and $\mathcal{R} = \{0,1\}^L$. The probability that $\mathcal{A}$ is successful is at most*

$$q_v \cdot 2^{-L}.$$

Here, q_t is an upper bound on the number of tagging queries, q_{upd} is an upper bound on the number of update queries (such that $q = q_t + q_{upd}$), q_v is an upper bound on the number of verification queries.

Theorem 5.2.2. *Let $\mathcal{A}$ be any (computationally unbounded) an* IUF-BS*-adversary making a (q_t, q_v, q_{upd})-attack against the* MACR *with a function picked uniformly at random from $\mathcal{F}^*(\mathcal{D}, \mathcal{R})$ such that $\mathcal{D} = \{0,1\}^{2\ell}$ and $\mathcal{R} = \{0,1\}^L$. The probability that $\mathcal{A}$ is successful is at most*

$$q^2 \cdot 2^{\mathsf{n}-\ell} + q_v \cdot 2^{-L}.$$

Here, q_t is an upper bound on the number of tagging queries, q_{upd} is an upper bound on the number of update queries (such that $q = q_t + q_{upd}$), q_v is an upper bound on the number of verification queries.

5.2.2 XS Construction

Until this thesis work, the only strongly incremental authentication scheme was the *Xor-Scheme*, denoted XS in the following. It uses fresh random value for each processed block and was designed by Bellare, Goldreich and Goldwasser in [BGG95] (see Figure 5.15). This strong property comes with a cost: only basic security is claimed in [BGG94] and this algorithm needs to generate and store a lot of randomness. The tag algorithm generates a random value for each document block and these random values are part of the tag because they are necessary for the verification and the update operations. The Xor-Scheme is based on a pseudorandom function and a pseudorandom permutation and the incremental algorithms for (single block) insert and delete operations require only two applications of the underlying PRF and two applications of the underlying PRP. The Xor-Scheme relies on the concept of *pair block chaining* (which was later used in [GSC01] which involves taking each pair of two consecutive blocks of a message and feeding them into a pseudorandom function before chaining all the outputs of the PRF into the final hash. This scheme extends the XMAC schemes from [BGR95a] which are incremental only for replacement. Even if they share a similar name, these two algorithms are different: XMAC is not based on a pair block chaining structure and requires actually much less randomness.

[1] This algorithm is probabilistic for the randomized version and deterministic for the counter-based version.

5.2.2.1 Description of the Xor-Scheme

The Xor-Scheme XS as defined in [BGG94] is an incremental authenticated algorithm based on pair-wise chaining. The XS scheme is based on a pseudorandom function F_{K_1} and a pseudorandom permutation P_{K_2} as shown in Figure 5.15. The incremental algorithms for (single block) insert and delete operations require only two applications of F_{K_1} and two applications of the P_{K_2}. The XS scheme generates an authentication tag for a document D by repeatedly applying F_{K_1} to pairs of data blocks – each made of a document block $D[i]$ from the document D and random block r_i (pick uniformly at random and independently for each block).

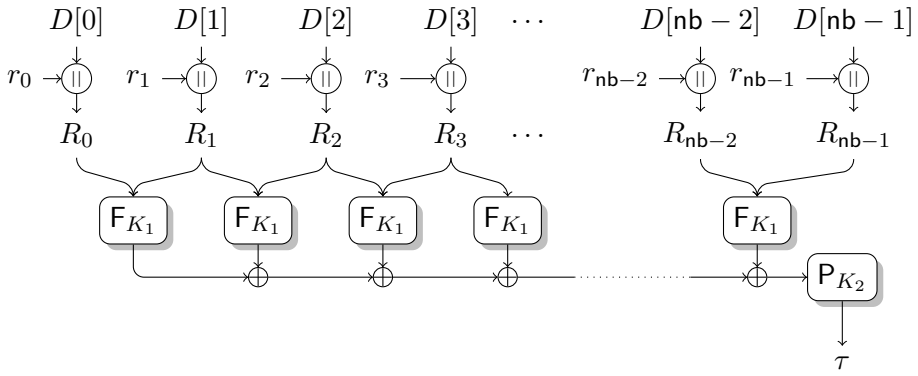

Figure 5.15: Description of the Xor-Scheme

The Xor-Scheme is probabilistic scheme and not nonce based then instead of a nonce space XS.NS, we have a random space XS.RS from which the random data blocks are randomly sampled. XS is stateless then the Initialisation algorithm XS.init is omitted (e.g. the state st is empty). It is defined as follows:

$$\mathsf{XS} = (\mathsf{KS}, \mathsf{BS}, \mathsf{DS}, \mathsf{RS}, \mathsf{kg}, \mathsf{tag}, \mathsf{upd}, \mathsf{ver}).$$

The detail of each algorithm is given in Figure 5.16.

- XS is based on a pseudorandom function family $\mathsf{F} = (\mathsf{KS}, \mathsf{Dom}, \mathsf{Rng}, \mathsf{eval})$ and a pseudorandom permutation family $\mathsf{P} = (\mathsf{KS}, \mathsf{Dom}, \mathsf{eval}, \mathsf{inverse})$ such that $\mathsf{F} : \mathsf{F.KS} \times \mathsf{F.Dom} \rightarrow \mathsf{F.Rng}$ and $\mathsf{P} : \mathsf{P.KS} \times \mathsf{P.Dom} \rightarrow \mathsf{P.Dom}$. It follows that $\mathsf{F.Dom} = \mathsf{XS.RS}^2 \times \mathsf{XS.BS}^2$ and $\mathsf{F.Rng} = \mathsf{P.Dom} = \{0,1\}^{t\ell}$ where $t\ell$ is the tag size.
- The **key generation algorithm** XS.kg is a probabilistic algorithm that takes no input and returns a key $K \in \mathsf{XS.KS}$ such that $\mathsf{XS.KS} = \mathsf{F.KS} \times \mathsf{P.KS}$.
- The **tagging algorithm** XS.tag takes as inputs the key $K \in \mathsf{XS.KS}$, a document $D \in \mathsf{XS.DS}$ and outputs a tag $t := (r, \tau)$. For each document block $D[i] \in \mathsf{XS.BS}$, a random block value $r_i \in \mathsf{XS.RS}$ is randomly sampled and its bit length is denoted rl. The concatenation of these values is denoted $R_i := D[i]||r_i$. Each couple (R_{i-1}, R_i) is processed by the function F and outputs a value denoted h_i then the bitwise XOR (eXclusive OR) of all the values (denoted Σ) is processed by the permutation P to give the value τ.

- The **Verification algorithm** $\mathsf{XS.ver}$ takes as inputs the key $K \in \mathsf{XS.KS}$, the document D and the tag $t := (r, \tau)$. It re-computes the value τ from the inputs r and D. It returns 1 if this value is equal to the input τ and 0 otherwise.

- The **Update algorithm** $\mathsf{XS.upd}$ takes as inputs the key $K \in \mathsf{XS.KS}$, the document the operation $op \in \mathsf{OpCodes}$ where $\mathsf{OpCodes} = \{\mathtt{I}, \mathtt{D}\}$, the set of argument arg and the tag t. The argument arg is composed by the position i where the block value has to be inserted or deleted and the new document block $x \in \mathsf{XS.BS}$ to insert or ε if it is a delete operation: $\text{arg} = \langle i, x \rangle$. To be more specific:
  - If $op = \mathtt{I}$, the update algorithm enables to insert a block value in a document. It takes as argument the position i where the block value has to be inserted, the document D (the previous block value $D[i-1]$ [5] and the block value $D[i]$ [6]are necessary), the new block value x and the tag t. It outputs the new tag.
  - If $op = \mathtt{D}$, the update algorithm enables to delete a block from the document. It takes as inputs the position i where the block as to be deleted, the document D (the block value to delete $D[i]$, the previous and next block values $D[i-1]$ and $D[i+1]$ are necessary) and the tag t.

Specific features. The update algorithm is intuitive and given in Figure 5.16 for update operations at a position different from the first block position. They can be adapted to be applied to the first block. In the original version, it is specified that a prefix and postfix are added to the document. For a document $D = D[1] \dots D[\mathsf{nb}]$, the authentication tag is computed on $D[0]||D[1] \dots D[\mathsf{nb}]||D[\mathsf{nb}+1]$ where $D[0]$ and $D[\mathsf{nb}+1]$ are specific prefix and postfix values. In the description given in Figure 5.16, we give the description of the original Xor-Scheme defined for basic security ($\mathsf{IUF\text{-}BS}$ security game). In this work, these specifications are not taken into account: it does not prevent our attack and the repaired scheme is proven secure without it.

Xor-Scheme limits. Supporting insert, delete and consequently replace operations should make the Xor-Scheme very efficient in term of update time running. The fresh random values r_i generated by this scheme for each new document block are necessary for security. But generating so much randomness is *time-consuming*: for an nb-block document D, a $\mathsf{nb} \cdot \mathsf{rl}$-random bits value r is generated. Random generation also slows down the insertion operation. Another drawback is the tag *expansion*: the random value r is part of the tag and needs to be stored. For a nb-block document, random storage costs $\mathsf{nb} \cdot \mathsf{rl}$ bits. Even if today storage is not an issue, having short tags is desirable.

Remark 5.2.3. A counter-based version of the Xor-Scheme can be defined: instead of having a random value space $\mathsf{XS.RS}$, it has as a parameter a nonce space NS. In this version, a counter can be used and it has to be a stateful algorithm to store this value in order to assure that it never repeats. It has the advantage to save all the random value generation that can be very consuming from a practical point of view but instead the price to pay is to store the counter.

[5] For the first position, there is no previous block.
[6] For the last position, there is no next block.

$\underline{\mathsf{XS.kg}()}$
$K_1 \twoheadleftarrow \mathsf{F.KS}$; $K_2 \twoheadleftarrow \mathsf{P.KS}$
return $K_1||K_2$

$\underline{\mathsf{XS.upd}(K, D, op, \langle i, x\rangle, t)}$
if $op \neq \mathtt{D}$ **and** $op \neq \mathtt{I}$ **then return** $\perp$
$(K_1, K_2) \leftarrow K$; $\mathsf{nb} \leftarrow |D|_{\mathsf{n}}$
$r \leftarrow t[0 : (\mathsf{nb}\text{-}1)\mathsf{rl}\text{-}1]$; $\tau \leftarrow t[(\mathsf{nb}\text{-}1)\mathsf{rl} :]$
$R_i \leftarrow r_i||D[i]$
$R_{i\text{-}1} \leftarrow r_i||D[i\text{-}1]$
if $op = \mathtt{I}$ **then**
  $r'_i \twoheadleftarrow \mathsf{XS.RS}$
  $R'_i \leftarrow r'_i||x$
  $h'_i \leftarrow \mathsf{F}(K_1, R_{i-1}||R'_i)$
  $h'_{i+1} \leftarrow \mathsf{F}(K_1, R_{i'}||R_i)$
  $h_i \leftarrow \mathsf{F}(K_1, R_{i-1}||R_i)$
  $\Sigma \leftarrow h'_i \oplus h'_{i+1} \oplus h_i$
if $op = \mathtt{D}$ **then**
  $R_{i+1} \leftarrow r_{i+1}||D[i+1]$
  $h_i \leftarrow \mathsf{F}(K_1, R_{i-1}||R_i)$
  $h_{i+1} \leftarrow \mathsf{F}(K_1, R_i||R_{i+1})$
  $h'_i \leftarrow \mathsf{F}(K_1, R_{i-1}||R_{i+1})$
  $\Sigma \leftarrow h_i \oplus h_{i+1} \oplus h'_i$
$\Sigma \leftarrow \Sigma \oplus \mathsf{P}^{-}(K_2, \tau)$
$\tau \leftarrow \mathsf{P}(K_2, \Sigma)$
$r \leftarrow r_0 \ldots r_{i-1}||r_{i+1} \ldots r_{\mathsf{nb}}$
$t \leftarrow (r, \tau)$
return t

$\underline{\mathsf{XS.tag}(K, D)}$
$(K_1, K_2) \leftarrow K$; $\Sigma \leftarrow 0^L$
$\mathsf{nb} \leftarrow |D|_{\mathsf{n}}$
$r_0 \twoheadleftarrow \mathsf{XS.RS}$
$r \leftarrow r_0$
$R_0 \leftarrow D[0]||r_0$
for $i = 1$ **to** $\mathsf{nb} - 1$ **do**
  $r_i \twoheadleftarrow \mathsf{XS.RS}$
  $r \leftarrow r||r_i$
  $R_i \leftarrow r_i||D[i]$
  $h_i \leftarrow \mathsf{F}(K_1, R_{i-1}||R_i)$
  $\Sigma \leftarrow \Sigma \oplus h_i$
$\tau \leftarrow \mathsf{P}(K_2, \Sigma)$
$t \leftarrow (r, \tau)$
return t

$\underline{\mathsf{XS.ver}(K, D, t)}$
$(K_1, K_2) \leftarrow K$; $\Sigma \leftarrow 0^L$
$\mathsf{n} \leftarrow |D|_{\mathsf{n}}$
$r \leftarrow t[0 : (\mathsf{n} - 1)\mathsf{rl} - 1]$
$\tau \leftarrow t[(\mathsf{n} - 1)\mathsf{rl} :]$
$R_0 \leftarrow r_0||D[0]$
for $i = 1$ **to** $\mathsf{n} - 1$ **do**
  $R_i \leftarrow r_i||D[i]$
  $h_i \leftarrow \mathsf{F}(K_1, R_{i-1}||R_i)$
  $\Sigma \leftarrow \Sigma \oplus h_i$
τ'$\leftarrow \mathsf{P}(K_2, \Sigma)$
return $(\tau' = \tau)$

Figure 5.16: XS Original Xor-Scheme Algorithm

5.2.2.2 Forgery Attacks against the Xor-Scheme

An analysis on some incremental hash functions was provided by Phan and Wagner [PW06]. They give, inter alia, patterns that could give collisions on a hash function based on pair block chaining. Two cases are of interest for the Xor-Scheme: *non-distinct blocks* and *cycling chaining*. The first one considers repeated blocks messages like $A||B||C||B||A$ and $B||C||B||A||B$ that would have the same sum value if no randomness was used (*cf.* Figure 5.15) but as underlined by the authors the random values appended to each message block prevents these repetitions. The second one considers a variant of the Xor-Scheme [GSC01] where the first and the last block are chained so that some repeated patters like $A||B||A$ and $B||A||B$ would have the same sum value but it not the case in the original version from [BGG95]. Therefore, until this thesis work, no attacks were known against the original strongly incremental *Xor-Scheme* proposed in [BGG95].

According to the IUF-BS security game (named basic security in [BGG94]) described in

Figure 5.4, the adversary $\mathcal{A}$ wins the game if it finds a new pair (D^*, t^*) such that the verification operation returns 1. If an adversary has access to any tag t (such that $t = (r, \tau)$) returned by the tag algorithm on a document D (for example $D[0]||D[1]||D[2]$), it can forge a different document D^* having the same value τ. The value τ is computed as follows:

$$\tau = \mathsf{P}(K_2, \mathsf{F}(K_1, D[0]||r_0 \;||\; D[1]||r_1) \oplus \mathsf{F}(K_1, D[1]||r_1 \;||\; D[2]||r_2)] \quad (5.1)$$

$$\Sigma = \mathsf{F}(K_1, R_0||R_1) \oplus \mathsf{F}(K_1, R_1||R_2) = h_1 \oplus h_2 \quad (5.2)$$

$\mathcal{A}$ can build a document $D^* \neq D$ and a value r^* such that the corresponding Σ^* value collides with Σ even if there is no weakness on F. A way to do so consists of inserting a specific block chain in document D in order to cancel all the new values h'_i introduced by these repetitions as shown in Figure 5.17. It seems that the chaining structure of the Xor-Scheme should prevent this behaviour because changing or inserting a block value will affect two values h_i then the tag τ will be different. These modifications have to be compensated: the values h'_i introduced have to be canceled by xoring the same value and all the original values h_i that are deleted have to be re-introduced. We use this trick to break the claimed *basic security*.

$$\begin{array}{ccccccccccc} (R_0, R_1) & & & (.,.) & & (.,.) & & (R_1, R_2) & & & \\ \downarrow & & & \downarrow & & \downarrow & & \downarrow & & & \\ h_1 & \oplus & \underbrace{\ldots} & \oplus & \underbrace{\ldots} & \oplus & & h_2 & = & \Sigma & \\ & & & & = 0 & & & & & & \end{array}$$

Figure 5.17: Xor cancellation strategy in the Xor-Scheme

Forgery Attack. Applying this strategy gives us an adversary $\mathcal{A}$ wining the game $\mathbf{G}^{\mathsf{IUF\text{-}BS}}_{\mathsf{XS}}$ with probability 1 and that requires only one query to the oracle Tag.

Adversary $\mathcal{A}^{\text{Tag,Update,Verify,Reset}}$

1 $\mathcal{A}$ calls Initialize.

