Centro de Investigación y de Estudios Avanzados
del Instituto Politécnico Nacional

Unidad Zacatenco

Departamento de Computación

Algunos Aspectos en la Seguridad de los Esquemas de Cifrado Tweakable

Tesis que presenta

Vicente Hernández Jiménez

para obtener el grado de

Maestro en Ciencias

en Computación

Director de la tesis

Dr. Debrup Chakraborty

México, D.F. **Diciembre 2013**

Centro de Investigación y de Estudios Avanzados
del Instituto Politécnico Nacional

Zacatenco Campus

Computer Science Department

Some issues on Security of Tweakable Enciphering Schemes

Submitted by

Vicente Hernández Jiménez

as fulfillment of the requirement for the degree of

Master in

Computer Science

Advisor:

Dr. Debrup Chakraborty

Mexico, D.F. **December 2013**

Resumen

Los Esquemas de Cifrado Tweakable (TES, por sus siglas en inglés) son un modo de operación de los cifradores por bloque. Este tipo de esquemas han sido propuestos para ser utilizados en las aplicaciones de cifrado de disco. Durante los últimos años se ha visto una gran actividad en el diseño y seguridad de los TES. La IEEE ha elaborado un estándar (IEEE Std 1619.2-2010) que específica dos TES para ser utilizados en el cifrado de dispositivos de almacenamiento. Esta tesis cubre dos aspectos relacionados con la seguridad de los TES. El primer aspecto se relaciona con el análisis de seguridad de un TES conocido como XCB, y el segundo aspecto está relacionado con la noción de seguridad conocida como Mensajes que Dependen de la Llave.

El modo de operación XCB es uno de los dos esquemas que ha sido estandarizado por la IEEE para el cifrado de dispositivos de almacenamiento. Existen dos versiones de XCB, la primera versión (la cual llamaremos XCBv1) fue propuesta en 2004 y la segunda versión (la cual llamaremos XCBv2) fue propuesta en 2007. XCBv2 es una pequeña modificación a XCBv1. A pesar de que XCBv1 fue declarado un esquema de cifrado seguro no cuenta con una prueba de seguridad. En el caso de XCBv2 los autores proporcionaron una prueba de seguridad del esquema. XCBv2 fue adaptado en el estándar 1619.2-2010. En este trabajo exponemos dos fallas en XCBv2 las cuales ponen en duda su seguridad. La primera falla se debe a modificaciones en su estructura. Estas modificaciones hacen a XCBv2 inseguro para mensajes de longitud arbitraria. Restringiendo la longitud del mensaje se puede solventar este problema. Sin embargo, señalamos que incluso tomando en consideración esta restricción el análisis de seguridad de XCBv2 provisto por sus autores es erróneo. Presentamos un nuevo análisis de la seguridad de XCBv2 y también por primera vez presentamos una prueba de seguridad para XCBv1.

Por otro lado, analizamos la seguridad de los TES bajo una nueva noción de seguridad conocida como Mensajes que Dependen de la Llave (KDM, por sus siglas en inglés). KDM es una noción de seguridad más fuerte que la noción estándar de seguridad de los TES. Damos dos ejemplos de esquemas de cifrado que han sido probados seguros en el modelo estándar, pero que resultan inseguros en la noción de KDM. Para solucionar lo anterior proponemos una transformación para convertir los TES en esquemas seguros bajo la noción de KDM. Nuestra transformación resulta más eficiente y general que la existente en la literatura.

Abstract

Tweakable Enciphering Schemes (TES) are a block cipher mode of operation which can be suitable used for the application of in-place disk encryption. There have been lot of activities in designing secure TES in the last few years. IEEE has also formulated a standard (IEEE Std 1619.2-2010) which specifies two TES for the application of block oriented storage media. In this thesis we deal with two problems related to security of TES. The first is related with the security analysis of the Extended Codebook mode of operation and the second is related with the notion of security called as Key Dependent Messages.

The Extended Codebook (XCB) mode of operation has been standardized by the IEEE for wide block encryption for shared storage media. There are two versions of XCB, the first version (which we call XCBv1) was proposed in 2004 and a later version (which we call XCBv2) was proposed in 2007. XCBv2 is a minor modification of XCBv1. Though XCBv1 was claimed to be a secure TES, the authors neither provided a security bound nor a security proof. For XCBv2 the authors claimed a concrete security bound and provided its proof. XCBv2 was adapted in the standard 1619.2-2010. In this work we expose two flaws in XCBv2 which put in doubt its security. The first flaw is due to modifications in its structure. These modifications make XCBv2 insecure for arbitrary length messages. Restricting the length of the input messages can fix this glitch. However, we point out that even with this restriction the security analysis of XCBv2 as stated by its authors is wrong. We present a new analysis for the security of XCBv2 and as a consequence we state a new bound for its security. Also, for the first time we give a detailed security analysis for the original proposal of XCB (i.e. XCBv1).

On the other hand we analyze the security of TES under the notion of Key Dependent Messages (KDM). KDM is stronger than the standard security notion of TES. We give examples of well proved secure TES in the standard model which fail to be secure under KDM. We propose a transformation to secure an arbitrary TES under KDM. Our transformation turns out to be faster and more general than the other one existing in the literature.

Acknowledgments

I want to thank CINVESTAV for let me be part of such great institution. I also want to thank CONACyT for its financial support along these two years and for the CONACyT project 166763.

I want to give special thanks to Professor Debrup Chakraborty for his dedication and patience towards this project.

I would like to thank Dr. Luis Gerardo de la Fraga and Dr. Guillermo Morales Luna for the time they spent reading and commenting on this document.

I would also like to thank my family for its constant support and encouragement during times of distress.

And last but not least, I would like to thank the invisible force which has guided me up to this point.

> No one who achieves success does so without acknowledging the help of others. The wise and confident acknowledge this help with gratitude.
>
> Alfred North Whitehead

Contents

List of Figures

List of Tables

1. Introduction

> We can only see a short distance
> ahead, but we can see plenty
> there that needs to be done.
>
> Alan Turing

In this thesis we deal with some problems of practical interest in a formal cryptographic setting. In modern cryptography, given a cryptographic scheme Π, one tries to define security of Π in formal terms and then prove that Π indeed provides the security as specified in its security definition. This paradigm, which is popularly known as "provable security", has gained a lot of attention in the last two decades. Now, for a cryptographic scheme to be acceptable, it is required that the scheme is provably secure in some model. This paradigm, though very appealing, should be used with utmost care, as there are many ways in which such a definition-proof paradigm for security can fail. Firstly, it is nontrivial to define security in mathematical terms. Developing a mathematical model which encompasses all possible threats from adversaries may generally not even be possible. There are many recent examples where a provably secure scheme has suffered an attack, as the definition did not consider that specific attack scenario [BFK$^+$12, OST06]. Thus definition of security for a scheme may evolve with time as different possibilities of new attacks arise [BRS02]. Secondly, the proofs attached with cryptographic schemes tends to be complex and lengthy, and thus are prone to errors. In the recent years many errors in security proofs have also been discovered [Jou03, MLMI13]. In this thesis we address two issues related to the provable security paradigm of a class of cryptographic schemes called Tweakable Enciphering Schemes (TES).

Tweakable enciphering schemes are a class of block-cipher mode of operation, which are meant to provide security as that of a tweakable strong pseudorandom permutation (SPRP) (we define this formally in Chapter 3, in page 17). Designing efficient TES which provides the required security in provable terms is a challenging problem. In the last decade there have been some intense activities in designing such schemes and proving their security [HR03, HR04, Hal04, MF04, WFW05, CS06a, MV04, Sar09, CS06b, Hal07, Sar07]. TES are also practically interesting, as they are the most suitable cryptographic schemes for the application of disk encryption (or encryption of any storage media which are organized as sectors). This practical side of TES has lead to standardization activities for the application of disk encryption

[P1611].

To date, there are around ten different proposals for TES. Most of these schemes have a proof of security attached to it. Among the existing schemes, XCB [MV04] is one of the oldest and also this is part of the IEEE Std 1619.2-2010 standard for disk encryption. In this thesis we show that the security proof for XCB (given in [MF07]) is wrong. We provide several counter examples which shows the security theorem of XCB to be false. Our analysis on XCB is not only of theoretical interest, as some of the attacks that we point out can have practical relevance.

The second problem that we deal with in this thesis is a definitional issue. The standard security definition for TES does not consider the possibility of the scheme encrypting its own key. This possibility, of an encryption scheme encrypting its own key has long been known and has traditionally been considered to be an abuse of encryption. During the formulation of the standard IEEE Std 1619.2-2010, this issue for TES surfaced out, and the standards committee considered that a "good" TES should be able to encrypt its own key securely. This property has been called as Key Dependent Message (KDM) security. This observation of the P1619 committee (committee responsible for the IEEE Std 1619.2-2010 standard) has recently lead to some constructions of TES which can securely encrypt their own keys. In this problem, we have two contributions. Firstly we explicitly show some weaknesses in existing TES when they encrypt their own keys, these attacks are new to the literature. Secondly, we propose a new transformation which can convert any (SPRP secure) TES to a KDM secure TES. We also argue that our proposed transformation is better both in functionality and efficiency compared to the existing one.

The rest of this chapter is organized as follows. We begin with some background material in Sections 1.1 and 1.2. In Section 1.1 we mention three basic principles of modern cryptography: precise definitions of security, clearly stated and well known believed assumptions, and proofs of security. In Section 1.1 we informally discuss TES and its usage. In Section 1.3 we elaborate on the problems addressed in this thesis. We finish this chapter with a discussion on the contents of the chapters that follow.

1.1 Modern Cryptography: The Paradigm of Provable Security

Historically, cryptography was more of an art than a science. Cryptographic schemes were designed in an ad-hoc manner and then evaluated based on their perceived complexity or cleverness. Schemes of such kind, no matter how clever, were eventually broken. Notable examples are the Caesar and Vigenère ciphers [Sti02].

Modern cryptography tries to organize itself in a mathematical fashion. It rests on scientific foundations, it focuses in three basic principles [KL07]:

- **Formulation of rigorous and precise definitions of security:** Formal definitions are important for design, analysis and usage of cryptographic schemes:

  - In terms of design, it is much better to define what is needed first and then begin the construction of the scheme.
  - If a strict definition of security exists, then analysis of the scheme can be done much more easily, and one can precisely measure (in some cases) the level up to which a scheme satisfies the required security as specified by its definition.
  - A practical problem may have specific security requirements. A precise model of security of a scheme would in most cases clearly indicate the practical scenarios where the scheme can be used.

- **Reliance on clearly stated assumptions:** Security of most practical cryptographic schemes depends on un-proven assumptions. The most used assumptions in cryptography are the intractability of factorization, the intractability of the discrete logarithm problem in certain groups, the existence of one-way functions, etc. It is unlikely that these assumptions would be settled in near future, as answers to these questions are related to some very old unsolved problems in computational complexity theory. Thus, to design workable cryptographic solutions we need to rely on assumptions. When the security of a cryptographic construction relies on an assumption, the assumption must be precisely stated and it must be as minimal as possible. Moreover, schemes should be based only on well studied assumptions.