2 $\mathcal{A}$ calls the Tag oracle with a document D as input such that $D = D[0]||D[1]||D[2]$ and receives the corresponding authentication-tag $t = (r, \tau)$.

$$\begin{array}{ccccc} (R_0, R_1) & & (R_1, R_2) & & \\ \downarrow & & \downarrow & & \\ h_1 & \oplus & h_2 & = & \Sigma \end{array}$$

Figure 5.18: Σ computation for 3 block document

3 $\mathcal{A}$ builds a document D^* from D such that $D^* = D[0]||D[1]||D[2]||D[1]||D[2]||D[1]||D[2]$ and a value r^* from r such that $r^* = r_0||r_1||r_2||r_1||r_2||r_1||r_2$. $\mathcal{A}$ calls Finalize with (D^*, t^*) such that $t^* = (r^*, \tau)$.

$$\begin{array}{cccccccccccc} (R_0,R_1) & & (R_1,R_2) & & (R_2,R_1) & & (R_1,R_2) & & (R_2,R_1) & & (R_1,R_2) & \\ \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & \\ h_1 & & \oplus \cancel{h_2} & & \oplus \cancel{h_2} & & \oplus \cancel{h_2} & & \oplus \cancel{h_2} & & \oplus h_2 & = \Sigma \end{array}$$

Figure 5.19: Attack on the Xor-Scheme. Here $R_i = D[i]||r[i]$.

The document D^* is different from D but it has the same value τ. The document D^* given in Figure 5.19 is an example of a forgery and many other examples can be given. To be more general, for any $x \in \mathsf{XS.BS}$, any $x' \in \mathsf{XS.RS}$ and for any valid pair (D,t) such that $D = D[0]\dots D[i]||D[i+1]\dots D[\mathsf{nb}]$, many forgeries $(D^*,(r^*,\tau))$ can be built by inserting the specific block chain $D[i]||x||D[i]||x$ in D (and the corresponding random value chain $r_i||x'||r_i||x'$ in r for any x') in such a way that:

$$D^* = D[0]\dots D[i-1]||\ \underbrace{D[i]||x||D[i]||x||D[i]}\ ||D[i+1]\dots D[\mathsf{nb}]$$
$$r^* = r_0\dots r_{i-1}||\ \underbrace{r_i||x'||r_i||x'||r_i||}\ r_{i+1}\dots r_{\mathsf{nb}}.$$

A variant of this forgery is to insert only a repeated document block $D[i]$ (and r_i) is the following:

$$D^* = D[0]\dots D[i-1]||\ \underbrace{D[i]||D[i]||D[i]}\ ||D[i+1]\dots D[\mathsf{nb}].$$
$$r^* = r_0\dots r_{i-1}||\ \underbrace{r_i||r_i||r_i||}\ r_{i+1}\dots r_{\mathsf{nb}}.$$

A more powerful forgery can be built from (D,t) by inserting any values x and y in D (and any values x' and y' in r) such that:

$$D^* = D[0]\dots D[i-1]||\ \underbrace{D[i]||x||y||x||y||x||D[i]||x||D[i]}\ ||D[i+1]\dots D[\mathsf{nb}].$$
$$r^* = r_0\dots r_{i-1}||\ \underbrace{r_i||x'||y'||x'||y'||x'r_i||x'||r_i||}\ r_{i+1}\dots r_{\mathsf{nb}}.$$

For all these attacks, the underbraced chains can be repeated many times. These three attacks are some of the possible attacks, following this *canceling* strategy, some exotic chains can be inserted in order to end with a value τ that corresponds to a legitimate tag. A first observation is that all these attacks are performed by inserting blocks and providing a forgery D^* that has the same length that the original one looks impossible or at least harder.

Remark 5.2.4. *It is worth noticing that this attack also breaks the* IUF1-X *security notions (in the SD and MD settings). Indeed, in our attack, the* Tag *oracle is called once, the* Update *oracle is not called and all the other oracles are the same in the* IUF-BS *and the* IUF1-X *security games.*

5.2.2.3 Modification of the Xor-Scheme

The previous section described an attack that breaks IUF-BS and IUF1-MD securities of the Xor-Scheme by producing a document D^* using a MAC query (D,t) where $\tau = \tau^*$ and $|D|_{\mathsf{n}} \neq |D^*|_{\mathsf{n}}$. All the forgeries D^* produced are longer that the original document D. One can notice that if the adversary $\mathcal{A}$ is only allowed to tag and verify documents that have the same length nb then the attack presented in Section 5.2.2.2 will fail. A first naive idea is to

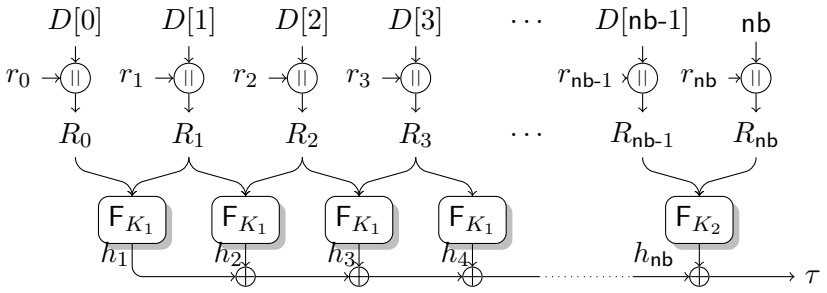

Figure 5.20: Description of the fixed Xor-Scheme

force all documents to have the same length (documents that are too small can be padded in the MAC algorithm) but this solution is not realistic and the incremental property will be lost. A natural way to fix this flaw is to use the document length nb for the computation of the value τ in order to make it *size dependent*. The size can be expressed according to any units: number of bits, bytes, blocks. Choosing the number of blocks nb is sufficient. A suffix block containing the number of blocks nb can be added at the end of the document then the computation of the value Σ will become:

$$\Sigma = \mathsf{F}(K_1, D[0]||r_0 \,||\, D[1]||r_1) \oplus \mathsf{F}(K_1, D[1]||r_1 \,||\, D[2]||r_2) \oplus \cdots \oplus \mathsf{F}(K_1, D[\mathsf{nb}\text{-}1]||r_{\mathsf{nb}\text{-}1} \,||\, r_{\mathsf{nb}}||\mathsf{nb})$$

The last block works as a mask for each value τ: incremental operation will refresh the last random value r_{nb} in order to have a different mask value for any modification. As a consequence, the pseudorandom permutation P is not necessary anymore ($\tau = \Sigma$), it is removed in the modified scheme (Figure 5.20).

The last random block value r_{nb} (concatenated with the document length nb) is necessary otherwise the corresponding h_{nb} value can be canceled. If it is omitted, the following attack is indeed possible:

1. $\mathcal{A}$ calls the Initialize oracle.
2. $\mathcal{A}$ chooses a document $D_1 = D[0]||D[1]$ and calls the Tag oracle with D as input. He receives the tag $t_1 = (r_0||r_1, \tau_1)$.
3. $\mathcal{A}$ calls the Update oracle to delete the first block of D by giving as inputs to the oracle $(D_1, \mathtt{D}, \langle 0\rangle, \tau_1)$. The resulting document is $D_2 = \mathsf{DES.Edit}(D_1, \mathtt{D}, \langle 0\rangle)$. He receives $t_2 = (r_1, \tau_2)$.
4. $\mathcal{A}$ calls the Update oracle to insert the block $D[0]'$ at the first position of the document D_2. He gives as inputs $(D_2, \mathtt{I}, \langle 0, D[0]'\rangle, \tau_2)$ and receives $t_3 = (r'_0||r_1, \tau_3)$. The resulting document is $D_3 = \mathsf{DES.Edit}(D_2, \mathtt{I}, \langle 0, D[0]'\rangle)$.
5. $\mathcal{A}$ calls the Update oracle to insert the block $D[2]$ at the position 2 of the original document D_1 using as input $(D_1, \mathtt{I}, \langle 2, D[2]\rangle, \tau_1)$. He receives $t_4 = (r_0||r_1||r_2, \tau_4)$. The resulting document is $D_4 = \mathsf{DES.Edit}(D_1, \mathtt{I}, \langle 2, D[2]\rangle)$.
6. $\mathcal{A}$ builds the document $D^* = D[0]'||D[1]||D[2]$ and the tag $t^* = (r'_0||r_1||r_2, \tau_1 \oplus \tau_3 \oplus \tau_4)$. $\mathcal{A}$ calls the Finalize oracle with (D^*, t^*).

Functions	Scheme	rl-bits Gen	F	P	Xor
tag	XS	nb	nb − 1	1	nb − 1
	MXS	nb + 1	(nb − 1) + 1	0	nb
ver	XS	0	nb − 1	1	nb − 1
	MXS	0	(nb − 1) + 1	0	nb
upd.D	XS	0	3	2	3
	MXS	1	5	0	5
upd.I	XS	1	3	2	3
	MXS	2	5	0	5

Figure 5.21: Complexity of the Xor-Scheme XS and the Modified Xor-Scheme MXS.

The couple (D^*, r^*) is a forgery: it is not in the list DL of tagged document and it has a valid tag. To avoid such attacks, for incremental operations the last random block (concatenated with the document length) needs to be always refreshed. To be sure that none of the previous attacks are practical, an independent key is used to process the last couple $(R_{\mathsf{nb}-1}, R_{\mathsf{nb}})$. That way, it would be hard for an adversary to make a forgery from a linear combination of tagged documents. This version of the Xor-Scheme is the subject of the next section.

5.2.3 MXS Construction

We give here the description of the patched XS, called in the following Modified Xor-Scheme MXS, and we prove its security in the IUF-BS security model.

5.2.3.1 Description of MXS

The MXS is a probabilistic scheme, we have a random space MXS.RS from which the random data blocks are randomly sampled. MXS is stateless then the Initialisation algorithm MXS.init is omitted (e.g. the state st is empty). It is defined as follows:

$$\mathsf{MXS} = (\mathsf{KS}, \mathsf{BS}, \mathsf{DS}, \mathsf{RS}, \mathsf{kg}, \mathsf{tag}, \mathsf{upd}, \mathsf{ver}).$$

- MXS is based on a pseudorandom function family $\mathsf{F} = (\mathsf{KS}, \mathsf{Dom}, \mathsf{Rng}, \mathsf{eval})$ such that $\mathsf{F} : \mathsf{F.KS} \times \mathsf{F.Dom} \rightarrow \mathsf{F.Rng}$. It follows that $\mathsf{F.Dom} = \mathsf{MXS.RS}^2 \times \mathsf{MXS.BS}^2$ and $\mathsf{F.Rng} = \{0,1\}^{t\ell}$ where $t\ell$ is the tag size.

- The **key generation algorithm** MXS.kg is a probabilistic algorithm that takes no input and returns a key $K \in \mathsf{MXS.KS}$ such that $\mathsf{MXS.KS} = \mathsf{F.KS}^2$.

- The **tagging algorithm** MXS.tag takes as inputs the key $K \in \mathsf{MXS.KS}$, a document $D \in \mathsf{MXS.DS}$ and outputs a tag $t := (r, \tau)$. Similarly to the XS scheme, for each document block $D[i] \in \mathsf{MXS.BS}$, a random block value $r_i \in \mathsf{MXS.RS}$ is randomly sampled and its bit length is denoted MXS.rl. The concatenation of these values is denoted $R_i := D[i]||r_i$. Each couple (R_{i-1}, R_i) is processed by the function F_{K_1}. The concatenation of the last value R_{nb} and the number of blocks nb encoded as an ℓ-bit block is processed by a pseudorandom permutation function F_{K_2} with a different key K_2. Then the output values denoted h_i are bitwise Xored to give the value τ.

- The **Verification algorithm** MXS.ver takes as inputs the key $K \in \mathsf{MXS.KS}$, the document D and the tag $t := (r, \tau)$. It re-computes the value τ from the inputs r and D. It returns 1 if this value is equal to the input τ and 0 otherwise.
- The **Update algorithm** MXS.upd takes as inputs the key $K \in \mathsf{MXS.KS}$, the document the operation $op \in \mathsf{OpCodes}$ where $\mathsf{OpCodes} = \{\mathtt{I}, \mathtt{D}\}$, the set of argument arg and the tag t. The argument arg is composed by the position i where the block value has to be inserted or deleted and the new document block $x \in \mathsf{MXS.BS}$ to insert or ε if it is a delete operation: $\text{arg} = \langle i, x \rangle$.

Contrary to XS, the MXS scheme is based on two pseudorandom functions F_{K_1} and F_{K_2} with independent keys but it does not rely on a pseudorandom permutation. The details of the MXS scheme is given in Figure 5.16.

Complexity. The modified Xor-Scheme is slightly slower than the original one. For the tag and the incremental operations the P call is removed but a call to the function F_{K_2} is added as shown in Figure 5.21. The delete D and insert I operations are slightly slower because of the last block update: the last value R_i depending on the document length has to be removed and the a new value R'_i with the new document length nb' is added.

Other Solutions. In the original Xor-Scheme (Figure 5.15), the document length can be added differently in the algorithm (but still with a random value r_{nb}):

1 Before the last operation $\mathsf{P}_2(K_2, \Sigma)$, an intermediate operation $\mathsf{P}_3(K_3, r_{\mathsf{nb}} || \mathsf{nb} \,||\, \Sigma)$ can be added such that $\tau = \mathsf{P}_2[K_2, \mathsf{P}_3(K_3, r_{\mathsf{nb}} || \mathsf{nb} \,||\, \Sigma)]$.

2 The block length can be processed individually as a last block such that $\tau = \mathsf{P}_2[K_2, \mathsf{P}_3(K_3, r_{\mathsf{nb}} || \mathsf{nb}) \oplus \Sigma]$.

5.2.3.2 Security Proof

The security proof follows the proof strategy used in [BGR95a] for proving the security of XMAC.

Information theoretic case. As in [BGR95a], we first consider the case where the two underlying PRFs F_{K_1} and F_{K_2} are replaced by two truly random functions f_1 and f_2 from $\{0,1\}^{2\ell}$ to $\{0,1\}^L$. We consider an unbounded adversary and the following theorem claims the security of this modified scheme in the information theoretic case. More precisely, it provides an absolute bound on the success of the adversary in terms of the number of oracle queries it makes.

Theorem 5.2.5. *Let $\mathcal{A}$ be any (computationally unbounded) an* IUF-BS*-adversary making a (q_t, q_v, q_{upd})-attack against the Modified Xor-Scheme with two functions picked uniformly at random from $\mathcal{F}^*(\mathcal{D}, \mathcal{R})$ such that $\mathcal{D} = \{0,1\}^{2\ell}$ and $\mathcal{R} = \{0,1\}^L$. The probability that $\mathcal{A}$ is successful is at most*

$$\mathbf{Adv}^{\mathsf{IUF\text{-}BS}}_{\mathsf{MXS}}(\mathcal{A}) \leq q^2 \cdot 2^{\mathsf{n}-\ell} + q_v \cdot (\mathsf{nb}^2 \cdot 2^{\mathsf{n}-\ell} + 2^{-L}).$$

Here, q_t is an upper bound on the number of tagging queries, q_{upd} is an upper bound on the number of update queries (such that $q = q_t + q_{upd}$), q_v is an upper bound on the number of verification queries and nb *denotes the maximal block-length of the documents authenticated in the security game.*

Theorem 5.2.5 (Sketch.) The proof follows closely the proof from [BGR95a]. The main difference is that we use two different random functions in the modified scheme and that we need the following simple lemma to prove that some specific matrix (close to the one used in [BGR95a]) is full-rank. For the reader familiar with [BGR95a], we use similar notations in the following.

Lemma 5.2.6. *Let X be some finite set and let $\mathsf{nb} \in \mathbb{N}$. Let $(R_0, R_1, \ldots, R_{\mathsf{nb}}) \in X^{\mathsf{nb}+1}$ with $R_i \neq R_j$ for all $i \neq j$ then if there exists $(R_0^*, R_1^*, \ldots, R_{\mathsf{nb}}^*) \in X^{\mathsf{nb}+1}$ such that*

$$\{(R_0, R_1), (R_1, R_2), \ldots (R_{\mathsf{nb}-1}, R_{\mathsf{nb}})\} = \{(R_0^*, R_1^*), (R_1^*, R_2^*), \ldots (R_{\mathsf{nb}-1}^*, R_{\mathsf{nb}}^*)\}$$

then for all $i \in \{0, \ldots, \mathsf{nb}\}$, $R_i = R_i^*$.

Lemma 5.2.6. This lemma can be easily proved by induction over nb. Let us denote S_{nb} the first set $\{(R_0, R_1), (R_1, R_2), \ldots (R_{\mathsf{nb}-1}, R_{\mathsf{nb}})\}$ where all R_i are distinct. In particular, the set S_{nb} contains exactly nb different couples. One can notice that the first member of each couple is the second member of the previous couple except the first and the last couples. In others words a value R_i appears in two couples: once as a first member and once as a second member except the first one R_0 and the last one R_{nb}.
The case $\mathsf{nb} = 1$ is trivial. We consider the case $\mathsf{nb} = 2$ that provides greater clarity. Let assume that there exists $(R_0^*, R_1^*, R_2^*) \in X^3$ such that

$$\{(R_0, R_1), (R_1, R_2)\} = \{(R_0^*, R_1^*), (R_1^*, R_2^*)\}$$

and $\#\{R_0, R_1, R_2\} = 3$. As there are exactly two couples in each set, we have the following two cases:

- **case 1:** $(R_0, R_1) = (R_0^*, R_1^*)$ and $(R_1, R_2) = (R_1^*, R_2^*)$ then in this case, we get $R_0 = R_0^*, R_1 = R_1^*, R_2 = R_2^*$;
- **case 2:** $(R_0, R_1) = (R_1^*, R_2^*)$ and $(R_1, R_2) = (R_0^*, R_1^*)$. The first equality implies $R_1^* = R_0$ and the second equality implies $R_1^* = R_2$ and thus $R_0 = R_2$ which contradicts the statement $R_0 \neq R_2$.

Suppose now that Lemma 5.2.6 holds for all integers $k \leq \mathsf{nb} - 1$ for some positive integer nb. We will show that it holds for nb.

Let us suppose that there exists $(R_0^*, R_1^*, \ldots, R_{\mathsf{nb}}^*) \in X^{\mathsf{nb}+1}$ such that $S_{\mathsf{nb}} = S_{\mathsf{nb}}^*$ where S_{nb}^* is the set $\{(R_0^*, R_1^*), (R_1^*, R_2^*), \ldots (R_{\mathsf{nb}-1}^*, R_{\mathsf{nb}}^*)\}$. Again, as all the values R_i are different in S_{nb} then the n couples are different. The equality of these two sets S_{nb} and S_{nb}^* implies that they contain exactly the same nb couples and that in each set a couple appears only once. We have the following two cases:

- **case 1:** $(R_{\mathsf{nb}-1}, R_{\mathsf{nb}}) = (R_{\mathsf{nb}-1}^*, R_{\mathsf{nb}}^*)$ and $S_{\mathsf{nb}-1} = S_{\mathsf{nb}-1}^*$. From the induction hypothesis, for all $i \in \{0, \ldots, n-1\}$, $R_i = R_i^*$.
- **case 2:** $(R_{\mathsf{nb}-1}, R_{\mathsf{nb}}) \neq (R_{\mathsf{nb}-1}^*, R_{\mathsf{nb}}^*)$ Then there exists $i \in \{0, \ldots, \mathsf{nb} - 1\}$ such that $(R_{\mathsf{nb}-1}, R_{\mathsf{nb}}) = (R_{i-1}^*, R_i^*)$. It implies $R_i^* = R_n$ and according to the structure of these sets, there is a couple in S_{nb}^* that has a first member equal to $R_i^* = R_{\mathsf{nb}})$ and it has to be the case in S_{nb}. But as mentioned above, R_{nb} is a value that appears only in one couple of S_{nb} and we get a contradiction.

□

We will use this lemma with $X = \{0,1\}^\ell$ at the end of the proof to show that different messages of the same block-length involve different input pairs for the underlying PRF F_{K_1}.

Since the adversary $\mathcal{A}$ is computationally unbounded we may assume without loss of generality that it is deterministic. The probabilistic choices in $\mathcal{A}$'s attack on the scheme are thus the initial choice of f_1 and f_2 of the random functions in $\mathcal{F}^*(\{0,1\}^{2\ell}, \{0,1\}^L)$ and the choices of random coins made by the authentication oracles in the security game. We assume (again without loss of generality) that $\mathcal{A}$ makes exactly $q = q_t + q_{upd}$ authentication queries (either as a direct tagging query or as an update query with the insert or the delete operations). As in [BGR95a], there is no loss of generality to assume that $\mathcal{A}$ makes all its authentication queries and then makes exactly one verify query (for its purported forgery). We prove that in this case the probability of the event (denoted Succ) $\mathcal{A}$'s forgery is valid is upper-bounded by

$$q^2 \cdot 2^{\mathsf{n}-\ell} + \mathsf{nb}^2 \cdot 2^{\mathsf{n}-\ell} + 2^{-L}.$$

and using a classical argument (see e.g. [BGR95a]) we get the claimed bound for general adversaries.

We consider the simple case where all the random coins used in the last block of each authenticated document are different. Note that in all authentication queries (from a tagging query or an update query), this random block is picked uniformly at random and independently of the previous blocks. To analyse the probability of this event (denoted $\mathsf{Distinct}$), we can therefore use the following simple lemma:

Lemma 5.2.7 ([BGR95a, Fact A.1])**.** *Let $P(m,t)$ denote the probability of at least one collision in the experiment of throwing t balls, independently at random, into m buckets. Then $P(m,t) \leq t^2/m$.*

We thus have

$$\begin{aligned}
\Pr[\mathsf{Succ}] &= \Pr[\mathsf{Succ}|\mathsf{Distinct}] \cdot \Pr[\mathsf{Distinct}] + \Pr[\mathsf{Succ}|\overline{\mathsf{Distinct}}] \cdot \Pr[\overline{\mathsf{Distinct}}] \\
&\leq \Pr[\mathsf{Succ}|\mathsf{Distinct}] + \Pr[\overline{\mathsf{Distinct}}] \\
&\leq \Pr[\mathsf{Succ}|\mathsf{Distinct}] + P(2^{\mathsf{n}-\ell}, q) \\
&\leq \Pr[\mathsf{Succ}|\mathsf{Distinct}] + q^2 \cdot 2^{\mathsf{n}-\ell}.
\end{aligned}$$

and it remains to upper-bound $\Pr[\mathsf{Succ}|\mathsf{Distinct}]$.

Let us fix a particular sequence of q documents $D_1, \ldots, D_q$ (each made of at most nb blocks of n bits) corresponding to all documents authenticated in the security game by some authentication queries (either as a tagging query or as an update query with the insert or the delete operations). We also fix $r_1, \ldots, r_q$ some bit-strings possibly used as random values in the modified Xor-Scheme for these documents (*i.e.* r_i consists in $1 \leq \mathsf{nb}_i \leq \mathsf{nb}$ blocks of $\ell - \mathsf{n}$ bits if D_i is made of nb_i blocks of n bits) and we assume that the last blocks of all of them are all different. Finally we fix $\tau_1, \ldots, \tau_q$ some possible corresponding tags in $\{0,1\}^L$ for these documents. We consider only bit-strings $(D_1, \ldots, D_q)$, $(r_1, \ldots, r_q)$ and $(\tau_1, \ldots, \tau_q)$ for which the probability that there exists two functions f_1 and f_2 such that $t_i = (r_i, \tau_i)$ is a valid tag for D_i (for all $i \in \{1, \ldots, q\}$) for f_1 and f_2 is non-zero.

We will compute the probability of the event that $\mathcal{A}$'s forgery is valid conditioned on the event that the authentication queries made by $\mathcal{A}$ are on the documents $D_1, \ldots, D_q$, use the random coins $(r_1, \ldots, r_q)$ and result in the tags $(\tau_1, \ldots, \tau_q)$. More precisely, we will show that this probability is upper-bounded by $\mathsf{nb}^2 \cdot 2^{\mathsf{n}-\ell} + 2^{-L}$ (and since the bit-strings $(D_1, \ldots, D_q)$, $(r_1, \ldots, r_q)$ and $(\tau_1, \ldots, \tau_q)$ are arbitrary, we will get the result by standard conditioning arguments).

We consider a possible forgery output by $\mathcal{A}$ and we denote D_{q+1} the corresponding document, r_{q+1} the used randomness and τ_{q+1} the tag. It is worth noting that the pair (D_{q+1}, r_{q+1}) is different from all pairs (D_i, r_i) for all $i \in \{1, \ldots, q\}$ (since otherwise, this is not an actual forgery) but we cannot assume that the last block of r_{q+1} is different from the last blocks of all previous random values r_i for all $i \in \{1, \ldots, q\}$ (since $\mathcal{A}$ may choose it arbitrarily and it can reuse a value obtained in a previous authentication query).

For $i \in \{1, \ldots, q+1\}$, we denote $D_i[j]$ for all $j \in \{1, \ldots, \mathsf{nb}_i\}$, the j-th block of the document D_i and similarly $r_i[j]$ for all $j \in \{1, \ldots, \mathsf{nb}_i + 1\}$, the j-th block of the randomness r^i. As in [BGR95a], we consider the matrix B with $q+1$ rows and $2^{2\ell+1}$ columns over $\mathbb{F}_2 = \{0, 1\}$ where the entry in row $i \in \{1, \ldots, q+1\}$ and column $j \in \{1, \ldots, 2^{2\ell+1}\}$ is defined as follows:

- for $j \in \{1, \ldots, 2^{2\ell}\}$, the entry is equal to 1 if j is the index of the 2ℓ-bit string $(D_i[\mathsf{nb}_i]||r_i[\mathsf{nb}_i]||\mathsf{nb}_i||r_i[\mathsf{nb}_i + 1])$ in lexicographic order (and 0 otherwise).

- for $j \in \{2^{2\ell}+1, \ldots, 2^{2\ell+1}\}$, the entry is equal to 1 if $j - 2^{2\ell}$ is the index of the 2ℓ-bit string $(D_i[k]||r_i[k]||D_i[k+1]||r_i[k+1])$ in lexicographic order for some $k \in \{1, \ldots, \mathsf{nb}_i - 1\}$ (and 0 otherwise).

In other words, the matrix B contains a 1 on the row i for $i \in \{1, \ldots, q+1\}$ only at positions corresponding to bit-strings of length 2ℓ used as inputs to the random functions f_1 and f_2 in the modified Xor-Scheme (where the left part consisting of the first $2^{2\ell}$ columns of the matrix corresponds to the unique input of f_2 and the right part corresponds to all inputs to f_1).

We have the following lemma:

Lemma 5.2.8. *The matrix B has full rank with probability at least $1 - \mathsf{nb}^2 \cdot 2^{\mathsf{n}-\ell}$.*

Lemma 5.3.3. The proof is similar to the proof of [BGR95a, Lemma A.3]. If the pair $(\mathsf{nb}_{q+1}, r_{q+1}[\mathsf{nb}_{q+1}+1])$ is different from all $(\mathsf{nb}_i, r_i[\mathsf{nb}_i+1])$ for $i \in \{1, \ldots, q\}$, then the matrix B is in echelon form (in its left part) and is thus trivially of full rank.

Otherwise, we assume that $r_{q+1}[\mathsf{nb}_{q+1}+1]$ is equal to some $r_i[\mathsf{nb}_i+1]$ and if $\mathsf{nb}_{q+1} = \mathsf{nb}_i$ (the last block of randomness of $\mathcal{A}$'s forgery is equal to the last block of randomness of the i-th authenticated message and the block-length of these two messages are equal). It is worth noting that there exists only one index $i \in \{1, \ldots, q\}$ such that this is the case (since we assume that these last blocks of randomness are all different). For this i-th document, the random blocks $r_i[j]$ for $j \in \{1, \ldots, \mathsf{nb}_i\}$ are all different with probability at least $1 - \mathsf{nb}_i^2 \cdot 2^{\mathsf{n}-\ell} \geq 1 - \mathsf{nb}^2 \cdot 2^{\mathsf{n}-\ell}$ by Lemma 5.2.7. Since the pair (D_{q+1}, r_{q+1}) is different from (D_i, r_i) and since the pairs $(D_i[k], r_i[k])$ are all different for $k \in \{1, \ldots, \mathsf{nb}_i\}$ (with probability at least $1 - \mathsf{nb}^2 \cdot 2^{\mathsf{n}-\ell}$), we can apply Lemma 5.2.6 to the sets (of the same length $\mathsf{nb}_i = \mathsf{nb}_{q+1}$):

$$\{D_i[1]||r_i[1]||D_i[2]||r_i[2], D_i[2]||r_i[2]||D_i[3]||r_i[3], \ldots, D_i[\mathsf{nb}_i - 1]||r_i[\mathsf{nb}_i - 1]||D_i[\mathsf{nb}_i]||r_i[\mathsf{nb}_i]\}$$

and

$$\{D_{q+1}[1]||r_{q+1}[1]||D_{q+1}[2]||r_{q+1}[2], D_{q+1}[2]||r_{q+1}[2]||D_{q+1}[3]||r_{q+1}[3], \ldots, \\ D_{q+1}[\mathsf{nb}_i - 1]||r_{q+1}[\mathsf{nb}_i\text{-}1]||D_{q+1}[\mathsf{nb}_i]||r_{q+1}[\mathsf{nb}_i]\}.$$

We thus obtain that there exists an index $k \in \{1, \ldots, \mathsf{nb}_i - 1\}$ such that

$$(D_{q+1}[k]||r_{q+1}[k]||D_{q+1}[k+1]||r_{q+1}[k+1]) \neq (D_i[k]||r_i[k]||D_i[k+1]||r_i[k+1]).$$

Therefore in this case the left part of the last row (consisting of the first $2^{2\ell}$ columns) is identical to the left part of the i-th row but these rows differ in at least one position in the right part of the matrix B. By elementary operations on the rows, one can easily transform the matrix B in echelon form and it is therefore of full rank (with probability at least $1 - \mathsf{nb}^2 \cdot 2^{\ell - \mathsf{n}}$). □

To conclude the proof, one can identify the functions f_1 and f_2 to their vector of values in $(\{0,1\}^{2\ell})^L$ by denoting $f_i(x) = (\varphi_{i,1}^{(x)}, \ldots, \varphi_{i,L}^{(x)})$ for $x \in \{0,1\}^{2\ell}$ and $i \in \{1,2\}$, where $\varphi_{i,j} \in \{0,1\}^{2\ell}$ for $i \in \{1,2\}$ and $j \in \{1, \ldots, L\}$. In this case by construction, τ^i is the authentication tag of the document D_i with randomness r^i for all $i \in \{1, \ldots, q+1\}$ if and only if for all $j \in \{1, \ldots, L\}$, the j-th bit $\tau_i^{(j)}$ of τ_i is equal to the dot product of the i-th row of the matrix B and the vector $\varphi_{2,i}||\varphi_{1,i}$. Using the same argument as in [BGR95a], since B is of full rank, the number of vectors satisfying this $q+1$ equations is 2^L times smaller than the number of vectors satisfying only the first q equations (corresponding to the first q rows of B), and therefore we obtained that the forgery τ^{q+1} output by the adversary is valid with probability 2^{-L} if the matrix B is full rank.

We have thus proved that, in the simplified case, the probability that $\mathcal{A}$'s forgery is valid is upper-bounded by

$$q^2 \cdot 2^{\mathsf{n}-\ell} + \mathsf{nb}^2 \cdot 2^{\mathsf{n}-\ell} + 2^{-L}.$$

and thus the claimed bound for general adversaries. □

Computational case. If we replace the (truly) random functions by pseudorandom functions in the previous result, we obtain readily the following computational security result:

Theorem 5.2.9. *Let* $\mathsf{F} = (\mathsf{KS}, \mathsf{Dom}, \mathsf{Rng}, \mathsf{eval})$ *be a pseudorandom function family such that* $\mathsf{F.Dom} = \{0,1\}^{2\cdot\ell}$ *and* $\mathsf{F.Rng} = \{0,1\}^L$. *Let* $\mathcal{A}$ *be any adversary making a* (q_t, q_v, q_{upd})*-attack against the modified Xor-Scheme with two functions picked uniformly at random from* F *and running in time* λ.
There exist an adversary $\mathcal{B}$ *against the pseudorandomness property of* F *that makes* $q' = q \cdot t$ *queries to* F, *runs in time* $\lambda' = \lambda + O(q'(\ell + L))$ *such that*

$$\mathbf{Adv}^{\mathsf{IUF\text{-}BS}}_{\mathsf{MXS}}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{prf}}_{\mathsf{F}}(\mathcal{B}) + (q^2 + \mathsf{nb}^2) \cdot 2^{\mathsf{n}-\ell} + 2^{-L}.$$

Here, q_t *is an upper bound on the number of tagging queries,* q_{upd} *is an upper bound on the number of update queries (such that* $q = q_t + q_{upd}$*),* q_v *is an upper bound on the number of verification queries and* nb *denotes the maximal block-length of the documents authenticated in the security game.*

Theorem 5.2.9. The proof is identical to the proof of [BGR95a, Theorem 4.2] and is left to the reader. □

$\underline{\mathsf{MXS.kg}()}$
$K_1 \twoheadleftarrow \mathsf{F.KS}$; $K_2 \twoheadleftarrow \mathsf{F.KS}$
return $K_1||K_2$