- **Proof of Security:** Cryptographic constructions should be accompanied by a proof of security with respect to a formulated definition. Modern cryptography stresses the importance of proofs of security, the fact that the ciphertext looks garbled, does not necessarily mean that a sophisticated adversary is unable to break the scheme.

A definition of security must accurately model the real world in order to deliver a mathematical promise of security. Sometimes it had happened that implementations of provably secure schemes had weaknesses. For example, there are encryption schemes that were proven secure and then implemented on smart cards. Due to physical properties of the smart cards, it was possible for an adversary to monitor the power usage of the smart card when the encryption scheme was being run, and it turned out that this information could be used to determine the key [KL07]. The problem was simply that there was a mismatch between the definition and the real world implementation of the scheme.

Such gaps in security definitions are very likely, as security in practice is complex and often it is difficult to judge its various dimensions. Thus, it is expected that a security definition is dynamic in nature, and it evolves as new threats are discovered.

It is to be understood that a security theorem is a mathematical statement, which only states the extent to which a scheme achieves security in accordance to the definition. Thus, interpreting such theorems in the real world should be done carefully.

Moreover, most proofs of security are based on unproven assumptions, thus security theorems almost never state the absolute truth.

In this work we deal with two problems, both of which highlight some subtleties of the paradigm of provable security. We show a glaring error in a security theorem of a scheme called XCB. This proof had been around in the literature for more than five years. There have been other cases in the literature where a long-standing proof has been shown to be incorrect. In the other problem, we deal with a definition which we believe gets closer to a real world scenario. This definition deals with encryption schemes ciphering their own decryption key. A scenario which has been largely ignored before. We show that when this new dimension of security is considered then most schemes which are proven secure in the standard definition breaks down. This shows that security definitions need to be updated, and such updated definitions also call for new constructions.

1.2 Tweakable Enciphering Schemes

This work focuses on a specific type of encryption scheme known as Tweakable Enciphering Schemes (TES). It has been suggested that tweakable enciphering schemes can be suitably used for the application of low level disk encryption. For an intuitive understanding of TES, let us imagine that we want to do a low level encryption of a hard disk. A low level disk encryption scheme resides in the disk controller, it sees the hard disk as a collection of sectors and is ignorant of the high level structures like files, directories, etc. A low level disk encryption scheme functions as follows:

- When the disk controller is asked to write data, it passes the data through an encryption algorithm and writes the output in the hard disk, so the hard disk only contains cipher data.
- When the disk controller is asked to read data, it reads cipher data from the hard disk; then it uses the decryption algorithm and delivers as output plain data.

For an encryption algorithm to function in the above mentioned application it needs to have the following characteristics:

- **Length preserving:** As the smallest unit that gets written or read from a disk is the sector, hence a low level disk encryption scheme does sector wise encryption. An important property that is required is that the length of the ciphertext should be same as that of the plaintext. Hence there is no scope to include states, nonces, etc. as part of the cipher text, which are common in other symmetric encryption schemes. The requirement of length preservation, also makes such schemes inherently deterministic.
- **Ciphertext Variability:** As the encryption algorithm is deterministic, so if two sectors contains exactly the same information then the corresponding cipher data of both sectors would be exactly the same. This is not desirable, as

this may reveal some pattern to an adversary. To mend this scheme, disk encryption algorithms takes in as input an extra public quantity called the tweak. For the application, sector addresses are considered as the tweaks. Tweakable enciphering schemes obtain ciphertext variability through the use of the tweaks.

- **Security:** Disk encryption schemes must be secure against adaptive chosen plaintext and adaptive chosen ciphertext adversaries. Such schemes are generally called CCA secure schemes (secure against chosen ciphertext attacks). Achieving CCA security means that no adversary can be able to distinguish the ciphertexts from random strings, and additionally the attacker must not be able to modify the ciphertext so that it gets decrypted to something meaningful.

The properties which we described above are all provided by *Tweakable Enciphering Schemes*. TES are length preserving, their security model is a strong pseudorandom permutation indexed by a tweak. The tweak is an extra public parameter that increases the variability of the ciphertext, i.e., if two different plaintext are encrypted using the same key but the tweaks are different the resulting ciphertexts will be different. Other important property of TES is that if any part of the ciphertext is changed the plaintext obtained after decryption looks like random strings. To date there are more than ten different proposals for such schemes, additionally there is a recent standard (IEEE Std 1619.2-2010) on such schemes formulated by IEEE. We describe TES more formally in Chapter 3, in page 17.

This thesis is devoted to the study of two problems related to security of TES. We discuss the two problems in the next section.

1.3 The Problems Considered in this Thesis

This work deals with two major problems. We describe them next:

1. **Security of XCB:** XCB is a TES which was first proposed in 2004 [MV04]. The original proposal did not have a security proof, later in 2007 [MF07] an updated version of XCB was proposed along with a proof of security. For convenience we shall call these two versions of XCB as XCBv1 and XCBv2 respectively. Later XCBv2 was incorporated in the standard IEEE Std 1619.2-2010. We show that the security of XCBv2 as claimed in [MF07] is wrong. We show a simple attack, which establishes that XCBv2 is not a tweakable SPRP. This situation can be repaired, if we consider a restricted message space for XCBv2, the restriction required is that the messages should be such that their lengths are multiples of the block length of the block cipher. If this is the case then our simple attack does not work, but still we show that the security bound claimed in [MF07] is erroneous. The counter examples that we use for refuting the claimed bound involves some interesting combinatorial analysis which heavily depends on some techniques recently used in [IOM12] for analyzing another standardized authenticated encryption scheme called GCM.

We also provide a new security proof for the restricted version of XCBv2. The security bound that we derive is quite different (and weaker) than that claimed in [MF07]. Our analysis thus reveals that there are multiple problems in XCB, and it is not clear why this scheme was standardized.

Another contribution on this problem is that we provide a complete security proof for XCBv1, such a proof does not exist in the literature.

2. **KDM Security for TES:** The other problem that we consider is: "whether it is safe for a TES to encrypt its own key ". This scenario of "self-encryption" has been discussed in the Security in Storage Working Group (IEEE P1619). Initially the working group considered this scenario as a theoretical possibility, but when the group was informed that there existed real world implementations which fall in this scenario the "self-encryption" was considered as a real problem.

   This notion of "self-encryption" was captured in the definition of Key Dependent Messages (KDM) proposed and studied by Black, Rogaway, and Shrimpton [BRS02]. These definitions were later extended in [HK07]. A definition for KDM security for TES also appears in [HK07].

   TES are known to be secure in terms of strong pseudo-random permutation (SPRP), however we do not have any security proof in the sense of KDM for the existing schemes. Moreover there are also no known attacks which shows KDM insecurity of the existing TES. We explicitly show KDM insecurity of two TES, XCB and HCTR.

   In a recent work Bellare, Cash and Keelveedhi [BCK11] gave two transformations to convert a tweakable enciphering scheme secure in the SPRP sense into a KDM secure one. The transformations suggested to add an additional layer of encryption to the existing TES to thwart attacks of KDM type. We provide a new transformation for this task. We provide a detailed security argument to establish that our transformation really work. Moreover we show that our transformations are better than the one proposed in [BCK11] both in terms of efficiency and functionality.

1.4 Organization of the Thesis

The rest of the document is organized into seven chapters; next we discuss in short the contents of these chapters:

- In Chapter 2, we give some notations that we use throughout the document. We introduce the Game Playing Technique, a widely used technique for security proofs of encryption schemes. In this chapter we also give the standard definitions of security.

- In Chapter 3, we discuss about Tweakable Enciphering Schemes. We go into detail about their definition and their security. We give a brief history of TES

and also we describe the TES EME2, XCB and HCTR. The reason that we chose these schemes is that the first two have been standardized by the IEEE for wide-block encryption for shared storage media. In case of HCTR, it is an efficient scheme, and later we also use HCTR as an example for KDM insecure scheme.

- In Chapter 4, we focus on XCB and its security proof. There are two versions of XCB, the security proof given by the authors of XCB was for the second version while the first remained without a proof. In this chapter, first we describe some issues that were not considered in the original proof of XCB, and that can lead to unwanted situations. And then we proceed with the fix of the proof of the second version. In the last section of this chapter we also give for the first time a security proof for the first version.

- In Chapter 5, we introduce the notion of Key Dependent Messages. We give a definition and also we establish what does it mean for an encryption scheme to be KDM secure. Additionally we present attacks on TES that fail to be secure in the sense of KDM. At the end of this chapter we present the transformation derived from the work of Bellare, Cash and Keelveedhi to turn a secure TES into a KDM secure one.

- In Chapter 6, we present MStE, our transformation to turn a secure TES into a KDM secure one. We accompany the transformation with its security proof.

- In Chapter 7, we show the results of the implementation of EME2, XCB and HCTR. We take these implementations as a starting point to compare our proposed transformation to that of Bellare et al. Outcomes of this comparison, where our transformation results the winner, are shown in the last section of this chapter.

- In Chapter 8, we conclude this work, and discuss about some issues which we would like to take up for future study.

2. Theoretical Framework

> It is possible to build a cabin with no foundations, but not a lasting building.
>
> Eng. Isidor Goldreich

In this chapter we introduce the notation and some basic concepts used in this work. We give a brief introduction to a proof technique used in the field of provable security, which is called the game-playing technique. We also define pseudorandom functions, pseudorandom permutations, block ciphers and almost universal hash functions. These objects would be fundamental to the schemes that we describe in the rest of the thesis.

2.1 General Notation

We denote the set of all binary strings by $\{0,1\}^*$ and the set of all n-bit strings by $\{0,1\}^n$. We shall denote the concatenation of two strings $X, Y \in \{0,1\}^*$ by $X||Y$. For a bit string $X \in \{0,1\}^*$, $|X|$ is its length in bits, and $|X|_\ell = \lceil |X|/\ell \rceil$ is the length in ℓ-bit blocks. For a bit string X and an integer ℓ such that $|X| \geq \ell$, $\mathsf{msb}_\ell(X)$ is the most significant ℓ bits (the leftmost ℓ bits) of X, and $\mathsf{lsb}_\ell(X)$ is the least significant ℓ bits (the rightmost ℓ bits) of X. For non-negative integers a and ℓ with $a \leq 2^\ell - 1$, $\mathsf{bin}_\ell(a)$ will denote the ℓ-bit binary representation of a, i.e., if $a = a_{\ell-1}2^{\ell-1} + \ldots + a_1 2 + a_0$ for $a_{\ell-1}, \ldots, a_1, a_0 \in \{0,1\}$, then $\mathsf{bin}_\ell(a) = a_{\ell-1} \ldots a_1 a_0 \in \{0,1\}^\ell$. For a bit string $X = X_{\ell-1} \ldots X_1 X_0 \in \{0,1\}^\ell$, $\mathsf{int}(X)$ will denote the integer $X_{\ell-1}2^{\ell-1} + \ldots + X_1 2 + X_0$. For a finite set $\mathcal{X}$, $|\mathcal{X}|$ denotes its cardinality, and $X \xleftarrow{\$} \mathcal{X}$ means the uniform sampling of an element from $\mathcal{X}$ and assigning it to X.