$\underline{\mathsf{MXS.tag}(K, D)}$
$(K_1, K_2) \leftarrow K$; $\Sigma \leftarrow 0^L$, $\mathsf{nb} \leftarrow |D|_{\mathsf{XS.n}}$
$r_0 \twoheadleftarrow \mathsf{MXS.RS}$
$r \leftarrow r_0$
$R_0 \leftarrow r_0||D[0]$
for $i = 1$ **to** nb-1 **do**
  $r_i \twoheadleftarrow \mathsf{MXS.RS}$
  $r \leftarrow r||r_i$
  $R_i \leftarrow r_i||D[i]$
  $h_i \leftarrow \mathsf{F}(K_1, R_{i-1}||R_i)$
  $\Sigma \leftarrow \Sigma \oplus h_i$
$r_{\mathsf{nb}} \twoheadleftarrow \mathsf{MXS.RS}$
$R_{\mathsf{nb}} \leftarrow r_{\mathsf{nb}}||\mathsf{nb}$
$h_{\mathsf{nb}} \leftarrow \mathsf{F}(K_2, R_{\mathsf{nb}-1}||R_{\mathsf{nb}})$
$\tau \leftarrow \Sigma \oplus h_{\mathsf{nb}}$
$t \leftarrow (r, \tau)$
return t

$\underline{\mathsf{MXS.ver}(K, D, t)}$
$(K_1, K_2) \leftarrow K$; $\Sigma \leftarrow 0^L$; $\mathsf{nb} \leftarrow |D|_{\mathsf{XS.n}}$
$r \leftarrow t[0 : \mathsf{nb} \cdot \mathsf{rl}\text{-}1]$; $\tau \leftarrow t[\mathsf{nb} \cdot \mathsf{rl} :]$
$R_0 \leftarrow D[0]||r_0$
for $i = 1$ **to** nb-1 **do**
  $R_i \leftarrow r_i||D[i]$
  $h_i \leftarrow \mathsf{F}(K_1, R_{i\text{-}1}||R_i)$
  $\Sigma \leftarrow \Sigma \oplus h_i$
$R_{\mathsf{nb}} \leftarrow \mathsf{nb}||r_{\mathsf{nb}}$
$h_{\mathsf{nb}} \leftarrow \mathsf{F}(K_2, R_{\mathsf{nb}\text{-}1}||R_{\mathsf{nb}})$
$\tau \leftarrow \Sigma \oplus h_{\mathsf{nb}}$
return $\tau' = \tau$

$\underline{\mathsf{MXS.upd}(K, D, op, \langle i, x\rangle, t)}$
if $i >$ nb **then return** $\perp$
$K_1||K_2 \leftarrow K$; $\mathsf{nb} \leftarrow |D|_{\mathsf{MXS.n}}$
$r \leftarrow t[0 : \mathsf{nb} \cdot \mathsf{rl}\text{-}1]$; $\tau \leftarrow t[\mathsf{nb} \cdot \mathsf{rl} :]$
$R_i \leftarrow r_i||D[i]$
$R_{i\text{-}1} \leftarrow r_i||D[i\text{-}1]$
$R_{\mathsf{nb}\text{-}1} \leftarrow r_{\mathsf{nb}\text{-}1}||D[\mathsf{nb}\text{-}1]$
$R_{\mathsf{nb}} \leftarrow r_{\mathsf{nb}}||\mathsf{nb}$
$h_{\mathsf{nb}} \leftarrow \mathsf{F}(K_2, R_{\mathsf{nb}\text{-}1}||R_{\mathsf{nb}})$
$h_i \leftarrow \mathsf{F}(K_1, R_{i\text{-}1}||R_i)$
if $op = \mathtt{I}$ **then**
  $r'_i \twoheadleftarrow \mathsf{MXS.RS}$
  $R'_i \leftarrow r'_i||x$
  $r'_{\mathsf{nb}+1} \twoheadleftarrow \mathsf{MXS.RS}$
  $R'_{\mathsf{nb}+1} \leftarrow r'_{\mathsf{nb}+1}||\mathsf{nb}{+}1$
  $h'_{\mathsf{nb}+1} \leftarrow \mathsf{F}(K_2, R_{\mathsf{nb}\text{-}1}||R'_{\mathsf{nb}+1})$
  $h'_i \leftarrow \mathsf{F}(K_1, R_{i\text{-}1}||R'_i)$
  $h'_{i+1} \leftarrow \mathsf{F}(K_1, R_{i'}||R_{i+1})$
  $\Sigma \leftarrow h'_i \oplus h'_{i+1} \oplus h_i \oplus h_{\mathsf{nb}} \oplus h'_{\mathsf{nb}+1}$
  $r \leftarrow r_0 \ldots r_{i-1}||r'_i||r_i \ldots r_{\mathsf{nb}-1}||r'_{\mathsf{nb}+1}$
if $op = \mathtt{D}$ **then**
  $R_{i+1} \leftarrow r_{i+1}||D[i{+}1]$
  $h_{i+1} \leftarrow \mathsf{F}(K_1, R_i||R_{i+1})$
  $h'_i \leftarrow \mathsf{F}(K_1, R_{i-1}||R_{i+1})$
  $r'_{\mathsf{nb}} \twoheadleftarrow \mathsf{MXS.RS}$
  $R'_{\mathsf{nb}} \leftarrow r'_{\mathsf{nb}}||\mathsf{nb}\text{-}1$
  $h'_{\mathsf{nb}} \leftarrow \mathsf{F}(K_2, R_{\mathsf{nb}-1}||R'_{\mathsf{nb}})$
  $\Sigma \leftarrow h_i \oplus h_{i+1} \oplus h'_i \oplus h_{\mathsf{nb}} \oplus h'_{\mathsf{nb}}$
  $r \leftarrow r_0 \ldots r_{i-1}||r_{i+1} \ldots r_{\mathsf{nb}-1}||r'_{\mathsf{nb}}$
$\tau \leftarrow \tau \oplus \Sigma$
$t \leftarrow (r, \tau)$
return (t)

Figure 5.22: Modified Xor-Scheme Algorithm.

5.3 Incremental MACs with IUF2 Security

5.3.1 XMAC Constructions

Security analysis. As shown by the adversary $\mathcal{A}$ described below, XMACC it not IUF2-SD. This adversary can be easily adapted for XMACR.

Attack. We give the description of an adversary $\mathcal{A}$ against the IUF2-SD security of XMAC scheme.

1. $\mathcal{A}$ calls the Initialize oracle.
2. $\mathcal{A}$ builds the 3-blocks document D_1 such that $D_1 = D[1]||D[2]||D[3]$ for any document blocks.
3. A calls the Tag oracle and gives as inputs the nonce value 0 and the document D_1. It receives the tag t_1.
4. $\mathcal{A}$ builds the document D_2 such that $D_1^* = D[1]||D[2]||D[2]$.
5. $\mathcal{A}$ calls the Update oracle giving as inputs the nonce value 1, the document D_1^*, the operation R, the argument $\langle D[2], 3\rangle$ and the tag t_1. It receives the tag t_2.
6. $\mathcal{A}$ builds $D_2 = D[1]||D[2]||D[2]$ and calls the Finalize oracle with (D_2, t_2) as input.

During the Tag call, the document $D_1 = D[1]||D[2]||D[3]$ is added to the list DL. An during the Update call, the document $D_2 = D[1]||D[3]||D[3]$ is added to the list DL because the document is updated according the current document which was D_1 in this case. Then we can see that the document $D_2 = D[1]||D[2]||D[2]$ provided by the adversary is not in the list DL.

Remark 5.3.1. (Difference between Xor-MAC and Plus-MAC) This attack works *only when the operator of* XMAC *is a "XOR"*, it does not work with the modular addition. In fact, during the *replace* operation, the image by F of the deleted block x_d at position i and the image by F of the inserted block x_i at position i are XORed: removing a block at some position and deleting a block at the same position are done by the same operation "XOR" which in not the case for the modular addition: adding "+" and removing "-" are not done by the same operation.

The previous attack does not break the IUF2-SD security, there is an other adversary that breaks it: the following adversary $\mathcal{B}$.

1. $\mathcal{B}$ calls the Initialize oracle.
2. $\mathcal{B}$ builds the document D_1 such that $D_1 = D[1]||D[2]$ where $D[1]$ and $D[2]$ are any document block. $\mathcal{B}$ calls the Tag oracle giving as input the document D_1. It receives the tag t_1.
3. $\mathcal{B}$ builds the document D_1^* such that $D_1^* = D[1]'||D[2]$ and calls the Update oracle giving as input the operation R, the document D_1^*, the argument $\langle D[1]'', 1\rangle$ and the tag t_1. It receives the tag t_2.

4. $\mathcal{B}$ calls the UPDATE oracle giving as inputs the document D_1, the operation R, the argument $\langle D[1]', 1\rangle$ and the tag t_2. It receives the tag t_3.
5. $\mathcal{B}$ calls once again the UPDATE oracle giving as input the document D_1, the operation R, the argument $\langle D[2]', 2\rangle$ and the tag t_3. It receives the tag t_4.
6. $\mathcal{B}$ calls the FINALIZE oracle giving as input the document D^* such that $D^* = D[1]''||D[2]'$ and the tag t_4.

The document list DL is filled as follows: [2] during TAG call, the document $D_1^{\mathrm{DL}} = D[1]||D[2]$; [3] during the first UPDATE call the document $D_2^{\mathrm{DL}} = D[1]''||D[2]$; [4] during the second UPDATE call, the document $D_3^{\mathrm{DL}} = D[1]'||D[2]$ and [5] during the last UPDATE call, the document $D_4^{\mathrm{DL}} = D[1]'||D[2]'$ is added to the list. The document D^* is not in the list DL. A game where the tag and the document can be tampered with is a strong security game that is hard to achieve.

5.3.2 MXS Construction

In this section we prove that the MXS scheme achieves IUF2-SD-security. The security proof follows closely the IUF-BS-security proof of MXS (Theorem 5.2.5 and Theorem 5.2.9).

INFORMATION THEORETIC CASE. As above, we first consider again the case where the two underlying PRFs F_{K_1} and F_{K_2} are replaced by two truly random functions f_1 and f_2 from $\{0,1\}^{2\ell}$ to $\{0,1\}^L$. We consider an unbounded adversary and the following theorem claims the security of this modified scheme in the information theoretic case.

Theorem 5.3.2. *Let $\mathcal{A}$ be any (computationally unbounded) IUF2-SD-adversary, in the single-document setting, making a $(1, q_v, q_{upd})$-attack against the MXS with two functions picked uniformly at random from $\mathcal{F}^*(\mathcal{D}, \mathcal{R})$ such that $\mathcal{D} = \{0,1\}^{2\ell}$ and $\mathcal{R} = \{0,1\}^L$. The probability that $\mathcal{A}$ is successful is at most*

$$\mathbf{Adv}^{\mathsf{IUF2\text{-}SD}}_{\mathsf{MXS}}(\mathcal{A}) \leq q^2 \cdot 2^{\mathsf{n}-\ell} + q_v \cdot (\mathsf{nb}^2 \cdot 2^{\mathsf{n}-\ell} + \cdot 2^{-L}).$$

Here the number of tagging queries is exactly 1 (single-document setting), q_{upd} is an upper bound on the number of update queries (such that $q = 1 + q_{upd}$), q_v is an upper bound on the number of verification queries and nb *denotes the maximal block-length of the documents authenticated in the security game.*

Theorem 5.3.2 (Sketch.) As in the proof of Theorem 5.2.5, since the adversary $\mathcal{A}$ is computationally unbounded we may assume without loss of generality that it is deterministic. The probabilistic choices in $\mathcal{A}$'s attack on the scheme are thus the initial choice of f_1 and f_2 of the random functions in $\mathcal{F}^*(\{0,1\}^{2\ell}, \{0,1\}^L)$ and the choices of random coins made by the authentication oracles in the security game. We assume (again without loss of generality) that $\mathcal{A}$ makes exactly $q = q_t + q_{upd}$ authentication queries (either as a direct tagging query or as an update query with the insert or the delete operations). Again, there is no loss of generality to assume that $\mathcal{A}$ makes all its authentication queries and then makes exactly one verify query (for its purported forgery). We prove that in this case the probability of the event (denoted Succ) $\mathcal{A}$'s forgery is valid is upper-bounded by

$$q^2 \cdot 2^{\mathsf{n}-\ell} + \mathsf{nb}^2 \cdot 2^{\mathsf{n}-\ell} + 2^{-L}.$$

We consider again the event (denoted Distinct) where all the random coins used in the last block of each authenticated document are different. We have

$$\Pr[\mathsf{Succ}] \leq \Pr[\mathsf{Succ}|\mathsf{Distinct}] + q^2 \cdot 2^{\mathsf{n}-\ell}.$$

and it remains to upper-bound $\Pr[\mathsf{Succ}|\mathsf{Distinct}]$.

A priori, we cannot consider a sequence of q documents $D_1, \ldots, D_q$ corresponding to all documents authenticated in the security game by some authentication queries since the adversary can query the UPDATE oracle on invalid tags and therefore obtain invalid tags. However, we can adapt the previous argument in the IUF2-SD setting.

After querying the TAG oracle on a first document D_1, the adversary will query the UPDATE oracle times on several documents $D_2^*, \ldots, D_q^*$ (where D_j^* is made of nb_j blocks of n bits for $j \in \{2, \ldots, q\}$), with corresponding claimed tags $t_2^*, \ldots, t_q^*$ and some operation (deletion or insertion) on the blocks $i_2, \ldots, i_q$ (respectively) with $t_j^* = (r_j^*, \tau_j^*)$ and $i_j \leq \mathsf{nb}_j$ for $j \in \{2, \ldots, q\}$.

The documents $D_2^*, \ldots, D_q^*$, the randomness $r_2^*, \ldots, r_q^*$ and the tags $\tau_2^*, \ldots, \tau_q^*$ are chosen arbitrarily by the adversary. After the query to the UPDATE oracle, the adversary will obtain a new tag $t_j = (r_j, \tau_j)$ such that r_j is modified in its last block and one block of randomness is added or removed (depending of the performed operation) and $\tau_j = \tau_j^* \oplus X_j$ where X_j depends (at most) on the documents blocks $D_j^*[i-1]$, $D_j^*[i]$, $D_j^*[i+1]$ and $D_j^*[\mathsf{nb}_j]$, the random blocks $r_j^*[i-1]$, $r_j^*[i]$ and $r_j^*[i+1]$ and $r_j^*[\mathsf{nb}_j]$ and the new last block of r_j. Since the queried tags are not verified, we can assume without loss of generality that the adversary queries always use values $\tau_i^* = 0^L$ for $j \in \{2, \ldots, q\}$ (and thus that it gets $\tau_j = X_j$ as a part of the response to its query). The value τ_j is then equal to XOR of at most five values of the underlying random functions, namely two values of f_2 and one or three values of f_1:

- If the j-th query for $j \in \{2, \ldots, q\}$, corresponds to insertion at the first position of D_j^* of the block D', the UPDATE will pick uniformly at random two blocks of randomness r' and r'', compute

$$\tau_j = f_1(D'\|r' \mid\mid D_j^*[0]\|r_j^*[0]) \oplus f_2(D_j^*[\mathsf{nb}_j]\|r_j^*[\mathsf{nb}_j] \mid\mid \mathsf{nb}\|r_j^*[\mathsf{nb}_j+1]) \\ \oplus f_2(D_j^*[\mathsf{nb}_j]\|r_j^*[\mathsf{nb}_j] \mid\mid \mathsf{nb}_j+1\|r'')$$

  and output $(r'||r_j^*[0]|| \ldots ||r_j^*[\mathsf{nb}_j]||r'', \tau_j)$ as the new authentication tag.

- If the j-th query for $j \in \{2, \ldots, q\}$, corresponds to insertion at the p-th position of D_j of the block D', the UPDATE will pick uniformly at random two blocks of randomness r' and r'', compute

$$\tau_j = f_1(D_j^*[p]\|r_j^*[p] \mid\mid D'\|r') \oplus f_1(D'\|r' \mid\mid D_j^*[p+1]\|r_j^*[p+1]) \\ \oplus f_1(D_j^*[p]\|r_j^*[p] \mid\mid D_j^*[p+1]\|r_j^*[p+1]) \oplus f_2(D_j^*[\mathsf{nb}_j]\|r_j^*[\mathsf{nb}_j] \mid\mid \mathsf{nb}_j\|r_j^*[\mathsf{nb}_j+1]) \\ \oplus f_2(D_j^*[\mathsf{nb}_j]\|r_j^*[\mathsf{nb}_j] \mid\mid \mathsf{nb}_j+1\|r'')$$

  and output $(r_j^*[1]|| \ldots ||r_j^*[p]||r'||r_j^*[p+1]|| \ldots ||r_j^*[\mathsf{nb}]||r'', \tau_j)$ as the new authentication tag.

- If the j-th query for $j \in \{2, \ldots, q\}$, corresponds to deletion of the first block of D_j, the UPDATE will pick uniformly at random one block of randomness r', compute

$$\tau_j = f_1(D_j^*[0] \| r_j^*[0] \,||\, D_j^*[1] \| r_j^*[1]) \oplus f_2(D_j^*[\mathsf{nb}_j] \| r_j^*[\mathsf{nb}_j] \,||\, \mathsf{nb}_j \| r_j^*[\mathsf{nb}_j{+}1]) \oplus f_2(D_j^*[\mathsf{nb}_j] \| r_j^*[\mathsf{nb}_j] \,||\, \mathsf{nb}_j\text{-}1 \| r')$$

  and output $(r_j^*[1] || \ldots || r_j^*[\mathsf{nb}_j\text{-}1] || r'', \tau_j)$ as the new authentication tag.

- If the j-th query for $j \in \{2, \ldots, q\}$, corresponds to deletion of the p-th block of D_j, the UPDATE will pick uniformly at random one block of randomness r', compute

$$\begin{aligned}\tau_j = {} & f_1(D_j^*[p\text{-}1] \| r_j^*[p\text{-}1] \,\|\, D_j^*[p] \| r_j^*[p]) \oplus f_1(D_j^*[p] \| r_j^*[p] \,||\, D_j^*[p{+}1] \| r_j^*[p{+}1]) \\ & \oplus f_1(D_j^*[p\text{-}1] \| r_j^*[p\text{-}1] \,\|\, D_j^*[p{+}1] \| r_j^*[p{+}1]) \oplus f_2(D_j^*[\mathsf{nb}_j] \| r^*_{\mathsf{nb}_j}, \mathsf{nb}_j \| r_j^*[\mathsf{nb}_j{+}1]) \\ & \oplus f_2(D_j^*[\mathsf{nb}_j] \| r_j^*[\mathsf{nb}_j] \,||\, \mathsf{nb}_j{+}1 \| r'')\end{aligned}$$

  and output $(r_j^*[1] || \ldots || r_j^*[p\text{-}1] || r_j^*[p{+}1] || \ldots || r_j^*[\mathsf{nb}] || r', \tau_j)$ as the new authentication tag.

The adversary is thus given some kind of authentication tags for (respectively):

- the document $(D', D_j^*[1])$ with "randomness" $(r', r_j^*[1])$;
- the document $(D_j^*[p], D', D_j^*[p{+}1])$ with "randomness" $(r_j^*[p], r', r_j^*[p{+}1])$;
- the document $(D_j^*[1], D_j^*[2])$ with "randomness" $(r_j^*[1], r_j^*[2])$;
- the document $(D_j^*[p\text{-}1], D_j^*[p], D_j^*[p+1])$ with "randomness" $(r_j^*[p\text{-}1], r_j^*[p], r_j^*[p{+}1])$.

which are masked by two values of f_2 which depends on $D_j[\mathsf{nb}_j]$ (with one which uses fresh new randomness) instead of just one.

We can now consider the sequence of q documents $D_1, \ldots, D_q$ (the first one made of at most nb blocks of n bits and the other one made of at most 3 blocks of n bits) corresponding to all documents constructed in this way during the security game. We also fix the bit-strings possibly used as random values for these documents and we assume that the last blocks of all of them are all different. Finally we fix $\tau_1, \ldots, \tau_q$ some possible corresponding tags in $\{0,1\}^L$ for these documents.

As in the proof of Theorem 5.2.5, we will compute the probability of the event that $\mathcal{A}$'s forgery is valid conditioned on the event that the authentication queries made by $\mathcal{A}$ corresponds to the constructed documents $D_1, \ldots, D_q$, use the specified random coins and result in the tags $(\tau_1, \ldots, \tau_q)$. More precisely, we will show that this probability is upper-bounded by $\mathsf{nb}^2 \cdot 2^{\mathsf{n}-\ell} + 2^{-L}$ (and since the bit-strings $(D_1, \ldots, D_q)$, $(r^1, \ldots, r^q)$ and $(\tau_1, \ldots, \tau_q)$ are arbitrary, we will get the result by standard conditioning arguments). We construct again a matrix B containing a 1 on the row i for $i \in \{1, \ldots, q+1\}$ only at positions corresponding to bit-strings of length 2ℓ used as inputs to the random functions f_1 and f_2 (where the left part consisting of the first $2^{2\ell}$ columns of the matrix corresponds to the two[7] inputs of f_2 and the right part corresponds to all inputs to f_1).

[7] There is only one input of f_2 for the first row of the matrix

By our assumption on the distinctness of the random values in the last block, we have the following lemma whose proof is identical to the one of the corresponding lemma in in the proof of Theorem 5.2.5:

Lemma 5.3.3. *The matrix B has full rank with probability at least $1 - \mathsf{nb}^2 \cdot 2^{\mathsf{n}-\ell}$.*

The claimed bound follows from this lemma for general adversaries as in the proof of Theorem 5.2.5.

□

Computational case. If we replace the (truly) random functions by pseudorandom functions in the previous result, we obtain readily the following computational security result:

Theorem 5.3.4. *Let $\mathsf{F} = (\mathsf{KS}, \mathsf{Dom}, \mathsf{Rng}, \mathsf{eval})$ be a pseudorandom function family such that $\mathsf{F.Dom} = \{0,1\}^{2\cdot\ell}$ and $\mathsf{F.Rng} = \{0,1\}^L$. Let $\mathcal{A}$ be any adversary making a $(1, q_v, q_{upd})$-attack against the modified Xor-Scheme with two functions picked uniformly at random from F and running in time λ.*
There exist an adversary $\mathcal{B}$ against the pseudorandomness property of F that makes $q' = q \cdot t$ queries to F, runs in time $\lambda' = \lambda + O(q'(\ell + L))$ such that

$$\mathbf{Adv}^{\mathsf{IUF2\text{-}SD}}_{\mathsf{MXS}}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{prf}}_{\mathsf{F}}(\mathcal{B}) + (q^2 + \mathsf{nb}^2) \cdot 2^{\mathsf{n}-\ell} + 2^{-L}.$$

Here, the number of tagging queries is exactly 1, q_{upd} is an upper bound on the number of update queries (such that $q = 1 + q_{upd}$), q_v is an upper bound on the number of verification queries and nb denotes the maximal block-length of the documents authenticated in the security game.

Proof. (Theorem 5.2.9) The proof is identical to the proof of [BGR95a, Theorem 4.2] and is left to the reader. □

Remark 5.3.5 (Multi-document security)**.** The MXS scheme is IUF2-SD and combining it with the transform **SDtoMD$_1$** given in Section 5.1.4, we have a IUF2-MD secure scheme.

Remark 5.3.6 (Compressed version of MXS)**.** The MXS is a strongly incremental MAC that achieves IUF-BS and IUF2-SD security but it requires to generate an MXS.rl-bit random value per document block and it has an impact on the tag length which is equal to $\mathsf{nb} \cdot \mathsf{MXS.rl} + t\ell$ bits for a an nb-block document. We can modify MXS in order to have smaller tag by using a counter ctr. Let consider a MXS scheme where each block value r is equal to $\mathsf{F}'_{K_3}(ctr)$ where ctr is the counter and has a smaller bit-length ($|ctr| \leq |r|$) and $\mathsf{F}' = (\mathsf{KS}, \mathsf{Dom}, \mathsf{Rng}, \mathsf{eval})$ a pseudorandom family such that $\mathsf{F}'.\mathsf{Dom} = \{0,1\}^{|ctr|}$ and $\mathsf{F}'.\mathsf{Rng} = \{0,1\}^{\mathsf{MXS.rl}}$. In practice, the length MXS.rl should be at least 128 bits whereas a counter of 32 bits in reasonable[8]. Then the tag contains the values of the counter used to process each block (the tag length becomes $\mathsf{nb} \cdot |ctr| + t\ell$). However, this change make the MXS stateful (the counter has to be in the state to be sure that the adversary cannot tamper with it) and requires a third key for F'.

Conclusion. In this chapter, we revisit incrementality for MACs by first, giving a definition of an incremental MAC with a specific syntax in order to encompass their variety and their

[8] Of course here, the number of processed blocks has to be smaller than 2^{32}

complexity. The security notions present in the literature, namely the basic security and the tamper-proof security, are analysed: a security game is provided for the first one whereas tamper-security proof seems to be a more vague and complex notion. This is the reason why we define the security notion IUF2-X, inspired by the tamper-proof security, that is close, in the single (X = SD) and multi-document (X = MD) setting. Another security notion IUF2R-X, were replay attacks are taken into account, reinforces the IUF2-X security notion. The relationships between all those security notions, represented in Figure 5.9, help to analyse the security of incremental MACs.

Moreover, we showed that the XS [BGG94] does not provide the claimed basic security: a forgery can be easily built from any tag by inserting specific document block chains to a legitimate document and the corresponding random value chains to the legitimate random value. We proposed the MXS that is not vulnerable to these attacks and we proved its security not only in the IUF-BS model but also in the IUF2-SD models. From this scheme, a IUF2-MD-secure scheme using the transform $\mathbf{SDtoMD_1}$ can be built[9]. Still from the MXS, a stateful scheme IUF2R-X-secure against replay attack can be built using a counter[10]. The XMAC is also analysed but it provides only basic security; we give attacks in the IUF2-X security model.

The MXS is the only secure strongly incremental algorithm and it is secure in a strong security model but unfortunately it has drawbacks that make it unpractical: it needs to generate a lot of randomness (around one random block for one document block) which slows down the algorithm; and the tag length is much bigger than standard MACs due to the storage of all the random blocks essential for the tag verification (even if it can be modified to have a smaller tag by using a counter as explained in Theorem 5.3.6). An open problem is to find an incremental MAC producing smaller tags and achieving the same security notion. Obviously, the MXS is not suitable as a global MAC (as defined in 2.3.2) due to those drawbacks.

[9] Details given in Theorem 5.3.5
[10] Details given in Theorem 5.1.5

Chapter 5

Chapter 6

Conclusion and Open Questions

6.1 Conclusion

The work presented in this thesis aims at analyzing the security of block-ciphers in the context of Full Disk Encryption. In order to go beyond the barrier of deterministic algorithms, in the very constraint model of Full Disk Encryption, we introduced the concept of *diversifier* which is, in a word, a random value or a nonce provided by the systems for free, in the sense that it was not meant to be used for data protection in the first place and it is not stored.

In addition, we gave the analysis of block-ciphers in a strong security model, the Key-Dependent Message security model, where it is possible to encrypt the encryption key itself or a function of this key. This model is of interest in FDE context; where the encryption key, a part of it or a more complex functions of the key can end up in the disk. Another contribution was the introduction of two methodologies to analyse the block-ciphers security in the KDM setting: the Splitting and Forgetting technique and a KDM adaptation of the well known H-coefficient technique. It gave notably that the Even-Mansour and KAF ciphers with the same key and same permutation/function will never achieve the same security level that the ideal cipher in this context (namely, KDM security for a claw-free set of functions) and this holds for any number of rounds. Therefore, if the keys are independent, KDM security set is less and less constrained when increasing the number of rounds as shown by the analysis of the one-round and two-round Even-Mansour configurations (we conjectured the KDM security of the 3-round configuration).

Knowing the limits of the FDE model, we took into account two models where metadata storage is possible; the ADE model which does not prevent replay attacks and the stronger model FADE that does. These ideas are natural and an ADE mechanism is already implemented. However, using standard MACs to achieve FADE security has a huge impact on performances. The read and write delays on the disk are crucial as they have a direct impact on the whole system and cryptographic operations, encryption/decryption and tag generation/verification, has to be as fast as possible. This is the reason why the incremental MACs were investigated: we revisit existing security notions and give new ones (some of them consider replay attacks) to understand how much incremental MACs can be secure. We also give a basic security attack on the Xor-Scheme which was the only strongly incremental and we provide a patched version. Unfortunately, achieving strong incremental security (stronger than basic security) has a cost in terms of storage.

6.2 Open Questions and futur work

In this section, we would like to hightlight some open problems and most of them can be considered as futur works.

Disk protection mechanisms in practice. The analysis of disk protection was made step by step; encryption (FDE with a diversifier), authenticated encryption for a sector (ADE), authenticated encryption robustness against replay attacks (FADE); but this analysis is only theoretical. It would be interesting to benchmark different implementations, on different devices and with different cryptographic mechanisms. For now, the incremental scheme that seems to be the most suitable is the Merkle tree (even if it is a bad incremental scheme in the sense that the update complexity is logarithmic) where the update and verify times are close.

Question 6.1 (Benchmark)**.** *How much a diversifier and/or a FADE mechanism slows down read and write operations ?*

The benchmark on different devices can be a first step to see if those solutions are viable or not.

Key-Dependent Message Security. The KDM security analysis of the Even-Mansour and the Key-Alternating ciphers were done for some configurations and with one KDM analysis techniques: splitting and forgetting technique and the H-coefficient technique (except the 1-round EM).

Question 6.2 (Technique comparison)**.** *It can be interesting to prove the KDM-security for different schemes and configurations (EM, KAF or others) with both techniques to see if for some cases, one of these methods gives a notably better results.*

Question 6.3 (Minimal KDM-secure configurations of EM and KAF ciphers)**.** *From how many rounds, the EM and the KAF schemes, with a simple configuration (same internal primitives and independent keys), are KDM secure under a claw-free KDM set ?*

More theoretically, we can ask the following questions:

Question 6.4 (Feistel network KDM security)**.** *A more general question is: can we prove the KDM security of the Feistel network where the underlying function are PRFs ? The main difficulty to solve this question is that here there is no information about how the key, for each round, is used.*

Incremental MACs. For now, only MXS construction is proved IUF2-X-secure but it is hard to use it in practice.

Question 6.5 (Better than MXS)**.** *Can we find a scheme that achieves* IUF2-X *security and that is more practical than* MXS *?*

By practical, we mean that this should generate smaller tag, require less random generation even if it is statefull. If it is stateful, the state has to be small.

Question 6.6 (Incremental MACs IUFR-2 suitable for disk protection)**.** *More specifically, can we have an incremental MAC that is an IUFR-2 secure MAC with that has a small tag and a small state ?*

Bibliography

[AB96] Ross J. Anderson and Eli Biham. Two practical and provably secure block ciphers: BEARS and LION. In Dieter Gollmann, editor, *Fast Software Encryption – FSE'96*, volume 1039 of *Lecture Notes in Computer Science*, pages 113–120, Cambridge, UK, February 21–23, 1996. Springer, Heidelberg, Germany.

[ACP+17] Joël Alwen, Binyi Chen, Krzysztof Pietrzak, Leonid Reyzin, and Stefano Tessaro. Scrypt is maximally memory-hard. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017, Part III*, volume 10212 of *Lecture Notes in Computer Science*, pages 33–62, Paris, France, April 30 – May 4, 2017. Springer, Heidelberg, Germany.

[AM13] Kevin Atighehchi and Traian Muntean. Towards fully incremental cryptographic schemes. In Kefei Chen, Qi Xie, Weidong Qiu, Ninghui Li, and Wen-Guey Tzeng, editors, *ASIACCS 13: 8th ACM Symposium on Information, Computer and Communications Security*, pages 505–510, Hangzhou, China, May 8–10, 2013. ACM Press.

[App14] Benny Applebaum. Key-dependent message security: Generic amplification and completeness. *Journal of Cryptology*, 27(3):429–451, July 2014.

[App18] Apple. ios security. Technical report, Apple, November 2018.

[Ati14] Kévin Atighehchi. Space-efficient, byte-wise incremental and perfectly private encryption schemes. Cryptology ePrint Archive, Report 2014/104, 2014. http://eprint.iacr.org/2014/104.

[BAA+08] Lakshmi N. Bairavasundaram, Andrea C. Arpaci-Dusseau, Remzi H. Arpaci-Dusseau, Garth R. Goodson, and Bianca Schroeder. An analysis of data corruption in the storage stack. *TOS*, 4(3):8:1–8:28, 2008.

[BBKN12] Mihir Bellare, Alexandra Boldyreva, Lars R. Knudsen, and Chanathip Namprempre. On-line ciphers and the hash-CBC constructions. *Journal of Cryptology*, 25(4):640–679, October 2012.

[BBO07] Mihir Bellare, Alexandra Boldyreva, and Adam O'Neill. Deterministic and efficiently searchable encryption. In Alfred Menezes, editor, *Advances in Cryptology – CRYPTO 2007*, volume 4622 of *Lecture Notes in Computer Science*, pages 535–552, Santa Barbara, CA, USA, August 19–23, 2007. Springer, Heidelberg, Germany.

[BCG+12] Julia Borghoff, Anne Canteaut, Tim Güneysu, Elif Bilge Kavun, Miroslav Knežević, Lars R. Knudsen, Gregor Leander, Ventzislav Nikov, Christof Paar, Christian Rechberger, Peter Rombouts, Søren S. Thomsen, and Tolga Yalçin.

PRINCE - A low-latency block cipher for pervasive computing applications - extended abstract. In Xiaoyun Wang and Kazue Sako, editors, *Advances in Cryptology – ASIACRYPT 2012*, volume 7658 of *Lecture Notes in Computer Science*, pages 208–225, Beijing, China, December 2–6, 2012. Springer, Heidelberg, Germany.

[BDJR97] Mihir Bellare, Anand Desai, Eric Jokipii, and Phillip Rogaway. A concrete security treatment of symmetric encryption. In *38th Annual Symposium on Foundations of Computer Science*, pages 394–403, Miami Beach, Florida, October 19–22, 1997. IEEE Computer Society Press.

[BE02] Hagai Bar-El. Security implications of hardware vs. software cryptographic modules. *Discretix White Paper*, 2002.

[Ber13] Daniel Bernstein. Caesar competition, 2013.