The set of n-bit strings, $\{0,1\}^n$, is also sometimes regarded as $GF(2^n)$, the finite field with 2^n elements. An n-bit string $a_{n-1} \ldots a_1 a_0 \in \{0,1\}^n$ corresponds to a formal polynomial $a(x) = a_{n-1} + a_{n-2}x + \ldots + a_1 x^{n-2} + a_0 x^{n-1}$. When $n = 128$ and the irreducible polynomial is not specified, we assume $p(x) = 1 + x + x^2 + x^7 + x^{128}$ as the irreducible polynomial. For $X, Y \in GF(2^n)$, $X \oplus Y$ and XY will denote addition and multiplication in the field respectively.

2.2 Game-Playing Technique

Game-playing technique is widely used in provable security. Bellare and Rogaway [BR06] gave a framework for the proofs based on this technique. In this section we give a brief introduction to the technique based on the work of Bellare and Rogaway.

A game-playing proof in cryptography is any proof where one conceptualizes the adversary's interaction with its environment as a kind of game, the proof proceeds by constructing a "chain" of such games. The *adversary* is considered to be a probabilistic algorithm which models the source of all possible threats.

In our treatment, games are programs written in pseudocode; as we develop it, game-playing centers around making disciplined transformations to code. A game consists of an *initialization procedure* (Initialize), a *finalization procedure* (Finalize) and named *oracles* (each a procedure). The adversary makes calls to the oracles. The initialization and finalization procedures may be absent, and there may be any number of oracles. All variables in a game are global, and they are not visible to the adversary.

To begin, variables are given initial values. Integer variables are initialized to 0; Boolean variables are initialized to false; string variables are initialized to the empty string ε; set variables are initialized to the empty set $\emptyset$; and array variables hold the value undefined $\perp$, at every point. These conventions often enable omitting explicit initialization code.

The Initialize procedure is the first to execute, possibly producing an output. This is provided as input to the adversary procedure A, which now runs. The only way for an oracle to return a value to the adversary is via a return statement. When adversary A halts, possibly with some *adversary output*, we call Finalize providing it any such output. The Finalize procedure returns a string that we call the *game output*. We write $\Pr[A^G \Rightarrow 1]$ for the probability that the adversary output is 1 when we run game G with adversary A.

2.2.1 Advantages

If G and H are games and A is an adversary, let $\mathbf{Adv}(A^G, A^H) = \Pr[A^G \Rightarrow 1] - \Pr[A^H \Rightarrow 1]$. This represents the *advantage* of the adversary in distinguishing the games G and H, measured via adversary output.

We will often use the fact that

$$\mathbf{Adv}(A^G, A^I) = \mathbf{Adv}(A^G, A^H) + \mathbf{Adv}(A^H, A^I),$$

for any games G, H, I and any adversary A. This follows from the definition of the advantage:

$$\begin{aligned} \mathbf{Adv}(A^G, A^I) &= \Pr[A^G \Rightarrow 1] - \Pr[A^I \Rightarrow 1] \\ &= \Pr[A^G \Rightarrow 1] - \Pr[A^H \Rightarrow 1] + \Pr[A^H \Rightarrow 1] - \Pr[A^I \Rightarrow 1] \\ &= \mathbf{Adv}(A^G, A^H) + \mathbf{Adv}(A^H, A^I). \end{aligned}$$

2.2.2 Fundamental lemma of game playing

Fundamental Lemma is about the probability that an adversary can distinguish between games (programs) that differ in a certain syntactic way.

Let G and H be games and let *bad* be a flag that occurs in both of them. Then we say that G and H are *identical-until-bad* if their code is the same unless the *bad* flag is set to true.

We write $\Pr[A^G \text{ sets } \mathsf{bad}]$ to refer to the probability that the flag *bad* is true at the end of the execution of the adversary A with game G.

The fundamental lemma says that the advantage of an adversary in distinguishing a pair of *identical-until-bad* games is at most the probability that its execution sets *bad* in one of them.

Lemma 2.2.1 (Fundamental lemma of game-playing)**.** *Let G and H be identical-until-bad games and let A be an adversary. Then*

$$\boldsymbol{Adv}(A^G, A^H) \leq \Pr[A^G \textit{ sets } \mathsf{bad}].$$

The proof of the above Lemma and more information about the game-playing technique can be found in [BR04].

2.3 Pseudorandom Functions and Pseudorandom Permutations

Useful definitions to understand the security notions we cover in this work, are those of pseudorandom functions (PRFs), pseudorandom permutations (PRPs) and strong pseudorandom permutations (SPRPs). In this section we introduce PRFs, PRPs and SPRPs, and discuss their basic properties.

2.3.1 Function families

A *function family* is a map $F : \mathcal{K} \times D \to R$. Here $\mathcal{K}$ is the set of keys of F and D is the domain of F and R is the range of F. The set of keys $\mathcal{K}$ and the range R are finite, and all the sets are nonempty. The two-input function F takes a key K and an input X to return a point Y we denote by $Y = F(K, X)$.

For any key $K \in \mathcal{K}$ we define the map $F_K : D \to R$ by $F_K(X) = F(X, Y)$. We call the function F_K an instance of a function family F. Thus F specifies a collection of maps, one for each key. That's why we call F a *function family* or *family of functions*.

There is some probability distribution on the set of keys $\mathcal{K}$. Unless otherwise indicated, this distribution will be the uniform one. We denote by $f \xleftarrow{\$} F$ the operation: $K \xleftarrow{\$} \mathcal{K}$; $F \leftarrow F_K$. In other words, let f be the function F_K where K is a randomly chosen key. We are interested in the input-output behavior of this randomly chosen instance of the family.

Game Real_F	Game Rand_R
procedure Initialize	**procedure Fn**(x)
$K \xleftarrow{\$} \mathcal{K}$	if $T[x] = \perp$ then
	$T[x] \xleftarrow{\$} R$
procedure Fn(x)	return $T[x]$
return $F_K(x)$	

Figure 2.1: Games used to define PRFs.

A *permutation* is a bijection whose domain and range are the same set. We say that F is a family of permutations if for every $K \in \mathcal{K}$ F_K is a permutation.

2.3.2 Random functions and permutations

Let $\mathsf{Func}(\mathcal{D}, \mathcal{R})$ be the set of all functions from $\mathcal{D}$ to $\mathcal{R}$. By a *random function* (from $\mathcal{D}$ to $\mathcal{R}$) we mean a function selected uniformly at random from the set $\mathsf{Func}(\mathcal{D}, \mathcal{R})$. The randomness of the function refers to the way it was chosen, not to an attribute of the selected function itself.

Game Rand_R, shown on the right hand side of Fig. 2.1 provides to an adversary an oracle **Fn** that implements a random function. This means that on any query the oracle returns a random point from R as a response subject to the restriction that if it is queried twice on the same point, the response is the same both times. The game maintains the function in the form of a table T where $T[X]$ holds the values of the function at X. Initially, the table is everywhere undefined.

Let $\mathsf{Perm}(\mathcal{D})$ be the set of all permutations from $\mathcal{D}$ to $\mathcal{D}$. A *random permutation* on $\mathcal{D}$ is a permutation selected uniformly random from $\mathsf{Perm}(\mathcal{D})$. The game Perm_D shown on the right hand side of Fig. 2.2 provides to an adversary access to an oracle that implements a random permutation over the finite set D.

2.3.3 Pseudorandom functions

A PRF is a family of functions with the property that the input-output behavior of a random instance of the family is "computationally indistinguishable" from that of a random function.

Definition 2.3.1. *Let $F : \mathcal{K} \times D \to R$ be a family of functions, and let A be an algorithm that takes an oracle and returns a bit. We consider two games as described in Fig. 2.1. The* prf-advantage *of A is defined as*

$$\mathbf{Adv}^{\text{prf}}_F(A) = \Pr[A^{\text{Real}_F} \Rightarrow 1] - \Pr[A^{\text{Rand}_R} \Rightarrow 1]. \tag{2.1}$$

Game Real_F picks a random instance F_K of a family F and then runs adversary A with oracle $\mathbf{Fn} = F_K$. Adversary A interacts with its oracle, querying it and

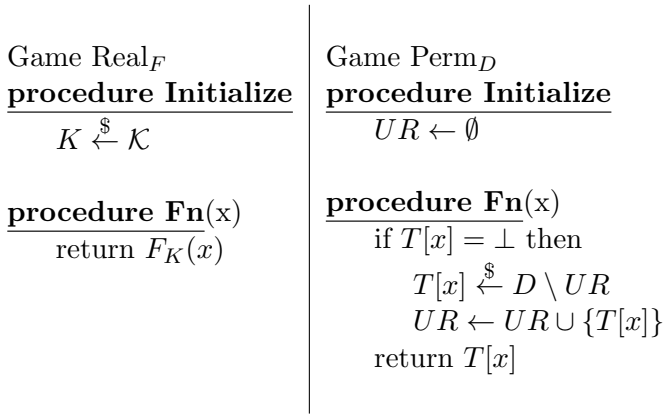

Figure 2.2: Games used to define PRPs.

getting back answers, and eventually outputs a bit. Game Rand$_R$ implements **Fn** as a random function with range R. Again, adversary A interacts with the oracle, eventually returning a bit that is the output of the game.

The task of the adversary is to determine which game it is playing with based on the input-output behavior of **Fn**. Outputting the bit "1" means that A "thinks" it is interacting with Real$_F$, outputting the bit "0" means that A thinks it is interacting with Rand$_R$.

Another way we write Eq. (2.1) using the notation $\mathsf{Func}(\mathcal{D}, \mathcal{R})$ is the following:

$$\mathbf{Adv}^{\mathrm{prf}}_F(A) = \Pr[K \xleftarrow{\$} \mathcal{K} : A^{F_K} \Rightarrow 1] - \Pr[\rho \xleftarrow{\$} \mathsf{Func}(\mathcal{D}, \mathcal{R}) : A^{\rho} \Rightarrow 1].$$

We say the family F is prf-secure if $\mathbf{Adv}^{\mathrm{prf}}_F(A)$ is "small" for all "efficient" adversaries. It should be noted that the family F is public and known to the adversary.

Note that we leave the terms "small" and "efficient" undefined, which is quite standard in the paradigm of concrete security. These terms are to be interpreted in the context. We will follow this for all later security definitions, i.e., we shall never define "small" advantage and "efficient" adversaries.

2.3.4 Pseudorandom permutations

A family of functions $F : \mathcal{K} \times D \to R$ is a Pseudorandom Permutation (PRP) if the input-output behavior of a random instance of the family is "computationally indistinguishable" from that of a random permutation on D.