[Ber18] Daniel Bernstein. Caesar finalists, 2018.

[BF15] Manuel Barbosa and Pooya Farshim. The related-key analysis of Feistel constructions. In Carlos Cid and Christian Rechberger, editors, *Fast Software Encryption – FSE 2014*, volume 8540 of *Lecture Notes in Computer Science*, pages 265–284, London, UK, March 3–5, 2015. Springer, Heidelberg, Germany.

[BGG94] Mihir Bellare, Oded Goldreich, and Shafi Goldwasser. Incremental cryptography: The case of hashing and signing. In Yvo Desmedt, editor, *Advances in Cryptology – CRYPTO'94*, volume 839 of *Lecture Notes in Computer Science*, pages 216–233, Santa Barbara, CA, USA, August 21–25, 1994. Springer, Heidelberg, Germany.

[BGG95] Mihir Bellare, Oded Goldreich, and Shafi Goldwasser. Incremental cryptography and application to virus protection. In *27th Annual ACM Symposium on Theory of Computing*, pages 45–56, Las Vegas, NV, USA, May 29 – June 1, 1995. ACM Press.

[BGR95a] Mihir Bellare, Roch Guérin, and Phillip Rogaway. XOR MACs: New methods for message authentication using finite pseudorandom functions. In Don Coppersmith, editor, *Advances in Cryptology – CRYPTO'95*, volume 963 of *Lecture Notes in Computer Science*, pages 15–28, Santa Barbara, CA, USA, August 27–31, 1995. Springer, Heidelberg, Germany.

[BGR95b] Mihir Bellare, Roch Guérin, and Phillip Rogaway. Xor macs: New methods for message authentication using finite pseudorandom functions. In *Annual International Cryptology Conference*, pages 15–28. Springer, 1995.

[BHHO08] Dan Boneh, Shai Halevi, Michael Hamburg, and Rafail Ostrovsky. Circular-secure encryption from decision Diffie-Hellman. In David Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 108–125, Santa Barbara, CA, USA, August 17–21, 2008. Springer, Heidelberg, Germany.

[BKL+07] Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, Yannick Seurin, and C. Vikkelsoe.

PRESENT: An ultra-lightweight block cipher. In Pascal Paillier and Ingrid Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems – CHES 2007*, volume 4727 of *Lecture Notes in Computer Science*, pages 450–466, Vienna, Austria, September 10–13, 2007. Springer, Heidelberg, Germany.

[BKY02] Enrico Buonanno, Jonathan Katz, and Moti Yung. Incremental unforgeable encryption. In Mitsuru Matsui, editor, *Fast Software Encryption – FSE 2001*, volume 2355 of *Lecture Notes in Computer Science*, pages 109–124, Yokohama, Japan, April 2–4, 2002. Springer, Heidelberg, Germany.

[BM97] Mihir Bellare and Daniele Micciancio. A new paradigm for collision-free hashing: Incrementality at reduced cost. Cryptology ePrint Archive, Report 1997/001, 1997. http://eprint.iacr.org/1997/001.

[BM06] Brian N. Bershad and Jeffrey C. Mogul, editors. *7th Symposium on Operating Systems Design and Implementation (OSDI '06), November 6-8, Seattle, WA, USA*. USENIX Association, 2006.

[BMM10] Kevin R. B. Butler, Stephen E. McLaughlin, and Patrick D. McDaniel. Disk-enabled authenticated encryption. In *IEEE 26th Symposium on Mass Storage Systems and Technologies, MSST 2012, Lake Tahoe, Nevada, USA, May 3-7, 2010*, pages 1–6, 2010.

[BN00] Mihir Bellare and Chanathip Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In Tatsuaki Okamoto, editor, *Advances in Cryptology – ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 531–545, Kyoto, Japan, December 3–7, 2000. Springer, Heidelberg, Germany.

[BPM18] Milan Broz, Mikulás Patocka, and Vashek Matyás. Practical cryptographic data integrity protection with full disk encryption. In *ICT Systems Security and Privacy Protection - 33rd IFIP TC 11 International Conference, SEC 2018, Held at the 24th IFIP World Computer Congress, WCC 2018, Poznan, Poland, September 18-20, 2018, Proceedings*, pages 79–93, 2018.

[BR00] Mihir Bellare and Phillip Rogaway. Encode-then-encipher encryption: How to exploit nonces or redundancy in plaintexts for efficient cryptography. In Tatsuaki Okamoto, editor, *Advances in Cryptology – ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 317–330, Kyoto, Japan, December 3–7, 2000. Springer, Heidelberg, Germany.

[BR02] John Black and Phillip Rogaway. A block-cipher mode of operation for parallelizable message authentication. In Lars R. Knudsen, editor, *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 384–397, Amsterdam, The Netherlands, April 28 – May 2, 2002. Springer, Heidelberg, Germany.

[BR05] Mihir Bellare and Phillip Rogaway. Introduction to Modern Cryptography. In *UCSD CSE 207 Course Notes*, page 207, 2005.

[BR06] Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 409–426, St. Petersburg, Russia, May 28 – June 1, 2006. Springer, Heidelberg, Germany.

[Bro18] Milan Broz. *Authenticated and Resilient Disk Encryption*. PhD thesis, Masaryk University, 2018.

[BRS03] John Black, Phillip Rogaway, and Thomas Shrimpton. Encryption-scheme security in the presence of key-dependent messages. In Kaisa Nyberg and Howard M. Heys, editors, *SAC 2002: 9th Annual International Workshop on Selected Areas in Cryptography*, volume 2595 of *Lecture Notes in Computer Science*, pages 62–75, St. John's, Newfoundland, Canada, August 15–16, 2003. Springer, Heidelberg, Germany.

[BTCS+15] Ray Beaulieu, Stefan Treatman-Clark, Douglas Shors, Bryan Weeks, Jason Smith, and Louis Wingers. The simon and speck lightweight block ciphers. In *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2015.

[BW99] Alex Biryukov and David Wagner. Slide attacks. In Lars R. Knudsen, editor, *Fast Software Encryption – FSE'99*, volume 1636 of *Lecture Notes in Computer Science*, pages 245–259, Rome, Italy, March 24–26, 1999. Springer, Heidelberg, Germany.

[Cam78] Carl M. Campbell. Design and Specification of Cryptographic Capabilities. *IEEE Communications Society Magazine*, 16(6):15–19, November 1978.

[CB18] Paul Crowley and Eric Biggers. Adiantum: length-preserving encryption for entry-level processors. *IACR Transactions on Symmetric Cryptology*, 2018(4):39–61, 2018.

[CGM13] Omar Choudary, Felix Gröbert, and Joachim Metz. Security analysis and decryption of filevault 2. In *Advances in Digital Forensics IX - 9th IFIP WG 11.9 International Conference on Digital Forensics, Orlando, FL, USA, January 28-30, 2013, Revised Selected Papers*, pages 349–363, 2013.

[CJK+17] Anrin Chakraborti, Bhushan Jain, Jan Kasiak, Tao Zhang, Donald E. Porter, and Radu Sion. dm-x: Protecting volume-level integrity for cloud volumes and local block devices. In *Proceedings of the 8th Asia-Pacific Workshop on Systems, Mumbai, India, September 2, 2017*, pages 16:1–16:7, 2017.

[CK10] Manuel Costa and Engin Kirda, editors. *Proceedings of the Third European Workshop on System Security, EUROSEC 2010, Paris, France, April 13, 2010*. ACM, 2010.

[CLL+14] Shan Chen, Rodolphe Lampe, Jooyoung Lee, Yannick Seurin, and John P. Steinberger. Minimizing the two-round Even-Mansour cipher. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014, Part I*,

volume 8616 of *Lecture Notes in Computer Science*, pages 39–56, Santa Barbara, CA, USA, August 17–21, 2014. Springer, Heidelberg, Germany.

[CMLS15] Debrup Chakraborty, Cuauhtemoc Mancillas-Lopez, and Palash Sarkar. Disk encryption: Do we need to preserve length? Cryptology ePrint Archive, Report 2015/594, 2015. http://eprint.iacr.org/2015/594.

[CS14] Shan Chen and John P. Steinberger. Tight security bounds for key-alternating ciphers. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 327–350, Copenhagen, Denmark, May 11–15, 2014. Springer, Heidelberg, Germany.

[CS15] Benoit Cogliati and Yannick Seurin. On the provable security of the iterated Even-Mansour cipher against related-key and chosen-key attacks. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 584–613, Sofia, Bulgaria, April 26–30, 2015. Springer, Heidelberg, Germany.

[Daw14] Pawel Jakub Dawidek. GELI(8). https://www.freebsd.org/cgi/man.cgi?query=geli, 2014.

[Del09] Dell Computer Corporation, Hewlett Packard Corporation,Hitachi Global Storage Technologies, Inc., Intel Corporation, Maxim Integrated Products, Seagate Technology, Western Digital Corporation . Serial ATA International Organization, Serial ATA Revision 3.0. Technical report, 2009.

[DKS12] Orr Dunkelman, Nathan Keller, and Adi Shamir. Minimalism in cryptography: The Even-Mansour scheme revisited. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 336–354, Cambridge, UK, April 15–19, 2012. Springer, Heidelberg, Germany.

[dm-19a] dm-crypt: Linux device-mapper crypto target. Technical report, 2019.

[dm-19b] dm-verity: Linux device-mapper block integrity checking target. Technical report, 2019.

[DR01] Joan Daemen and Vincent Rijmen. The wide trail design strategy. In Bahram Honary, editor, *8th IMA International Conference on Cryptography and Coding*, volume 2260 of *Lecture Notes in Computer Science*, pages 222–238, Cirencester, UK, December 17–19, 2001. Springer, Heidelberg, Germany.

[DS14] Gareth T. Davies and Martijn Stam. KDM security in the hybrid framework. In Josh Benaloh, editor, *Topics in Cryptology – CT-RSA 2014*, volume 8366 of *Lecture Notes in Computer Science*, pages 461–480, San Francisco, CA, USA, February 25–28, 2014. Springer, Heidelberg, Germany.

[DS16] Yuanxi Dai and John P. Steinberger. Indifferentiability of 8-round Feistel networks. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part I*, volume 9814 of *Lecture Notes in Computer*

Science, pages 95–120, Santa Barbara, CA, USA, August 14–18, 2016. Springer, Heidelberg, Germany.

[DSB+13] Niv Dayan, Martin Kjær Svendsen, Matias Bjørling, Philippe Bonnet, and Luc Bouganim. EagleTree: Exploring the Design Space of SSD-Based Algorithms. *PVLDB*, 6(12):1290–1293, 2013.

[DSST17] Yuanxi Dai, Yannick Seurin, John Steinberger, and Aishwarya Thiruvengadam. Indifferentiability of iterated Even-Mansour ciphers with non-idealized key-schedules: Five rounds are necessary and sufficient. Cryptology ePrint Archive, Report 2017/042, 2017. http://eprint.iacr.org/2017/042.

[Dwo10] Morris Dworkin. Recommendation for Block Cipher Modes of Operation: The XTS-AES Mode for Confidentiality on Storage Devices. NIST SP 800-38E, 2010.

[EM93] Shimon Even and Yishay Mansour. A construction of a cipher from a single pseudorandom permutation. In Hideki Imai, Ronald L. Rivest, and Tsutomu Matsumoto, editors, *Advances in Cryptology – ASIACRYPT'91*, volume 739 of *Lecture Notes in Computer Science*, pages 210–224, Fujiyoshida, Japan, November 11–14, 1993. Springer, Heidelberg, Germany.

[Fer06] Niels Ferguson. AES-CBC + Elephant diffuser: A Disk Encryption Algorithm for Windows Vista. http://www.microsoft.com/en-us/download/details.aspx?id=13866, 2006.

[Fis97a] Marc Fischlin. Incremental cryptography and memory checkers. In Walter Fumy, editor, *Advances in Cryptology – EUROCRYPT'97*, volume 1233 of *Lecture Notes in Computer Science*, pages 293–408, Konstanz, Germany, May 11–15, 1997. Springer, Heidelberg, Germany.

[Fis97b] Marc Fischlin. Lower bounds for the signature size of incremental schemes. In *38th Annual Symposium on Foundations of Computer Science*, pages 438–447, Miami Beach, Florida, October 19–22, 1997. IEEE Computer Society Press.

[FKV17] Pooya Farshim, Louiza Khati, and Damien Vergnaud. Security of even–mansour ciphers under key-dependent messages. *IACR Transactions on Symmetric Cryptology*, 2017(2):84–104, 2017.

[FP15] Pooya Farshim and Gordon Procter. The related-key security of iterated Even-Mansour ciphers. In Gregor Leander, editor, *Fast Software Encryption – FSE 2015*, volume 9054 of *Lecture Notes in Computer Science*, pages 342–363, Istanbul, Turkey, March 8–11, 2015. Springer, Heidelberg, Germany.

[FPR12] Aaron Fujimoto, Peter Peterson, and Peter L. Reiher. Comparing the power of full disk encryption alternatives. In *IGCC*, pages 1–6. IEEE Computer Society, 2012.

[Fru05] Clemens Fruhwirth. New methods in hard disk encryption. Master's thesis, Vienna University of Technology, 2005.

[fS09] International Organization for Standardization. Information technology – Security techniques – Authenticated encryption. Standard, Geneva, CH, 2009.

[fS11a] International Organization for Standardization. Information technology – Security techniques – Message Authentication Codes (MACs) – Part 1: Mechanisms using a block cipher. Standard, Geneva, CH, 2011.

[fS11b] International Organization for Standardization. Information technology – Security techniques – Message Authentication Codes (MACs) – Part 2: Mechanisms using a dedicated hash-function. Standard, Geneva, CH, 2011.

[Gjø05a] Kristian Gjøsteen. Security notions for disk encryption. In *Computer Security - ESORICS 2005, 10th European Symposium on Research in Computer Security, Milan, Italy, September 12-14, 2005, Proceedings*, pages 455–474, 2005.

[Gjø05b] Kristian Gjøsteen. Security notions for disk encryption. In Sabrina De Capitani di Vimercati, Paul F. Syverson, and Dieter Gollmann, editors, *ESORICS 2005: 10th European Symposium on Research in Computer Security*, volume 3679 of *Lecture Notes in Computer Science*, pages 455–474, Milan, Italy, September 12–14, 2005. Springer, Heidelberg, Germany.

[GL15] Chun Guo and Dongdai Lin. On the indifferentiability of key-alternating Feistel ciphers with no key derivation. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015: 12th Theory of Cryptography Conference, Part I*, volume 9014 of *Lecture Notes in Computer Science*, pages 110–133, Warsaw, Poland, March 23–25, 2015. Springer, Heidelberg, Germany.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[GM14] Johannes Götzfried and Tilo Müller. Analysing android's full disk encryption feature. *JoWUA*, 5(1):84–100, 2014.

[GR12] John Gantz and David Reinsel. The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east. *IDC iView: IDC Analyze the future*, 2007(2012):1–16, 2012.

[GSC01] Bok-Min Goi, M. U. Siddiqi, and Hean-Teik Chuah. Incremental hash function based on pair chaining & modular arithmetic combining. In C. Pandu Rangan and Cunsheng Ding, editors, *Progress in Cryptology - INDOCRYPT 2001: 2nd International Conference in Cryptology in India*, volume 2247 of *Lecture Notes in Computer Science*, pages 50–61, Chennai, India, December 16–20, 2001. Springer, Heidelberg, Germany.

[GW18] Chun Guo and Lei Wang. Revisiting key-alternating feistel ciphers for shorter keys and multi-user security. In Thomas Peyrin and Steven Galbraith, editors, *Advances in Cryptology – ASIACRYPT 2018, Part I*, Lecture Notes in Computer Science, pages 213–243, Brisbane,Queensland, Australia, December 2–6, 2018. Springer, Heidelberg, Germany.

[Hal06] Shai Halevi. Re: Lrw key derivation (formerly pink-herring). IEEE P1619 Mailing List, May 2006.

[HGX+09] Fangyong Hou, Dawu Gu, Nong Xiao, Fang Liu, and Hongjun He. Performance and consistency improvements of hash tree based disk storage protection. In *International Conference on Networking, Architecture, and Storage, NAS 2009, 9-11 July 2009, Zhang Jia Jie, Hunan, China*, pages 51–56, 2009.

[HK07] Shai Halevi and Hugo Krawczyk. Security under key-dependent inputs. In Peng Ning, Sabrina De Capitani di Vimercati, and Paul F. Syverson, editors, *ACM CCS 07: 14th Conference on Computer and Communications Security*, pages 466–475, Alexandria, Virginia, USA, October 28–31, 2007. ACM Press.

[HM06] J Hart and Kirk Martinez. Environmental sensor networks: A revolution in the earth system science? *Earth-Science Reviews*, 78:177–191, 10 2006.

[HPPT08] Alexander Heitzmann, Bernardo Palazzi, Charalampos Papamanthou, and Roberto Tamassia. Efficient integrity checking of untrusted network storage. In *Proceedings of the 2008 ACM Workshop On Storage Security And Survivability, StorageSS 2008, Alexandria, VA, USA, October 31, 2008*, pages 43–54, 2008.