In game Real$_F$ of Fig. 2.2, **Fn** is a random instance of F, i.e, it is the function F_K where K is a random chosen key. In game Perm$_D$, **Fn** is a random permutation on D.

As before the task for the adversary A is to determine which game it is playing with based on the input-output behavior of **Fn**.

Definition 2.3.2. *Let $F : \mathcal{K} \times D \to D$ be a family of permutations, and let A be an algorithm that takes an oracle* **Fn** *for a function* **Fn** $: D \to D$*, and returns a bit. We*

consider two games as described in Fig. 2.2. The prp-advantage *of A is defined as*

$$\mathbf{Adv}^{\mathrm{prp}}_{F}(A) = \Pr[A^{\mathrm{Real}_F} \Rightarrow 1] - \Pr[A^{\mathrm{Perm}_D} \Rightarrow 1]. \tag{2.2}$$

The intuition is similar to that for Definition 2.3.1. The difference is that here the object that F is being compared with is no longer a random function, but rather a random permutation. We also sometimes write Eq. (2.2) as:

$$\mathbf{Adv}^{\mathrm{prp}}_{F}(A) = \Pr[K \xleftarrow{\$} \mathcal{K} : A^{F_K} \Rightarrow 1] - \Pr[\pi \xleftarrow{\$} \mathsf{Perm}(\mathcal{D}) : A^{\pi} \Rightarrow 1].$$

When F is a family of permutations, one can also consider the case where the adversary gets, in addition to the function $\mathbf{Fn}$, an oracle for $\mathbf{Fn}^{-1}$. Giving an adversary access to both oracle we can define another object which is called a Strong Pseudorandom Permutation (SPRP).

Definition 2.3.3. *Let $F : \mathcal{K} \times D \to D$ be a family of permutations, and let A be an algorithm that takes an oracle $\mathbf{Fn}$ for a function $\mathbf{Fn} : D \to D$, and also an oracle $\mathbf{Fn}^{-1}$ for the function $\mathbf{Fn}^{-1} : D \to D$, and returns a bit. We consider two games as described in Fig. 2.3. The* sprp advantage *of A is defined as*

$$\mathbf{Adv}^{\pm\mathrm{prp}}_{F}(A) = \Pr[A^{\pm\mathrm{Real}_F} \Rightarrow 1] - \Pr[A^{\pm\mathrm{Perm}_D} \Rightarrow 1]. \tag{2.3}$$

In the above definition the adversary has more power: not only it can query $\mathbf{Fn}$, but it can directly query $\mathbf{Fn}^{-1}$. We also write Eq. (2.3) as:

$$\mathbf{Adv}^{\pm\mathrm{prp}}_{F}(A) = \Pr[K \xleftarrow{\$} \mathcal{K} : A^{F_K, F_K^{-1}} \Rightarrow 1] - \Pr[\pi \xleftarrow{\$} \mathsf{Perm}(\mathcal{D}) : A^{\pi,\pi^{-1}} \Rightarrow 1].$$

We say the family F is prp-secure if $\mathbf{Adv}^{\mathrm{prp}}_{F}(A)$ is "small" for all efficient adversaries A. In the same manner the family F is sprp-secure if $\mathbf{Adv}^{\pm\mathrm{prp}}_{F}(A)$ is "small".

2.3.5 Relations between the notions

For each advantage notion $\mathbf{Adv}^{xxx}_{\Pi}$ we write $\mathbf{Adv}^{xxx}_{\Pi}(\mathcal{RC})$ for the maximal value of $\mathbf{Adv}^{xxx}_{\Pi}(A)$ over all adversaries A that use resources at most $\mathcal{RC}$. Resources of interest are the running time t, the number of queries q and the total length of all queries σ.

PRP/PRF switching lemma

The PRP/PRF Switching Lemma says that a truly random permutation looks very much like a truly random function.

Lemma 2.3.1 (PRP/PRF Switching Lemma [BR04])**.** *Let A be an adversary that asks at most q oracle queries. Then*

$$|\Pr[\pi \xleftarrow{\$} \mathsf{Perm}(\mathcal{D}) : A^{\pi} \Rightarrow 1] - \Pr[\rho \xleftarrow{\$} \mathsf{Func}(\mathcal{D}, \mathcal{D}) : A^{\rho} \Rightarrow 1]| \leq \frac{q(q-1)}{2|\mathcal{D}|}.$$

From the above lemma we obtain

$$|\mathbf{Adv}^{\mathrm{prf}}_{F}(A) - \mathbf{Adv}^{\mathrm{prp}}_{F}(A)| \leq \frac{q(q-1)}{2|\mathcal{D}|}.$$

When $|\mathcal{D}|$ is large, $\frac{q(q-1)}{2|\mathcal{D}|}$ is "small". Thus prp-security implies prf-security.

Game $\pm\text{Real}_F$
procedure Initialize
$K \xleftarrow{\$} \mathcal{K}$

procedure Fn(x)
return $F_K(x)$

procedure Fn^{-1}(x)
return $F_K^{-1}(x)$

Game $\pm\text{Perm}_D$
procedure Initialize
$UR \leftarrow \emptyset$; $UD \leftarrow \emptyset$

procedure Fn(x)
if $T[x] = \perp$ then
$T[x] \xleftarrow{\$} D \setminus UR$
$S[T[x]] \leftarrow x$
$UR \leftarrow UR \cup \{T[x]\}$; $UD \leftarrow UD \cup \{x\}$
return $T[x]$

procedure Fn^{-1}(y)
if $S[y] = \perp$ then
$S[y] \xleftarrow{\$} D \setminus UD$
$T[S[y]] \leftarrow y$
$UD \leftarrow UD \cup \{S[y]\}$; $UR \leftarrow UR \cup \{y\}$
return $T[x]$

Figure 2.3: Games used to define SPRPs.

SPRP implies PRP

Let $F : \mathcal{K} \times \mathcal{D} \to \mathcal{D}$ be a family of permutations and let A be a PRP attacking adversary. Suppose that A runs in time t, asks q queries and these queries total σ length. Then there exists a SPRP attacking adversary B that runs in time t, asks q queries, and these queries total σ length, such that

$$\mathbf{Adv}_F^{\text{prp}}(A) \leq \mathbf{Adv}_F^{\pm\text{prp}}(B).$$

So sprp-security implies prp-security which implies prf-security. This is the reason we are interested in proving sprp-security.

2.4 Cryptographic Primitives

Cryptographic primitives are cryptographic algorithms designed to do a very specific task. By themselves, they do not provide something meaningful to an end user. Instead they are used as building blocks in the construction of crypto-systems. A primitive can provide different functionality depending on its mode of operation or usage. We will discuss two of these primitives: block ciphers and universal hash functions.

2.4.1 Block cipher

A block cipher is a function $E : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$. The first input is the key. The second is called the plaintext, and the output is called ciphertext. The *key-length* k and the *block-length* n are parameters associated to the block cipher.

For each key $K \in \{0,1\}^k$ let $E_K : \{0,1\}^n \to \{0,1\}^n$ be the function defined by $E_K(\cdot) = E(K,\cdot)$. For any block cipher, and any key K, it is required that the function E_K be a permutation on $\{0,1\}^n$. This means that it is a bijection of $\{0,1\}^n$ to $\{0,1\}^n$. For every $C \in \{0,1\}^n$ there is exactly one $M \in \{0,1\}^n$ such that $E_K(M) = C$. Accordingly E_K has an inverse, and we denote it by E_K^{-1}. We have $E_K^{-1}(E_K(M)) = M$ and $E_K(E_K^{-1}(C)) = C$ for all $M, C \in \{0,1\}^n$.

Ideally a block cipher is considered to be secure if it is a SPRP. Unfortunately we do not have any proof that the known block ciphers are SPRP. Hence we consider a block cipher to be secure if it behaves like a SPRP, which is assumed to be true for the block ciphers currently in use. The confidence in the assumption rests only on the absence of discovered attacks.

Block Cipher Modes of Operation. Block ciphers modes of operation are algorithms that specify the use of a block cipher to encrypt arbitrary long messages to provide confidentiality or authentication. The most common modes of operation are ECB, CBC, CFB, OFB and CTR [Dwo01].

2.4.2 Universal hash function

A hash function is used to construct a short "fingerprint" of some data; if the data is altered, then the fingerprint will no longer be valid. Let h be a hash function and let x be some data. The corresponding fingerprint, often referred as *message digest*, is defined to be $y = h(x)$.

A *keyed hash family* H is a family of functions $H : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$, where $\mathcal{K}$ is the key space and for any $K \in \mathcal{K}$, $h(K,\cdot) = h_K(\cdot)$ is a hash function. $\mathcal{X}$ is the set of possible messages and $\mathcal{Y}$ is the set of possible message digests.

$\mathcal{X}$ could be a finite or infinite set while $\mathcal{Y}$ is always a finite set. If $\mathcal{X}$ is a finite set it is always assumed that $|\mathcal{X}| \geq |\mathcal{Y}|$.

Definition 2.4.1 ([Sti91])**.** *A keyed hash family $H : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ is called* ϵ-almost universal *(ϵ-AU) provided that the following condition is satisfied for every $x, x' \in \mathcal{X}$ such that $x \neq x'$:*

$$\Pr[K \xleftarrow{\$} \mathcal{K} : h_K(x) = h_K(x')] \leq \epsilon.$$

H is called *ϵ-almost xor universal* (ϵ-AXU) if for all $x, x' \in \mathcal{X}$ such that $x \neq x'$ and all $y \in \mathcal{Y}$ we have $\Pr[K \xleftarrow{\$} \mathcal{K} : h_K(x) \oplus h_K(x') = y] \leq \epsilon$.

3. Tweakable Enciphering Schemes

> Any one who considers
> arithmetical methods of
> producing random digits is, of
> course, in a state of sin.
>
> John von Neumann

As mentioned in the introduction this thesis largely deals with security properties of some tweakable enciphering schemes (TES). In this chapter we will discuss the main ideas behind this important class of enciphering schemes. We begin this chapter with a brief introduction to tweakable block ciphers (TBC) which are the true predecessors of TES. In Section 3.2 we formally define the syntax and security of TES . In Section 3.3 we briefly trace the history of existing TES and finally in 3.4 we describe in details some of the existing TES, whose description would be necessary for later chapters.

3.1 Tweakable Block Ciphers

Tweakable Block Ciphers (TBC) were a predecessor idea of TES. They were proposed by Liskov, Rivest and Wagner [LRW02] as a cryptographic primitive. The main goal of TBC is to give variability to block cipher outputs while maintaining a fixed key.

An n-bit block cipher $E : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ is inherently deterministic, i.e, every encryption of a given message with a given key will be the same. TBC provide variability to a block cipher by using a special public quantity called the tweak. A TBC is formally defined as a function

$$\tilde{E} : \{0,1\}^k \times \{0,1\}^t \times \{0,1\}^n \to \{0,1\}^n.$$

Thus a TBC takes three inputs: a key $K \in \{0,1\}^k$, a tweak $T \in \{0,1\}^t$ and a message $M \in \{0,1\}^n$, and produce as output a ciphertext $C \in \{0,1\}^n$. The tilde above E serves as a reminder that the block cipher is tweakable.

The difference between the key and the tweak is that the function of the key is to provide uncertainty to the adversary, and the role of the tweak is to provide variability, note that unlike the key the tweak is public. This role separation is the reason to consider TBC as a separate cryptographic primitive.

A tweakable block cipher has the property that changing the tweak is less costly than changing the key. Many block ciphers have the property that changing the encryption key is very expensive, since a "key setup" operation needs to be performed, moreover, in some applications the key may be hard coded in a device, in such cases a key change may amount to change of a circuit or a significant part of the hardware. Another important property of TBC is that even if an adversary has control of the tweak input TBC remain secure, because each fixed setting of the tweak gives rise to a different, apparently independent, family of block cipher.

Security of Tweakable Block Cipher . Let $\mathsf{Perm}^t(n)$ be the set of functions $\pi : \{0,1\}^t \times \{0,1\}^n \to \{0,1\}^n$ such that $\pi(T,\cdot)$ is a permutation for all $T \in \{0,1\}^t$. We call such a permutation as a tweak indexed permutation. We define the advantage an adversary A in distinguishing a tweakable block cipher from a tweakable random permutation as

$$\begin{aligned}\mathbf{Adv}^{\pm\widetilde{\mathrm{prp}}}_{\tilde{E}}(A) &= \Pr[K \xleftarrow{\$} \{0,1\}^k : A^{\tilde{E}_K(\cdot,\cdot),\tilde{E}_K^{-1}(\cdot,\cdot)} \Rightarrow 1] \\ &\quad - \Pr[\pi \xleftarrow{\$} \mathsf{Perm}^t(n) : A^{\pi(\cdot,\cdot),\pi^{-1}(\cdot,\cdot)} \Rightarrow 1].\end{aligned}$$

Note, here all the oracles takes in two inputs, which signifies that the adversary is allowed to choose both the message and the tweak for each oracle call. A tweakable block cipher $\tilde{E}$ is considered secure if $\mathbf{Adv}^{\pm\widetilde{\mathrm{prp}}}_{\tilde{E}}(A)$ is "small" for all "efficient" adversaries A.

Liskov, Rivest and Wagner [LRW02] proposed two constructions for TBC. Given a block cipher $E_K : \{0,1\}^n \to \{0,1\}^n$ and a tweak $T \in \{0,1\}^n$, we have the following TBC

- $\tilde{E}_K(T,M) = E_K(T \oplus E_K(M))$
- $\tilde{E}_{K_1,K_2}(T,M) = E_{K_1}(M \oplus h_{K_2}(T)) \oplus h_{K_2}(T)$, where h is ϵ-AXU.

3.2 Tweakable Enciphering Schemes

Let a message space $\mathcal{M}$ be a set of strings, i.e., $\mathcal{M} = \bigcup_{i \in I}\{0,1\}^i$ for some nonempty index set $I \subseteq \mathbb{N}$. A *length-preserving permutation* is a map $\pi : \mathcal{M} \to \mathcal{M}$ where $\mathcal{M}$ is a message space and π is a permutation and $|\pi(P)| = |P|$ for all $P \in \mathcal{M}$. A *tweakable enciphering scheme* is a function $\mathcal{E} : \mathcal{K} \times \mathcal{T} \times \mathcal{M} \to \mathcal{M}$ where $\mathcal{K}$ (the key set) is a finite nonempty set and $\mathcal{T}$ (the tweak set) is a nonempty set and $\mathcal{M}$ is a message space and for every $K \in \mathcal{K}$ and $T \in \mathcal{T}$ we have that $\mathcal{E}(K,T,\cdot) = \mathcal{E}_K^T(\cdot)$ is a length-preserving permutation.

When $\mathcal{M}$ is a message space and $\mathcal{T}$ is a nonempty set we let $\mathrm{Perm}(\mathcal{M})$ denote the set of all functions $\pi : \mathcal{M} \to \mathcal{M}$ that are length-preserving permutations, and we let $\mathrm{Perm}^{\mathcal{T}}(M)$ denote the set of functions $\pi : \mathcal{T} \times \mathcal{M} \to \mathcal{M}$ for which $\pi(T,\cdot)$ is a length-preserving permutation for all $T \in \mathcal{T}$.

One can notice the similarity between a TBC and a TES. The main difference being that the message space of a TES can contain arbitrary long strings, thus informally a TES is a TBC on a "bigger" message space.

Let $\mathcal{E} : \mathcal{K} \times \mathcal{T} \times \mathcal{M} \to \mathcal{M}$ be an encryption scheme and let A be an adversary. We define the *advantage of* A in distinguishing a TES $\mathcal{E}$ from a random, tweakable, length-preserving permutation and its inverse as

$$\begin{aligned}\mathbf{Adv}_{\mathcal{E}}^{\pm\widetilde{\mathrm{prp}}}(A) &= \Big|\Pr\left[K \xleftarrow{\$} \mathcal{K} : A^{\mathcal{E}_K(.,.),\mathcal{E}_K^{-1}(.,.)} \Rightarrow 1\right] \\ &\quad - \Pr\left[\boldsymbol{\pi} \xleftarrow{\$} \mathrm{Perm}^{\mathcal{T}}(\mathcal{M}) : A^{\boldsymbol{\pi}(.,.),\boldsymbol{\pi}^{-1}(.,.)} \Rightarrow 1\right]\Big|.\end{aligned}$$

The tilde above the "prp" denotes prp is tweakable (as was also the case in TBC). We say that $\mathcal{E}$ is secure in the sense of a SPRP if the $\mathbf{Adv}_{\mathcal{E}}^{\pm\widetilde{\mathrm{prp}}}(A)$ is "small" for all "efficient" adversaries A.

Pointless queries: In the description of the advantage above we implicitly assume certain query restrictions of the adversary to rule out trivial wins. Let T, P and C represent tweak, plaintext and ciphertext respectively. We assume that an adversary never repeats a query, i.e., it does not ask the encryption oracle with a particular value of (T, P) more than once and neither does it ask the decryption oracle with a particular value of (T, C) more than once. Furthermore, an adversary never queries its deciphering oracle with (T, C) if it got C in response to an encipher query (T, P) for some P. Similarly, the adversary never queries its enciphering oracle with (T, P) if it got P as a response to a decipher query of (T, C) for some C. These queries are called *pointless* as the adversary knows what it would get as responses for such queries.

3.3 History of Tweakable Enciphering Schemes

Naor and Reingold [NR97, NR99] were the first to present a scheme very close to a TES, they provided a design for constructing a SPRP on N bits using a block cipher on $n < N$ bits. The proposed construction uses a paradigm which has been later generalized as the *Hash-Encrypt-Hash* paradigm. In this paradigm one applies to the input an invertible universal hash function, encrypts the result, and then applies another invertible hash function. They did not provide a tweakable SPRP, since their work predates the notion of tweaks, which was proposed later by Liskov, Rivest and Wagner [LRW02].

The first construction of a fully functional and secure tweakable SPRP, known as CMC, was proposed by Halevi and Rogaway [HR03]. In their work they first suggested that TES could be used for the application of low level disk encryption. CMC consists of two layers of CBC mode of encryption along with an intermediate masking layer, hence the name CMC which stands for CBC-Mask-CBC. Because of the nature of the CBC mode of operation, CMC is not parallelizable. A follow-on

work to CMC is the EME algorithm [HR04] proposed by the same authors of CMC. Unlike its predecessor, EME, which stands for ECB-Mix-ECB, is a parallelizable mode. EME has the limitation that it cannot securely encrypt messages in which number of blocks are more than the block length of the underlying block-cipher, for example, if AES is used as the block cipher in EME then it can securely encrypt messages containing less than 128 blocks. This limitation was fixed in EME2 [Hal04]. The modes CMC, EME, EME2 have been classified as Encrypt-Mask-Encrypt type modes, as they use two encryption layers along with a masking layer. For encrypting m blocks of messages, these modes require around $2m$ block cipher calls, the block cipher calls are the most expensive operations used by these modes.

A different paradigm of construction which has been named as the Hash-Counter-Hash type, consist of two universal hash functions with a counter mode of encryption in between. The first known construction of this kind is XCB [MF04]. Other known constructions of this category are HCTR [WFW05], HCH [CS06a], ABL [MV04] and HMCH [Sar09]. HCTR [WFW05] had a serious drawback in the claimed security, as in [WFW05] it was proved that HCTR has a cubic security bound, later in [CN08] a quadratic security bound was proved. In an attempt to fix the security bound of HCTR, HCH [CS06a] was proposed. HCH modifies HCTR in various ways to produce a new mode which uses one more block cipher call than HCTR but provides a quadratic security bound. ABL [MV04] is another construction of the Hash-Counter-Hash type, but it is inefficient compared to the other members of its category. Later in [Sar09] it was proposed that a new kind of polynomials called Bernstein-Rabin-Winograd (BRW) polynomials can be used for the universal hashing. A new mode called HMCH was proposed in [Sar09] which is structurally similar to HCH and HCTR but requires about half the number of multiplications compared to HCH and HCTR. The constructions of this type require finite field multiplications and block cipher calls. The efficient modes like HCTR, XCB and HCH use about m block cipher calls along with $2m$ finite field multiplications for encrypting an m block message.

The Hash-Encrypt-Hash paradigm used by Naor and Reingold [NR97, NR99] has also later been used to construct TES. Examples of such constructions are PEP [CS06b], TET [Hal07] and HEH [Sar07]. Like the Hash-Counter-Hash constructions, these modes also require about m block cipher calls and $2m$ finite field multiplications for encrypting an m block message.

A later modification of HEH [Sar09] using BRW polynomial requires only m multiplications, thus making it the most efficient in the category.

3.4 Description of some TES

In the following subsections we give complete description of three TES. Two of them fall in the category Hash-Counter-Hash, namely XCB and HCTR. The third, EME2, falls in the category Encrypt-Mask-Encrypt.

3.4.1 Extended codebook (XCB)

The Extended Codebook (XCB) mode of operation was proposed by D.A. Mc Grew and S. Fluhrer as a tweakable SPRP which accepts arbitrarily-sized plaintext and arbitrarily-sized tweak. There are two versions of XCB, one proposed in 2004 which we will call as XCBv1 and the other one proposed in 2007 which we will call as XCBv2[1]. In this section we will see first the construction of XCBv1 and then we will proceed with the changes made to get XCBv2. XCBv2 is one of the two algorithms which are part of the IEEE Standard for Wide-Block Encryption for Shared Storage Media (IEEE Std 1619.2-2010).