[HR03] Shai Halevi and Phillip Rogaway. A tweakable enciphering mode. In Dan Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 482–499, Santa Barbara, CA, USA, August 17–21, 2003. Springer, Heidelberg, Germany.

[HR04] Shai Halevi and Phillip Rogaway. A parallelizable enciphering mode. In Tatsuaki Okamoto, editor, *Topics in Cryptology – CT-RSA 2004*, volume 2964 of *Lecture Notes in Computer Science*, pages 292–304, San Francisco, CA, USA, February 23–27, 2004. Springer, Heidelberg, Germany.

[HR10] Viet Tung Hoang and Phillip Rogaway. On generalized Feistel networks. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 613–630, Santa Barbara, CA, USA, August 15–19, 2010. Springer, Heidelberg, Germany.

[HSTM15] Michael Halcrow, Uday Savagaonkar, Ted Ts'o, and Ildar Muslukhov. EXT4 Encryption Design Document (public version). Google Technical Report, 2015.

[HT16] Viet Tung Hoang and Stefano Tessaro. Key-alternating ciphers and key-length extension: Exact bounds and multi-user security. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part I*, volume 9814 of *Lecture Notes in Computer Science*, pages 3–32, Santa Barbara, CA, USA, August 14–18, 2016. Springer, Heidelberg, Germany.

[HU08] Dennis Hofheinz and Dominique Unruh. Towards key-dependent message security in the standard model. In Nigel P. Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 108–126, Istanbul, Turkey, April 13–17, 2008. Springer, Heidelberg, Germany.

[HWF15] Daniel Hein, Johannes Winter, and Andreas Fitzek. Secure Block Device–Secure, Flexible, and Efficient Data Storage for ARM TrustZone Systems. In *Trustcom/BigDataSE/ISPA, 2015 IEEE*, volume 1, pages 222–229. IEEE, 2015.

[IEE08] IEEE. IEEE Standard for Cryptographic Protection of Data on Block-Oriented Storage Devices. *IEEE Std 1619-2007*, pages 1–32, 2008.

[IEE11] IEEE. IEEE Standard for for Wide-Block Encryption for Shared Storage Media. *IEEE Std P1619.11619-2010*, 2011.

[IEE18] IEEE. IEEE Standard for Authenticated Encryption with Length Expansion for Storage Devices. *IEEE Std 1619.1-2018*, pages 1–45, 2018.

[IGFH18] Bernard Dickens III, Haryadi S. Gunawi, Ariel J. Feldman, and Henry Hoffmann. Strongbox: Confidentiality, integrity, and performance using stream ciphers for full drive encryption. In *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS 2018, Williamsburg, VA, USA, March 24-28, 2018*, pages 708–721, 2018.

[IK01] Tetsu Iwata and Kaoru Kurosawa. On the pseudorandomness of the AES finalists - RC6 and Serpent. In Bruce Schneier, editor, *Fast Software Encryption – FSE 2000*, volume 1978 of *Lecture Notes in Computer Science*, pages 231–243, New York, NY, USA, April 10–12, 2001. Springer, Heidelberg, Germany.

[IKC10] Wassim Itani, Ayman Kayssi, and Ali Chehab. Energy-efficient incremental integrity for securing storage in mobile cloud computing. In *2010 International Conference on Energy Aware Computing*, pages 1–2. IEEE, 2010.

[IS09] Kevin Igoe and Jerome Solinas. Aes galois counter mode for the secure shell transport layer protocol. Technical report, 2009.

[JMV02] Antoine Joux, Gwenaëlle Martinet, and Frédéric Valette. Blockwise-adaptive attackers: Revisiting the (in)security of some provably secure encryption models: CBC, GEM, IACBC. In Moti Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 17–30, Santa Barbara, CA, USA, August 18–22, 2002. Springer, Heidelberg, Germany.

[Jos11] Simon Josefsson. Pkcs# 5: password-based key derivation function 2 (pbkdf2) test vectors. Technical report, Internet Engineering Task Force, 2011.

[Jut00] Charanjit Jutla. Attack on Free-MAC, 2000. https://groups.google.com/d/msg/sci.crypt/4bkzm_n7UGA/5cDwfju6evUJ.

[Jut01] Charanjit S. Jutla. Encryption modes with almost free message integrity. In Birgit Pfitzmann, editor, *Advances in Cryptology – EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 529–544, Innsbruck, Austria, May 6–10, 2001. Springer, Heidelberg, Germany.

[KKJ+13] Himanshu Kumar, Sudhanshu Kumar, Remya Joseph, Dhananjay Kumar, Sunil Kumar Shrinarayan Singh, and Praveen Kumar. Rainbow table to crack password using md5 hashing algorithm. In *Information & Communication Technologies (ICT), 2013 IEEE Conference on*, pages 433–439. IEEE, 2013.

[Kle09] Peter Kleissner. Stoned bootkit. *Black Hat*, 2009.

[KMV17] Louiza Khati, Nicky Mouha, and Damien Vergnaud. Full disk encryption: Bridging theory and practice. In Helena Handschuh, editor, *Topics in Cryptology – CT-RSA 2017*, volume 10159 of *Lecture Notes in Computer Science*, pages 241–257, San Francisco, CA, USA, February 14–17, 2017. Springer, Heidelberg, Germany.

[KV18] Louiza Khati and Damien Vergnaud. Analysis and Improvement of an Authentication Scheme in Incremental Cryptography. In *Selected Areas in Cryptography - SAC 2018*, Calgary, Canada, August 2018.

[KVW04] Tadayoshi Kohno, John Viega, and Doug Whiting. CWC: A high-performance conventional authenticated encryption mode. In Bimal K. Roy and Willi Meier, editors, *Fast Software Encryption – FSE 2004*, volume 3017 of *Lecture Notes in Computer Science*, pages 408–426, New Delhi, India, February 5–7, 2004. Springer, Heidelberg, Germany.

[LB17] Ronan Loftus and Marwin Baumann. Android 7 file based encryption and the attacks against it. 2017.

[Lel13] Jakob Lell. Practical malleability attack against cbc-encrypted luks partitions. *Blog*, 2013.

[LLJ15] Xianhui Lu, Bao Li, and Dingding Jia. KDM-CCA security from RKA secure authenticated encryption. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 559–583, Sofia, Bulgaria, April 26–30, 2015. Springer, Heidelberg, Germany.

[LR86] Michael Luby and Charles Rackoff. How to construct pseudo-random permutations from pseudo-random functions (abstract). In Hugh C. Williams, editor, *Advances in Cryptology – CRYPTO'85*, volume 218 of *Lecture Notes in Computer Science*, page 447, Santa Barbara, CA, USA, August 18–22, 1986. Springer, Heidelberg, Germany.

[LS15] Rodolphe Lampe and Yannick Seurin. Security analysis of key-alternating Feistel ciphers. In Carlos Cid and Christian Rechberger, editors, *Fast Software Encryption – FSE 2014*, volume 8540 of *Lecture Notes in Computer Science*, pages 243–264, London, UK, March 3–5, 2015. Springer, Heidelberg, Germany.

[Mat97] Mitsuru Matsui. New block encryption algorithm MISTY. In Eli Biham, editor, *Fast Software Encryption – FSE'97*, volume 1267 of *Lecture Notes in Computer Science*, pages 54–68, Haifa, Israel, January 20–22, 1997. Springer, Heidelberg, Germany.

[Mau93] Ueli M. Maurer. A simplified and generalized treatment of Luby-Rackoff pseudorandom permutation generator. In Rainer A. Rueppel, editor, *Advances in Cryptology – EUROCRYPT'92*, volume 658 of *Lecture Notes in Computer Science*, pages 239–255, Balatonfüred, Hungary, May 24–28, 1993. Springer, Heidelberg, Germany.

[MF07] David A. McGrew and Scott R. Fluhrer. The security of the extended codebook (xcb) mode of operation. In Carlisle M. Adams, Ali Miri, and Michael J. Wiener, editors, *SAC 2007: 14th Annual International Workshop on Selected Areas in Cryptography*, volume 4876 of *Lecture Notes in Computer Science*, pages 311–327, Ottawa, Canada, August 16–17, 2007. Springer, Heidelberg, Germany.

[MF15] Tilo Müller and Felix C. Freiling. A Systematic Assessment of the Security of Full Disk Encryption. *IEEE Trans. Dependable Sec. Comput.*, 12(5):491–503, 2015.

[MGS15] Hristina Mihajloska, Danilo Gligoroski, and Simona Samardjiska. Reviving the idea of incremental cryptography for the zettabyte era use case: Incremental hash functions based on SHA-3. Cryptology ePrint Archive, Report 2015/1028, 2015. http://eprint.iacr.org/2015/1028.

[Mic97] Daniele Micciancio. Oblivious data structures: Applications to cryptography. In *29th Annual ACM Symposium on Theory of Computing*, pages 456–464, El Paso, TX, USA, May 4–6, 1997. ACM Press.

[ML15] Nicky Mouha and Atul Luykx. Multi-key security: The Even-Mansour construction revisited. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *Advances in Cryptology – CRYPTO 2015, Part I*, volume 9215 of *Lecture Notes in Computer Science*, pages 209–223, Santa Barbara, CA, USA, August 16–20, 2015. Springer, Heidelberg, Germany.

[MLF12] Tilo Müller, Tobias Latzo, and Felix C Freiling. Self-encrypting disks pose self-decrypting risks. In *the 29th Chaos Communinication Congress*, pages 1–10, 2012.

[MP03] Ueli M. Maurer and Krzysztof Pietrzak. The security of many-round Luby-Rackoff pseudo-random permutations. In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 544–561, Warsaw, Poland, May 4–8, 2003. Springer, Heidelberg, Germany.

[MPRS12] Ilya Mironov, Omkant Pandey, Omer Reingold, and Gil Segev. Incremental deterministic public-key encryption. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 628–644, Cambridge, UK, April 15–19, 2012. Springer, Heidelberg, Germany.

[MTF12] Tilo Müller, Benjamin Taubmann, and Felix C. Freiling. TreVisor - OS-independent software-based full disk encryption secure against main memory attacks. In Feng Bao, Pierangela Samarati, and Jianying Zhou, editors, *ACNS 12: 10th International Conference on Applied Cryptography and Network Security*, volume 7341 of *Lecture Notes in Computer Science*, pages 66–83, Singapore, June 26–29, 2012. Springer, Heidelberg, Germany.

[MTY11] Tal Malkin, Isamu Teranishi, and Moti Yung. Efficient circuit-size independent public key encryption with KDM security. In Kenneth G. Paterson, editor, *Advances in Cryptology – EUROCRYPT 2011*, volume 6632 of *Lecture Notes in*

Computer Science, pages 507–526, Tallinn, Estonia, May 15–19, 2011. Springer, Heidelberg, Germany.

[MvG18] Carlo Meijer and Bernard van Gastel. Self-encrypting deception: weaknesses in the encryption of solid state drives (ssds). 2018.

[Nan08] Mridul Nandi. Two new efficient CCA-secure online ciphers: MHCBC and MCBC. In Dipanwita Roy Chowdhury, Vincent Rijmen, and Abhijit Das, editors, *Progress in Cryptology - INDOCRYPT 2008: 9th International Conference in Cryptology in India*, volume 5365 of *Lecture Notes in Computer Science*, pages 350–362, Kharagpur, India, December 14–17, 2008. Springer, Heidelberg, Germany.

[NRS14] Chanathip Namprempre, Phillip Rogaway, and Thomas Shrimpton. Reconsidering generic composition. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 257–274, Copenhagen, Denmark, May 11–15, 2014. Springer, Heidelberg, Germany.

[OR05] Alina Oprea and Michael K. Reiter. Space-efficient block storage integrity. In *Proceedings of the Network and Distributed System Security Symposium, NDSS 2005, San Diego, California, USA*, 2005.

[OR07] Alina Oprea and Michael K. Reiter. Integrity checking in cryptographic file systems with constant trusted storage. In *Proceedings of the 16th USENIX Security Symposium, Boston, MA, USA, August 6-10, 2007*, 2007.

[Pat90] Jacques Patarin. Pseudorandom permutations based on the D.E.S. scheme. In *ESORICS'90: 1st European Symposium on Research in Computer Security*, Lecture Notes in Computer Science, pages 185–187, Toulouse, France, October 24–26, 1990. AFCET.

[Pat98] Jacques Patarin. About Feistel schemes with six (or more) rounds. In Serge Vaudenay, editor, *Fast Software Encryption – FSE'98*, volume 1372 of *Lecture Notes in Computer Science*, pages 103–121, Paris, France, March 23–25, 1998. Springer, Heidelberg, Germany.

[Pat04] Jacques Patarin. Security of random Feistel schemes with 5 or more rounds. In Matthew Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 106–122, Santa Barbara, CA, USA, August 15–19, 2004. Springer, Heidelberg, Germany.

[Pat09] Jacques Patarin. The "coefficients H" technique (invited talk). In Roberto Maria Avanzi, Liam Keliher, and Francesco Sica, editors, *SAC 2008: 15th Annual International Workshop on Selected Areas in Cryptography*, volume 5381 of *Lecture Notes in Computer Science*, pages 328–345, Sackville, New Brunswick, Canada, August 14–15, 2009. Springer, Heidelberg, Germany.

[Pau17] Paul A. Grassi, James L. Fenton, Elaine M. Newton, Ray A. Perlner,Andrew R. Regenscheid, William E. Burr,Justin P. Richer. Digital Identity Guidelines, Authentication and Lifecycle Management. NIST Special Publication 800-63B, 2017.

[PJ16] Colin Percival and Simon Josefsson. The scrypt password-based key derivation function. Technical report, 2016.

[Pri19] Primx. CRYHOD. Technical report, 2019.

[PW06] Raphael C-W Phan and David Wagner. Security considerations for incremental hash functions based on pair block chaining. *computers & security*, 25(2):131–136, 2006.

[RCPS07] Brian Rogers, Siddhartha Chhabra, Milos Prvulovic, and Yan Solihin. Using address independent seed encryption and bonsai merkle trees to make secure processors os-and performance-friendly. In *Proceedings of the 40th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 183–196. IEEE Computer Society, 2007.

[Res08] Eric Rescorla. Tls elliptic curve cipher suites with sha-256/384 and aes galois counter mode (gcm). 2008.

[RFC+07] Richard F Rizzolo, Thomas G Foote, James M Crafts, David A Grosch, Tak O Leung, David J Lund, Bryan L Mechtly, Bryan J Robbins, Timothy J Slegel, Michael J Tremblay, et al. Ibm system z9 efuse applications and methodology. *IBM Journal of Research and Development*, 51(1.2):65–75, 2007.

[Rog04a] Phillip Rogaway. Efficient instantiations of tweakable blockciphers and refinements to modes OCB and PMAC. In Pil Joong Lee, editor, *Advances in Cryptology – ASIACRYPT 2004*, volume 3329 of *Lecture Notes in Computer Science*, pages 16–31, Jeju Island, Korea, December 5–9, 2004. Springer, Heidelberg, Germany.

[Rog04b] Phillip Rogaway. Nonce-based symmetric encryption. In Bimal K. Roy and Willi Meier, editors, *Fast Software Encryption – FSE 2004*, volume 3017 of *Lecture Notes in Computer Science*, pages 348–359, New Delhi, India, February 5–7, 2004. Springer, Heidelberg, Germany.

[Rog11] Phillip Rogaway. Evaluation of Some Blockcipher Modes of Operation. Technical report, CRYPTREC Investigation Report, 2011.

[Rut09] Joanna Rutkowska. Evil maid goes after truecrypt. *Blog*, 2009.

[RZ11] Phillip Rogaway and Haibin Zhang. Online ciphers from tweakable blockciphers. In Aggelos Kiayias, editor, *Topics in Cryptology – CT-RSA 2011*, volume 6558 of *Lecture Notes in Computer Science*, pages 237–249, San Francisco, CA, USA, February 14–18, 2011. Springer, Heidelberg, Germany.

[Saa04] Markku-Juhani Olavi Saarinen. Encrypted watermarks and linux laptop security. In Chae Hoon Lim and Moti Yung, editors, *WISA 04: 5th International Workshop on Information Security Applications*, volume 3325 of *Lecture Notes in Computer Science*, pages 27–38, Jeju Island, Korea, August 23–25, 2004. Springer, Heidelberg, Germany.

[Sim11] Patrick Simmons. Security through amnesia: a software-based solution to the cold boot attack on disk encryption. In *Twenty-Seventh Annual Computer Security Applications Conference, ACSAC 2011, Orlando, FL, USA, 5-9 December 2011*, pages 73–82, 2011.

[SSTC02] Dafna Sheinwald, Julian Satran, Pat Thaler, and Vicente Cavanna. Internet protocol small computer system interface (iscsi) cyclic redundancy check (crc)/checksum considerations. *RFC*, 3385:1–23, 2002.

[STM17] STMicroelectronics. High-speed secure MCU with 32-bit ARM® SecurCore® SC300TM CPU with SWP, ISO, SPI, I2C and high-density Flash memory. Technical report, February 2017.