The construction of XCBv1

XCBv1 [MF04] uses two basic building blocks. An AXU hash function and a counter mode of operation. The hash function is defined as:

$$\begin{aligned} H_h(X,T) =& X_1 h^{m+p+1} \oplus X_2 h^{m+p} \oplus \ldots \oplus \mathsf{pad}(X_m) h^{p+2} \oplus T_1 h^{p+1} \\ & \oplus T_2 h^p \oplus \ldots \oplus \mathsf{pad}\,(T_p)\, h^2 \oplus (\mathsf{bin}_{\frac{n}{2}}(|X|)||\mathsf{bin}_{\frac{n}{2}}(|T|))h, \end{aligned} \tag{3.1}$$

where h is an n-bit hash key and $X = X_1||X_2||\ldots||X_m$, such that $|X_i| = n$ bits $(i = 1, 2, \ldots m-1)$, $0 < |X_m| \leq n$ and $T = T_1||T_2||\ldots||T_p$, such that $|T_j| = n$ bits $(j = 1, 2, \ldots p-1)$, $0 < |T_p| \leq n$. The pad function is defined as $\mathsf{pad}(X_m) := X_m||0^r$ where $r = n - |X_m|$. Thus, $|\mathsf{pad}(X_m)| = n$.

Given an n-bit string S, the counter mode is defined as follows.

$$\mathsf{Ctr}_{K,S}(A_1, \ldots, A_m) \;\; = \;\; (A_1 \oplus E_K(\mathsf{inc}^0(S)), \ldots, A_m \oplus E_K(\mathsf{inc}^{m-1}(S))).$$

In case the last block A_m is incomplete then $A_m \oplus E_K(\mathsf{inc}^{m-1}(S))$ in Ctr is replaced by $A_m \oplus \mathsf{drop}_r(E_K(\mathsf{inc}^{m-1}(S)))$, where $r = n - |A_m|$ and $\mathsf{drop}_r(E_K(\mathsf{inc}^{m-1}(S)))$ is the first $(n-r)$ bits of $E_K(\mathsf{inc}^{m-1}(S))$. In the definition of Ctr, for a bit string $X \in \{0,1\}^n$, $\mathsf{inc}(X)$ treats the least significant 32 bits (the rightmost 32 bits) of X as a non-negative integer, and increments this value modulo 2^{32}, i.e.,

$$\mathsf{inc}(X) \;\; = \mathsf{msb}_{n-32}(X)||\mathsf{bin}_{32}(\mathsf{int}(\mathsf{lsb}_{32}(X)) + 1 \bmod 2^{32}).$$

For $r \geq 0$, we write $\mathsf{inc}^r(X)$ to denote the r times iterative applications of inc on X. We use the convention that $\mathsf{inc}^0(X) = X$.

The encryption and decryption operations using XCBv1 are described in Fig. 3.1, and a high-level description is provided in Fig. 3.2.

There is no known proof of XCBv1. We will for the first time provide a proof of XCBv1 in Chapter 4, in page 31.

[1]These names XCBv1 and XCBv2 were not given by their proposers. We use this nomenclature to separate out the differences.

Algorithm $\mathbf{E}_K^T(P_1, \ldots, P_m)$	**Algorithm** $\mathbf{D}_K^T(C_1, \ldots, C_m)$
1. $h_1 \leftarrow E_K(0^{n-3}\|\|001)$	1. $h_1 \leftarrow E_K(0^{n-3}\|\|001)$
2. $h_2 \leftarrow E_K(0^{n-3}\|\|011)$	2. $h_2 \leftarrow E_K(0^{n-3}\|\|011)$
3. $K_e \leftarrow E_K(0^n)$	3. $K_e \leftarrow E_K(0^n)$
4. $K_d \leftarrow E_K(0^{n-3}\|\|100)$	4. $K_d \leftarrow E_K(0^{n-3}\|\|100)$
5. $K_c \leftarrow E_K(0^{n-3}\|\|010)$	5. $K_c \leftarrow E_K(0^{n-3}\|\|010)$
6. $CC \leftarrow E_{K_e}(P_1)$	6. $MM \leftarrow E_{K_d}(C_1)$
7. $S \leftarrow CC \oplus H_{h_1}(P_2\|\| \ldots \|\|P_m, T)$	7. $S \leftarrow MM \oplus H_{h_2}(C_2\|\| \ldots \|\|C_m, T)$
8. $(C_2, \ldots, C_m) \leftarrow \mathsf{Ctr}_{K_c,S}(P_2, \ldots, P_m)$	8. $(P_2, \ldots, P_m) \leftarrow \mathsf{Ctr}_{K_c,S}(C_2, \ldots, C_m)$
9. $MM \leftarrow S \oplus H_{h_2}(C_2\|\| \ldots \|\|C_m, T)$	9. $CC \leftarrow S \oplus H_{h_1}(P_2\|\| \ldots \|\|P_m, T)$
10. $C_1 \leftarrow E_{K_d}^{-1}(MM)$	10. $P_1 \leftarrow E_{K_e}^{-1}(CC)$
11. return $(C_1, C_2, \ldots, C_m)$	11. return $(P_1, P_2, \ldots, P_m)$

Figure 3.1: Encryption and decryption using XCBv1. K is the block-cipher key and T the tweak.

Specification of XCBv2

As stated earlier XCBv1 does not have a security proof associated with it. The authors proposed some small changes in XCBv1 in [MF07], we call this version as XCBv2.

According to the authors, the changes incorporated in XCBv2 make its security easier to analyze. Firstly, XCBv2 uses only a single hash key, this allows one to see some algebraic relations of the hash function. This change may also benefit software implementations by relieving them of the need to store precomputed tables for an additional hash key. Secondly, the inputs to the hash functions are slightly rearranged, in order to make use of the properties of the hash function. Additionally, the new design reorders the operations in a way that makes XCBv2 more amenable to pipelined implementation, by changing the way the plaintexts are mapped to internal variables.

The AXU hash function used in case of XCBv2 is defined as:

$$\begin{aligned} H_h(T, X) =& T_1 h^{p+m+1} \oplus T_2 h^{p+m} \oplus \ldots \oplus \mathsf{pad}\,(T_p)\, h^{m+2} \oplus X_1 h^{m+1} \\ & \oplus X_2 h^m \oplus \ldots \oplus \mathsf{pad}(X_m) h^2 \oplus (\mathsf{bin}_{\frac{n}{2}}(|T|)||\mathsf{bin}_{\frac{n}{2}}(|X|))h, \end{aligned}$$

as before h is an n-bit hash key and $X = X_1||X_2|| \ldots ||X_m$, such that $|X_i| = n$ bits $(i = 1, 2, \ldots m-1)$, $0 < |X_m| \leq n$ and $T = T_1||T_2|| \ldots ||T_p$, such that $|T_j| = n$ bits $(j = 1, 2, \ldots p-1)$, $0 < |T_p| \leq n$. The pad function is defined as $\mathsf{pad}(X_m) := X_m||0^r$ where $r = n - |X_m|$. Thus, $|\mathsf{pad}(X_m)| = n$. As shown above, the difference between the hash function used in XCBv1 and the hash function used in XCBv2 is the order of the inputs.

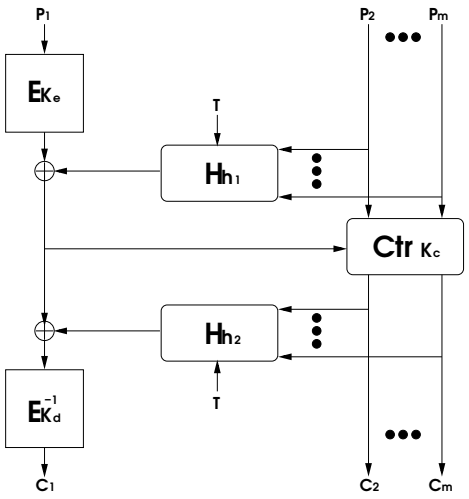

Figure 3.2: Encryption using XCBv1. Here K_c, K_d, K_e are the keys for the block cipher E and h_1, h_2 are the keys for the hash function H.

For the counter mode of operation, XCBv2 uses the same counter mode of XCBv1. The encryption and decryption operations using XCBv2 are described in Fig. 3.3, and a high-level description is provided in Fig. 3.4.

In [MF07] the authors proved a security bound of XCBv2. But we found the bound to be erroneous. We would discuss details of the security of XCBv2 in Chapter 4, in page 31.

3.4.2 HCTR

HCTR is an efficient mode of operation when it is compared with other TES. It is named HCTR because of the use of a special hash function and the counter mode of operation; the algorithm is based in the Hash-Counter-Hash construction. HCTR receives as input an arbitrary variable plaintext and an arbitrary but fixed-length tweak.

Specification of HCTR

The AXU hash used in case of HCTR is defined as:

$$H_h(X) = X_1 h^{m+1} \oplus X_2 h^m \oplus \ldots \oplus \mathsf{pad}_r(X_m) h^2 \oplus \mathsf{bin}_n(|X|) h,$$

where h is an n-bit hash key and $X = X_1 || X_2 || \ldots || X_m$, such that $|X_i| = n$ bits $(i = 1, 2, \ldots m-1)$, $0 < |X_m| \leq n$. The pad function is defined as $\mathsf{pad}_r(X_m) := X_m || 0^r$ where $r = n - |X_m|$. Thus, $|\mathsf{pad}_r(X_m)| = n$. If $X = \lambda$, the empty string, we define $H_h(\lambda) = h$.

HCTR requires a counter mode of operation. Given an n-bit string S, a sequence $S_1, \ldots, S_m$ is defined, where each S_i depends on S. Given such a sequence and a key

Algorithm $\mathbf{E}_K^T(P_1, \ldots, P_m)$

1. $h \leftarrow E_K(0^n)$
2. $K_e \leftarrow \mathsf{msb}_{|K|}(E_K(0^{n-3}||001)||E_K(0^{n-3}||010))$
3. $K_d \leftarrow \mathsf{msb}_{|K|}(E_K(0^{n-3}||011)||E_K(0^{n-3}||100))$
4. $K_c \leftarrow \mathsf{msb}_{|K|}(E_K(0^{n-3}||101)||E_K(0^{n-3}||110))$
5. $(P_{m-1}, P_m) \leftarrow (\mathsf{msb}_{|P_m|}(P_{m-1}), \mathsf{lsb}_n(P_{m-1}||P_m))$
6. $CC \leftarrow E_{K_e}(P_m)$
7. $S \leftarrow CC \oplus H_h(0^n||T, P_1|| \ldots ||\mathsf{pad}(P_{m-1})||0^n)$
8. $(C_1, \ldots, C_{m-1}) \leftarrow \mathsf{Ctr}_{K_c,S}(P_1, \ldots, P_{m-1})$
9. $MM \leftarrow S \oplus H_h(T||0^n, C_1|| \ldots ||\mathsf{pad}(C_{m-1})||(\mathsf{bin}_{\frac{n}{2}}(|T||0^n|)||\mathsf{bin}_{\frac{n}{2}}(|C_1|| \ldots ||C_{m-1}|)))$
10. $C_m \leftarrow E_{K_d}^{-1}(MM)$
11. return $(C_1, C_2, \ldots, C_m)$

Algorithm $\mathbf{D}_K^T(C_1, \ldots, C_m)$

1. $h \leftarrow E_K(0^n)$
2. $K_e \leftarrow \mathsf{msb}_{|K|}(E_K(0^{n-3}||001)||E_K(0^{n-3}||010))$
3. $K_d \leftarrow \mathsf{msb}_{|K|}(E_K(0^{n-3}||011)||E_K(0^{n-3}||100))$
4. $K_c \leftarrow \mathsf{msb}_{|K|}(E_K(0^{n-3}||101)||E_K(0^{n-3}||110))$
5. $(C_{m-1}, C_m) \leftarrow (\mathsf{msb}_{|C_m|}(C_{m-1}), \mathsf{lsb}_n(C_{m-1}||C_m))$
6. $MM \leftarrow E_{K_d}^{-1}(C_m)$
7. $S \leftarrow MM \oplus H_h(T||0^n, C_1|| \ldots ||\mathsf{pad}(C_{m-1})||(\mathsf{bin}_{\frac{n}{2}}(|T||0^n|)||\mathsf{bin}_{\frac{n}{2}}(|C_1|| \ldots ||C_{m-1}|)))$
8. $(P_1, \ldots, P_{m-1}) \leftarrow \mathsf{Ctr}_{K_c,S}(C_1, \ldots, C_{m-1})$
9. $CC \leftarrow S \oplus H_h(0^n||T, P_1|| \ldots ||\mathsf{pad}(P_{m-1})||0^n)$
10. $P_m \leftarrow E_{K_e}^{-1}(CC)$
11. return $(P_1, P_2, \ldots, P_m)$

Figure 3.3: Encryption and decryption using XCBv2. K is the block-cipher key and T the tweak.

K the counter mode is defined as follows.

$$\mathsf{Ctr}_{K,S}(A_1, \ldots, A_m) \;=\; (A_1 \oplus E_K(S_1), \ldots, A_m \oplus E_K(S_m)),$$

where $S_i = S \oplus \mathsf{bin}_n(i)$. In case the last block A_m is incomplete then $A_m \oplus E_K(S_m)$ in Ctr is replaced by $A_m \oplus \mathsf{drop}_r(E_K(S_m))$, where $r = n - |A_m|$ and $\mathsf{drop}_r(E_K(S_m))$ is the first $(n-r)$ bits of $E_K(S_m)$. The encryption and decryption operations using HCTR are described in Fig. 3.5, and a high-level description is provided in Fig. 3.6. If $m = 1$ (when we have one block message), we ignore line 4 in both encryption and decryption algorithm.

Security of HCTR. The authors of HCTR proved a weaker bound in [WFW05]. Later in [CN08] the bound was improved. The following theorem specifies the updated security bound of HCTR.

Theorem 3.4.1 (Security of HCTR [CN08])**.** *Fix n, σ to be positive integers and an n-bit block cipher $E : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$. Then*

$$\mathbf{Adv}^{\pm\widetilde{\mathrm{prp}}}_{\mathrm{HCTR}[\mathrm{Perm}(n)\,]}(\sigma) \leq \frac{4.5\sigma^2}{2^n}.$$

$$\mathbf{Adv}^{\pm\widetilde{\mathrm{prp}}}_{\mathrm{HCTR}[\mathrm{E}\,]}(\sigma, t) \leq \frac{4.5\sigma^2}{2^n} + \mathbf{Adv}^{\pm\mathrm{prp}}_{E}(\sigma, t')$$

where $t' = t + O(\sigma)$.

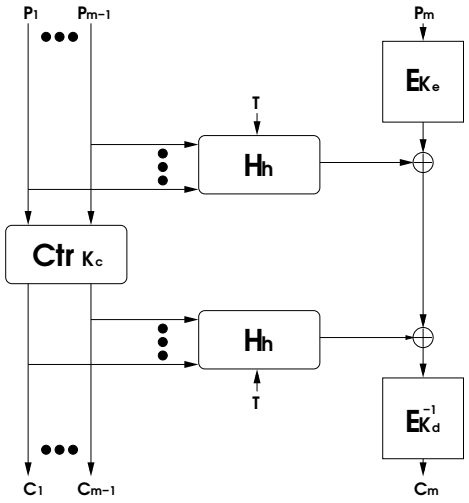

Figure 3.4: Encryption using XCBv2. Here K_c, K_d, K_e are the keys for the block cipher E and h is the key for the hash function H.

3.4.3 EME2

EME2 is the other algorithm which is part of the IEEE Standard for Wide-Block Encryption for Shared Storage Media (IEEE Std 1619.2-2010). EME2 is a modification of the TES EME, which was proposed by Halevi and Rogaway for the disk-sector encryption problem. EME2 solves some limitations founded in its predecessor: the input message had to be a multiple of n, the block-size of the underlying cipher; and the input message was limited to at most n^2 bits.

Specification of EME2

EME2 consists of two layers of masked ECB encryption with a layer called intermediate mixing in between. The complete specification of EME2 is given in Fig. 3.7, and a high-level description is provided in Fig. 3.8.

Security of EME2

Theorem 3.4.2 (Security of EME2 [Hal04])**.** *Fix n, σ to be positive integers and an n-bit block cipher $E : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$. Then*

$$\mathbf{Adv}^{\pm\widetilde{\mathrm{prp}}}_{\mathrm{EME2}[\mathrm{Perm}(n)\,]}(q,\sigma) \leq \frac{(2.5\sigma + 3q)^2}{2^{n+1}}.$$

$$\mathbf{Adv}^{\pm\widetilde{\mathrm{prp}}}_{\mathrm{EME2}[\mathrm{E}\,]}(q,\sigma,t) \leq \frac{(2.5\sigma + 3q)^2}{2^{n+1}} + 2\mathbf{Adv}^{\pm\mathrm{prp}}_{E}(\sigma,t')$$

where $t' = t + O(\sigma)$.

Algorithm $\mathbf{E}^T_{K,h}(P_1, \ldots, P_m)$	**Algorithm** $\mathbf{D}^T_{K,h}(C_1, \ldots, C_m)$
1. $MM \leftarrow P_1 \oplus H_h(P_2 \| \ldots \| P_m \| T)$;	1. $CC \leftarrow C_1 \oplus H_h(C_2 \| C_3 \| \ldots \| C_m \| T)$;
2. $CC \leftarrow E_K(MM)$;	2. $MM \leftarrow E_K^{-1}(CC)$;
3. $S \leftarrow MM \oplus CC$;	3. $S \leftarrow MM \oplus CC$;
4. $(C_2, \ldots, C_{m-1}, C_m) \leftarrow \mathsf{Ctr}_{K,S}(P_2, \ldots, P_m)$;	4. $(P_2, \ldots, P_{m-1}, P_m) \leftarrow \mathsf{Ctr}_{K,S}(C_2, \ldots, C_m)$;
5. $C_1 \leftarrow CC \oplus H_h(C_2 \| C_3 \| \ldots \| C_m \| T)$;	5. $P_1 \leftarrow MM \oplus H_h(P_2 \| \ldots \| P_m \| T)$;
6. return $(C_1, \ldots, C_m)$;	6. return $(P_1, \ldots, P_m)$;

Figure 3.5: Encryption using HCTR. K is the block-cipher key, h the hash key and T the tweak.

3.5 IEEE P1619

The Security in Storage Working Group (SISWG) is responsible for the standardization project IEEE P1619, which includes a family of standards for protection of stored data.

SISWG has four task groups: 1619 which deals with narrow block encryption, 1619.1 which deals with tape encryption, 1619.2 which deals with wide block encryption and 1619.3 which deals with key management.

The task groups 1619 and 1619.2 concentrated their effort on length preserving encryption stressing this to be an important criteria for logical block based devices, such as hard disk. As previously mentioned TES fit into this category.

The task group 1619 has come up with the standard document IEEE Std 1619-2007 for narrow block algorithms. The narrow block algorithms operate on small portions of data and have the advantage of efficient hardware implementation. The standard defines the XTS-AES tweakable block cipher and its use for encryption of sector based storage. XTS-AES is a tweakable block cipher that acts on data units of 128 bits or more and uses the AES block cipher as a subroutine. The XTS-AES can be seen as an electronic code book mode of tweakable block-ciphers where each block uses a different tweak, hence the name "narrow block mode".

The result of the task group 1619.2 was the standard document IEEE Std 1619.2-2010 for wide block algorithms. Wide block algorithms act on the whole logical block at once, and each bit on the input plaintext influences every bit of the output ciphertext. The standard specifies the EME2-AES and the XCB-AES wide block encryption algorithms, which use the AES block cipher as a subroutine.

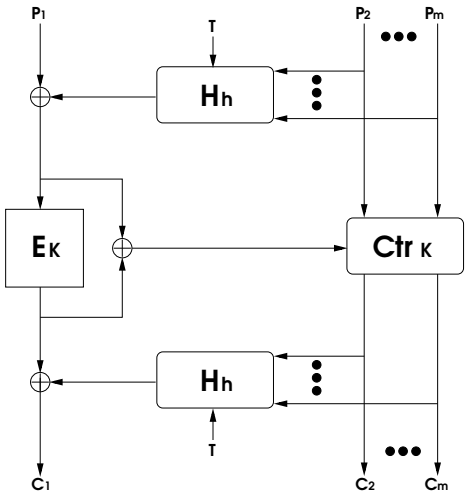

Figure 3.6: Encryption using HCTR. Here K is the key for the block cipher $E_K()$ and h is the key for the universal hash function $H_h()$.

function $H_{K,R}(T_1, \ldots, T_{\ell-1}, T_\ell)$
01. **if** T is empty **return** $E_K(R)$

10. **for** $i = 1$ to $\ell - 1$, $TTT_i \leftarrow E_K(2^i R \oplus T_i) \oplus 2^i R$
11. **if** $|T_\ell| = n$ **then** $TTT_\ell \leftarrow E_K(2^\ell R \oplus T_\ell) \oplus 2^\ell R$
12. **else** $TTT_\ell \leftarrow E_K(2^{\ell+1} R \oplus (T_\ell || 10 \ldots 0)) \oplus 2^{\ell+1} R$
13. **return** $TTT_1 \oplus \cdots \oplus TTT_\ell$

Algorithm $\mathbf{E}^T_{K,L,R}(P_1, \ldots, P_m)$

101. **if** $|P_m| = n$ **then** $lastFull \leftarrow m$
102. **else** $lastFull \leftarrow m - 1$
103. $PPP_m \leftarrow P_m$ padded with $10 \ldots 0$

110. **for** $i = 1$ to $lastFull$,
111. $PP_i \leftarrow 2^{i-1} L \oplus P_i$
112. $PPP_i \leftarrow E_K(PP_i)$