[SWZ05] Gopalan Sivathanu, Charles P. Wright, and Erez Zadok. Ensuring data integrity in storage: techniques and applications. In *Proceedings of the 2005 ACM Workshop On Storage Security And Survivability, StorageSS 2005, Fairfax, VA, USA, November 11, 2005*, pages 26–36, 2005.

[SY16] Yu Sasaki and Kan Yasuda. A new mode of operation for incremental authenticated encryption with associated data. In Orr Dunkelman and Liam Keliher, editors, *SAC 2015: 22nd Annual International Workshop on Selected Areas in Cryptography*, volume 9566 of *Lecture Notes in Computer Science*, pages 397–416, Sackville, NB, Canada, August 12–14, 2016. Springer, Heidelberg, Germany.

[Tec] IBM Linux Technology. ecryptfs.

[Ter10] Alexander Tereshkin. Evil maid goes after PGP whole disk encryption. In *Proceedings of the 3rd International Conference on Security of Informat ion and Networks, SIN 2010, Rostov-on-Don, Russian Federation, September 7-11, 2010*, page 2, 2010.

[UM16] Thomas Unterluggauer and Stefan Mangard. Exploiting the physical disparity: Side-channel attacks on memory encryption. In *Constructive Side-Channel Analysis and Secure Design - 7th International Workshop, COSADE 2016, Graz, Austria, April 14-15, 2016, Revised Selected Papers*, pages 3–18, 2016.

[vDRSD07] Marten van Dijk, Jonathan Rhodes, Luis F. G. Sarmenta, and Srinivas Devadas. Offline untrusted storage with immediate detection of forking and replay attacks. In *Proceedings of the 2nd ACM Workshop on Scalable Trusted Computing, STC 2007, Alexandria, VA, USA, November 2, 2007*, pages 41–48, 2007.

[ver19] dm-crypt: Linux device-mapper crypto target. Technical report, 2019.

[VG18] Andrea Visconti and Federico Gorla. Exploiting an hmac-sha-1 optimization to speed up pbkdf2. 2018.

[VM05] John Viega and D McGrew. The use of galois/counter mode (gcm) in ipsec encapsulating security payload (esp). Technical report, 2005.

[VV17] Serge Vaudenay and Damian Vizár. Under pressure: Security of caesar candidates beyond their guarantees. Cryptology ePrint Archive, Report 2017/1147, 2017. https://eprint.iacr.org/2017/1147.

[Yas11] Kan Yasuda. A new variant of PMAC: Beyond the birthday bound. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 596–609, Santa Barbara, CA, USA, August 14–18, 2011. Springer, Heidelberg, Germany.

List of Illustrations

Figures

1.1 (Right) Game for defining the prf-advantage of pseudorandom family functions F. (Left) Game for defining the prp-advantage of pseudorandom permutation family P. 6

2.1 Simplified view of components involved in disk encryption. 15
2.2 Simplified logical levels: from files to sectors. MD stands for Meta-data. . . . 16
2.3 HDD tracks and view of a sector. 16
2.4 Hierarchically organized SSD memory 17
2.5 Disk where memory are on platters for HDD and are flash cells for SSD. . . . 18
2.6 Malleability attack on CBC-ESSIV using a block cipher (E_K,D_k) where $D_k = E_k^-$. Bit flipping ("X") in the i-th ciphertext block compromises the entire i-th block and exactly one bit in the i+1-th block. 22
2.7 Malleability attack on XTS (here without ciphertext stealing) using a block cipher (E_K,D_k) where $D_k = E_k^-$. Bit flipping ("X") in the i-th ciphertext block compromises the i-th plaintext block (black block) after decryption. 23
2.8 Malleability attack on WBTC using a block cipher (E_K,D_k) where $D_k = E_k^-$. Bit flipping ("X") in the i-th ciphertext block compromises all the plaintext (black blockq) after decryption. 23
2.9 Disk Encryption and local authentication tag within each sector. Dashed boxes represent encrypted data and hatched boxes tags. AE stands for Authenticated Encryption. 26
2.10 Interleaved Meta-data. Large dashed boxes represent encrypted data and small dashed boxes meta-data. 27
2.11 Overview of incremental replace operation. The function f_K denotes the MAC primitive. The inputs τ_1, τ_2, τ_3 give the tag τ. After replacing the value τ_1 by τ_4, the tag τ is updated with τ'. 28
2.12 Perfect Merkle tree of n' levels. Nodes of level m only are stored $(m \leq n')$. . . 29

3.1 Description of the mode of operation ECB (*Electronic Codebook*) where E is a block cipher. 39
3.2 Description of the counter mode (CTR) where E is a block cipher and i is the counter. 39
3.3 Description of the CBC-ESSIV and IGE-ESSIV modes of operation where E is a block cipher and the keys K and K' are independent. 40
3.4 Description of the XEX mode of operation where E is a block cipher and K and K' are independant key. 40
3.5 Description of the TC1 and TC2 modes of operations where $\tilde{\mathsf{E}}$ is a tweakable block cipher. In FDE context, the tweak can be the sector number s (for TC2, $s_0||s_1 = \mathsf{P}(s)$ where P is a permutation. 41

3.6 Description of the TC3 mode of operation where $\tilde{\mathsf{E}}$ is a tweakable block cipher. The tweak can be the sector number s. 41
3.7 Description of the HCBC1 mode of operation where E_K is a block cipher, $h_{K'}$ a keyed hash function. The keys K and K' are independent. 42
3.8 WTBC where E_K is either a block cipher or a tweakable block cipher. WTBC can use several call to E_K to process one message block. 42
3.9 Game "up-to-block" for the IND-CCA-block security notion. 43
3.10 Game "up-to-repetition" for the IND-CPA-repetition security notion. 44
3.11 Game "up-to-prefix" for the IND-CCA-prefix security notion. 47
3.12 Game $\mathcal{G}_1$ and game $\mathcal{G}_2$ for CBC-ESSIV IND-CPA-ufb 50
3.13 Games $\mathcal{G}_3$, $\mathcal{G}_4$ and $\mathcal{G}_5$ CBC-ESSIV IND-CPA-ufb. The framed statement is included in game $\mathcal{G}_3$ only. The statement overlined in gray is included in game $\mathcal{G}_5$ only. 51
3.14 Games $\mathcal{G}_1$ to $\mathcal{G}_4$ for IGE-ESSIV IND-CPA-ufb 53

4.1 The Even–Mansour cipher (1-round). 63
4.2 Descriptions of the round function of Feistel Network (Left) and the one round KAF (Right). 64
4.3 KDM-CCA analysis for 2-round Even–Mansour. 66
4.4 Game defining Φ-KDM-CCA security for a block-cipher. 67
4.5 Game defining oracle splittability with respect to relation **sp**. 69
4.6 Game defining forgetful switching property. 69
4.7 Two-stage adversaries and their operation in the split game. 71
4.8 Algorithm $\mathcal{B}$ used for KDM analysis of the ideal cipher. 74
4.9 The r-round iterated Even–Mansour cipher. 76
4.10 Adversaries $\mathcal{A}^{\mathsf{P},\mathsf{EM}^{\mathsf{P},\mathsf{P}}[K,K,K]}$ and $\mathcal{A}^{\mathsf{P},\mathsf{EM}^{\mathsf{P},\mathsf{P}}[K_1,K_2,K_2]}$. 77
4.11 Algorithm $\mathcal{B}$ used in the KDM analysis of one-round EM. 79
4.12 Algorithm $\mathcal{B}$ used in the KDM analysis of two-round EM with two permutations. 80
4.13 Algorithm $\mathcal{B}$ used in the KDM analysis of two-round EM with a single permutation. 82
4.14 Real world rw and ideal world iw defining KDM-CCA-security and perfect world pw used in Theorem 4.3.5. 86
4.15 The r-round Key-Alternating Feistel (KAF). 89
4.16 (left) In red, collisions giving a bad transcript (Condition C-i). (Right) In red, for a good transcript τ, F is bad if the inputs w_i and z_i collide with any other input of F (conditions C'-i). For a good transcript τ, F is good if the values w_i,z_i are all distinct; 91
4.17 Details of the Sliding attack on 4-rounds KAF (same keys and same round function). 101
4.18 Notations for Sliding attack. 101

5.1 Examples of edit operations. 111
5.2 Games defining Correctness and Consistency for nonce-based incremental authentication scheme iMA for the single-document setting. 113
5.3 Games defining Correctness and Consistency for nonce-based incremental authentication scheme iMA for the multi-document setting. 114
5.4 Games defining IUF-BS security of an incremental authentication scheme iMA. 116

5.5 Games defining IUF1 security of an incremental authentication scheme iMA for both the Single-Document (SD) and the Multi-Document (MD) settings. The framed boxes are excluded from the SD security game. 117
5.6 Games defining IUF1R security of an incremental authentication scheme iMA for both the Single-Document (SD) and the Multi-Document (MD) settings. The framed boxes are excluded from the SD security game. 118
5.7 Game defining IUF2 security of an incremental authentication scheme iMA in the Single-Document (SD) and in the Multi-Document (MD) settings. The framed boxes are excluded from the SD game. 119
5.8 Game defining IUF2R security of an incremental authentication scheme iMA in the Single-Document (SD) and in the Multi-Document (MD) settings. The framed boxes are excluded from the SD game. 120
5.9 Relations among security notions for incremental MACs.
A black arrow is an implication, and in the directed graph given by the black arrows, there is a path from security notion A to security notion B if and only A implies B. The red arrows represent separations (i.e. if there is a red arrow from security notion A to security notion B, then – assuming the existence of one-way functions – there exist a scheme which achieves A but does not achieve B). All other relations follow automatically. 122
5.10 Transform $\mathbf{SDtoMD_1}$ from a single-document scheme to a multi-document scheme, denoted as $\mathsf{iMA2} = \mathbf{SDtoMD_1}(\mathsf{iMA1}, \mathsf{F})$. 123
5.11 Adversary $\mathcal{D}$ against the PRF game using the adversary $\mathcal{A}$ playing the indistinguishability game between G_0 and G_1 (see Figure 5.12) used in in the proof of Theorem 5.1.7. 124
5.12 Games for proof of security of the $\mathbf{SDtoMD_1}$ transform (Theorem 5.1.7). . 125
5.13 Construction of an adversary $\mathcal{B}$ against the IUFxy-MD security game using $\mathcal{A}$ playing the indistinguishability G_1 and G_2 in the proof of Theorem 5.1.7. . . 126
5.14 Description of the XMAC where v is a random value for XMACR and v is a nonce for XMACC. 127
5.15 Description of the Xor-Scheme 129
5.16 XS Original Xor-Scheme Algorithm 131
5.17 Xor cancellation strategy in the Xor-Scheme 132
5.18 Σ computation for 3 block document 132
5.19 Attack on the Xor-Scheme. Here $R_i = D[i]||r[i]$. 133
5.20 Description of the fixed Xor-Scheme 134
5.21 Complexity of the Xor-Scheme XS and the Modified Xor-Scheme MXS. 135
5.22 Modified Xor-Scheme Algorithm. 141

Tables

2.1 Operation and Storage costs of incremental MACS for n-block input. The column L is for *Locality property*. 29
2.2 Estimation of Stand-alone storage of Merkle tree with n actual data sectors where $\mathsf{n} = 2^{n'}$. Up/Ver stands for update and verify cost. 30
2.3 Estimation of Stand-alone storage of Merkle tree where $\mathsf{n} = 2^{28}$ and $L = 32$ Bytes 31

2.4 Estimation of Hybrid storage of Merkle Tree with n actual data sectors where $\mathsf{n} = 2^{n'}$ and $m \leq n'$. Up/Ver stands for update and verify cost. 32
2.5 Evaluation of the Hybrid storage of Merkle tree with $n = 2^{28}$, $m = 15$, $L = 32$ Bytes. 33
2.6 Evaluation of the Hybrid storage of Merkle tree with $n = 2^{28}$, $m = 25$, $L = 32$ Bytes. 33
2.7 Some implementations giving different levels of security for a stand-alone disk. 34

3.1 The security of FDE modes of operation when no diversifier is used. Here, ✓ means that there is a security proof, and ✗ means that there is an attack. Proofs of the security results can be found in Sect. 3.1.2. XTS: see [IEE08]. TC1, TC2 and TC3 [RZ11] are generalizations of the HCBC1 [BBKN12], HCBC2 [BBKN12] and MHCBC [Nan08] constructions. WTBC: wide tweakable block cipher. The * symbol indicates that the property holds for some constructions, but not for others. Here, $x \geq \log_2(\ell)$. 45
3.2 The security of FDE modes of operation when no diversifier is used, but the first plaintext block unique for any given sector. Here, ✓ means that there is a security proof, and ✗ means that there is an attack. 49
3.3 The security of FDE modes of operation when a diversifier is used. Here, ✓ means that there is a security proof, and ✗ means that there is an attack. . . 55
3.4 EagleTree Benchmarks for various diversifier sizes. 58

Personal Publications

- Full Disk Encryption: Bridging Theory and Practice
  Louiza Khati, Nicky Mouha and Damien Vergnaud.
  CT-RSA 2017

- Security of Even–Mansour Ciphers under Key Dependent Message Security
  Pooya Farshim, Louiza Khati and Damien Vergnaud.
  IACR Transactions on Symmetric Cryptology, Volume 2017.

- Analysis and Improvement of an Authentication Scheme in Incremental Cryptography
  Louiza Khati and Damien Vergnaud.
  Selected Areas in Cryptography - SAC 2018.

RÉSUMÉ

Cette thèse est dédiée à l'analyse de modes opératoires pour la protection des disques durs. Dans un premier temps, l'analyse des modes opératoires permettant de protéger la confidentialité des données est réalisée dans le modèle *Full Disk Encryption*. Ce modèle est très contraignant puisqu'il exclu tout mode qui ne ne conserve pas la longueur (la valeur en clair et chiffrée du secteur doivent avoir la même taille) et seuls des modes déterministes peuvent avoir cette propriété. Néanmoins, il est possible de tirer partie d'une valeur du système nommée le *diversifiant*, qui originellement a un autre but, pour apporter de l'aléa utile pour améliorer la sécurité des modes opératoires. Dans un second temps, nous introduisons deux méthodologies d'analyse dans le modèle *Key-Dependent Message*, où l'adversaire est autorisé à chiffrer des messages qui dépendent de la clé de chiffrement, qui nous ont permis d'analyser la sécurité des schémas Even-Mansour et Key-Alternating Feistel. Enfin, sachant qu'il est impossible de garantir l'authenticité des données dans le modèle FDE, la présence de codes d'authentification étant nécéssaire, deux modèles où le stockage de métadonnées est possible sont envisagés: le modèle ADE pour *Authenticated Disk Encryption* et le modèle FADE pour *Fully Authenticated Disk Encryption*. Le premier permet de garantir l'authenticité au niveau du secteur mais est vulnérable aux *attaques par rejeu* et le second garantit l'authenticité du disque en entier et prévient ce type d'attaque. Le stockage n'est pas le seul point à prendre en compte: les vitesses de lecture et d'écriture sont un enjeu de taille pour les constructeurs puisque ces dernières conditionnent fortement les performances d'un disque. C'est la raison pour laquelle, nous avons étudié les codes d'authentification incrémentaux puisque ces derniers ont la propriété d'être mis à jour en un temps proportionnel à la modification réalisée.

MOTS CLÉS

Cryptographie, Sécurité prouvée, Modes opératoires, Chiffrement de disque, MAC incrémentaux, Sécurité en présence de messages dependant de la clé.

ABSTRACT

This thesis is dedicated to the analysis of modes of operation in the context of disk protection usage. Firstly, we give modes of operation secure in the *Full Disk Encryption* (FDE) model where additional data storage are not allowed. In this context, encryption has to be length preserving which implies length-preserving encryption. However, it is possible to use a value already present in the system, called a *diversifier*, to randomize the encryption and to have a better security. Then, we introduce two methods to analyse symmetric primitive in the very constraint *Key-Dependent Message* (KDM) model which is of interest for disk encryption because the encryption key can end up in the disk. It enables to analyse the KDM security of the Even-Mansour and the Key-Alternating Feistel constructions which are the basis of different block-ciphers. Moreover, knowing that data authenticity cannot be ensured in the FDE model because tag storage is not allowed, we relax this constraint which gives us two models: the *Authenticated Disk Encryption* model (ADE) and the *Fully Authenticated Disk Encryption* (FADE). A secure mode in the ADE model ensures data authenticity of a sector but can be vulnerable to *replay attacks*; and a secure mode in the FADE model ensures the authenticity of the entire disk even against replay attacks. Storage is not the only point to take into account, the read and write delays on a sector is a competitive argument for disk manufacturers since disk performances tightly depend on it and adding the computation of codes of authentication does not help. That is why, we tend to analyse incremental Message Authentication Codes: they have the property to be updatable in a time proportional to the corresponding modification.

KEYWORDS

Cryptography, Security Proof, Modes of Operations, Full Disk Encryption, Incremental MACs, Key-Dependent Message Security.