120. $SP \leftarrow PPP_2 \oplus \cdots \oplus PPP_m$
121. $MP_1 \leftarrow PPP_1 \oplus SP \oplus H_{K,R}(T)$
122. **if** $|P_m| = n$ **then** $MC_1 \leftarrow E_K(MP_1)$
123. **else** $MM \leftarrow E_K(MP_1)$
124. $MC_1 \leftarrow E_K(MM)$
125. $C_m \leftarrow P_m \oplus (MM$ truncated$)$
126. $CCC_m \leftarrow C_m$ padded with $10 \ldots 0$
127. $M_1 \leftarrow MP_1 \oplus MC_1$

130. **for** $i = 2$ to $lastFull$,
131. $j = \lceil i/n \rceil, k = (i - 1) \bmod n$
132. **if** $k = 0$ **then**
133. $MP_j \leftarrow PPP_i \oplus M_1$
134. $MC_j \leftarrow E_K(MP_j)$
135. $M_j \leftarrow MP_j \oplus MC_j$
136. $CCC_i \leftarrow MC_j \oplus M_1$
137. **else** $CCC_i \leftarrow PPP_i \oplus 2^k M_j$

140. $SC \leftarrow CCC_2 \oplus \cdots \oplus CCC_m$
141. $CCC_1 \leftarrow MC_1 \oplus SC \oplus H_{K,R}(T)$
142. **for** $i = 1$ to $lastFull$,
143. $CC_i \leftarrow E_K(CCC_i)$
144. $C_i \leftarrow CC_i \oplus 2^{i-1} L$

150. **return** $C_1, \ldots, C_m$

Algorithm $\mathbf{D}^T_{K,L,R}(C_1, \ldots, C_m)$

101. **if** $|C_m| = n$ **then** $lastFull \leftarrow m$
102. **else** $lastFull \leftarrow m - 1$
103. $CCC_m \leftarrow C_m$ padded with $10 \ldots 0$

110. **for** $i = 1$ to $lastFull$,
111. $CC_i \leftarrow 2^{i-1} L \oplus C_i$
112. $CCC_i \leftarrow E_K^{-1}(CC_i)$

120. $SC \leftarrow CCC_2 \oplus \cdots \oplus CCC_m$
121. $MC_1 \leftarrow CCC_1 \oplus SC \oplus H_{K,R}(T)$
122. **if** $|C_m| = n$ **then** $MP_1 \leftarrow E_K^{-1}(MC_1)$
123. **else** $MM \leftarrow E_K^{-1}(MC_1)$
124. $MP_1 \leftarrow E_K^{-1}(MM)$
125. $P_m \leftarrow C_m \oplus (MM$ truncated$)$
126. $PPP_m \leftarrow P_m$ padded with $10 \ldots 0$
127. $M_1 \leftarrow MP_1 \oplus MC_1$

130. **for** $i = 2$ to $lastFull$,
131. $j = \lceil i/n \rceil, k = (i - 1) \bmod n$
132. **if** $k = 0$ **then**
133. $MC_j \leftarrow CCC_i \oplus M_1$
134. $MP_j \leftarrow E_K^{-1}(MC_j)$
135. $M_j \leftarrow MP_j \oplus MC_j$
136. $PPP_i \leftarrow MP_j \oplus M_1$
137. **else** $PPP_i \leftarrow CCC_i \oplus 2^k M_j$

140. $SP \leftarrow PPP_2 \oplus \cdots \oplus PPP_m$
141. $PPP_1 \leftarrow MP_1 \oplus SP \oplus H_{K,R}(T)$
142. **for** $i = 1$ to $lastFull$,
143. $PP_i \leftarrow E_K^{-1}(PPP_i)$
144. $P_i \leftarrow PP_i \oplus 2^{i-1} L$

150. **return** $P_1, \ldots, P_m$

Figure 3.7: Encryption using EME2. K is the block-cipher key, and T is the associated data.

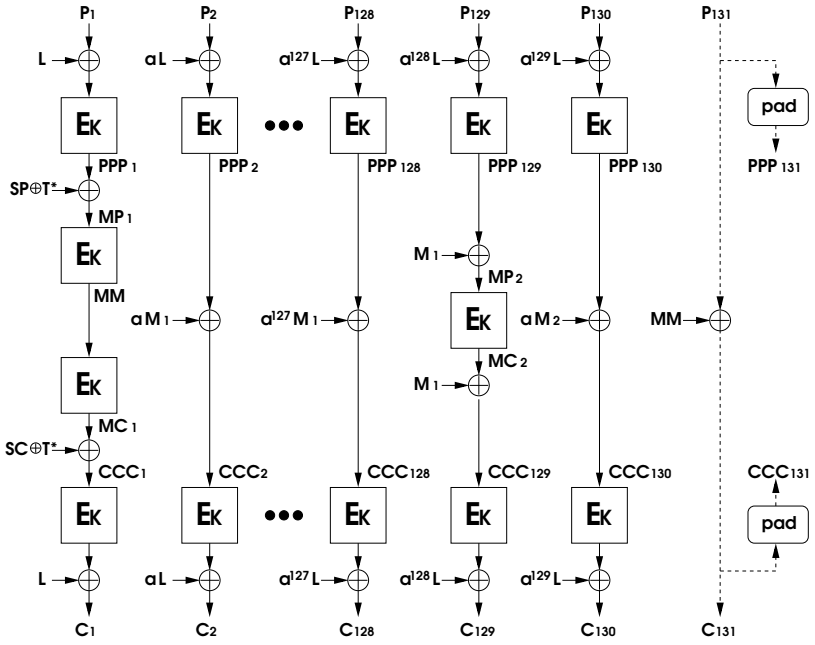

Figure 3.8: Encryption using EME2 with n+2 full blocks and one partial block. Here K is the key for the block cipher $E_K()$ and R in addition with K are the keys for the hash function $H_{K,R}$.

4. On Security of XCB

> About 1,000 instructions is a reasonable upper limit for the complexity of problems now envisioned.
>
> Herman Goldstine and John von Neumann (1946)

McGrew and Fluhrer proposed XCB in [MF04]. However, they did not give a proof of security for their construction. Later in [MF07] they made changes to the original construction proposed in [MF04] and proved security of the updated construction. In [MF07] the authors claim that the changes made to the original construction helps in an easy analysis of it. For ease of reference we shall call the different versions of XCB in [MF04] and [MF07] as XCBv1 and XCBv2 respectively.

In this chapter we do a rigorous analysis of both XCBv1 and XCBv2. As a result of our analysis we conclude that the security claims about XCBv2 made in [MF07] are largely erroneous. Primarily, XCBv2 is not at all secure for messages whose lengths are not the multiples of the block length of the underlying block cipher. We demonstrate this by a simple distinguishing attack on XCBv2. If we restrict the message space to contain only messages whose lengths are multiples of the block length of the block cipher then our distinguishing attack does not work. But in case of such restricted message spaces also the security bound claimed by the authors is not justified. We demonstrate this by pointing out an error in the security proof of XCBv2, the error occurs in a Theorem (Theorem 1 in [MF07]) which is central to the proof of the security theorem stated in [MF07]. We were not able to construct a counterexample for the security theorem itself, but surely the proof provided for the security theorem is incorrect.

We also provide a new security theorem for XCBv2, where the security bound is very different from that claimed in [MF07]. In particular, our theorem shows that XCBv2 is much less secure than that claimed in [MF07].

Additionally in this chapter we prove security for the original XCB (i.e. XCBv1) as proposed in [MF04]. Our security theorem for XCBv1 provides a concrete security bound which was not known before.

Before we continue we want to remark that XCBv2 with AES as the underlying block cipher has been accepted by the IEEE as a standard for wide block encryption

for shared storage media [P1611]. Being a part of the standard it is expected that XCBv2 would be soon deployed (or has been already deployed) as an encryption scheme for a wide range of storage applications. Our results in this chapter shows that XCBv2 has numerous security flaws, thus the users and implementors should be caution regarding its use. Moreover our analysis puts in serious doubt the process and outcome of the IEEE working group on security in storage, the group which proposed this standard.

4.1 Differences of XCBv2 with XCBv1

We have already described XCBv1 and XCBv2 in Chapter 3, but for easy reference we describe the encryption algorithms again in Fig. 4.1. Recall that the hash function H in Fig. 4.1 is defined as

$$\begin{aligned} H_h(X,T) =& X_1 h^{m+p+1} \oplus X_2 h^{m+p} \oplus \ldots \oplus \mathsf{pad}(X_m) h^{p+2} \oplus T_1 h^{p+1} \\ & \oplus T_2 h^p \oplus \ldots \oplus \mathsf{pad}\,(T_p)\, h^2 \oplus (\mathsf{bin}_{\frac{n}{2}}(|X|)||\mathsf{bin}_{\frac{n}{2}}(|T|))h, \end{aligned} \tag{4.1}$$

where h is an n-bit hash key and $X = X_1||X_2||\ldots||X_m$, such that $|X_i| = n$ bits $(i = 1, 2, \ldots m-1)$, $0 < |X_m| \leq n$ and $T = T_1||T_2||\ldots||T_p$, such that $|T_j| = n$ bits $(j = 1, 2, \ldots p-1)$, $0 < |T_p| \leq n$. The pad function is defined as $\mathsf{pad}(X_m) := X_m||0^r$ where $r = n - |X_m|$. Thus, $|\mathsf{pad}(X_m)| = n$.

Next we point out some of the main differences in the two versions.

1. **Only a single hash key is used:** XCBv1 has a different key for each computation of the hash function. In XCBv2 the computation of the hash function is done using the same key. This change enables algebraic relations about the hash function and also benefits software implementations by relieving them of the need to store precomputed tables for an additional hash key[1].

2. **Key sizes:** Both XCBv1 and XCBv2 are parameterized by the key K which is the key of the underlying block cipher. For using XCBv1 it has to be the case that the block length of the block cipher and the key lengths are the same. Notice, that in XCBv1 both K_d and K_c are block cipher outputs and are thus bound to be of the same length of that of the block length of the block cipher. Thus in XCBv1 one cannot use an AES with 192 or 256 bit keys. This limitation was first pointed out in [CS06a]. In XCBv2 this restriction has been removed and it can work for any block cipher whose key length is less or equal to twice its block length.

3. **Inputs to the hash function are rearranged:** The order of the inputs to the hash in the two different constructions differ along with some extra formatting

[1]Note that while arguing about efficiency of XCB the authors stress on a software implementation of the multiplier which uses pre-computed tables, and in such a software implementation only XCB *may have* its efficiency comparable with constructions which only uses block ciphers

of the inputs. In XCBv1 first and second computations of the hash function in lines 007 and 009 are the same, except that in line 007 the input to the hash is the tweak and the plaintext and in line 009 the inputs are the tweak and the ciphertext. In XCBv2 the two computations of the hash functions (in lines 107 and 109) differ in the following ways:

(a) In line 107, the first input to the hash is the tweak with a block of zeros concatenated in the beginning, whereas in line 109 the first input is the tweak with a block of zeros concatenated in the end.

(b) The second input in line 107 is the padded plaintext concatenated with a block of zeros. And in line 109, the second input is the padded ciphertext concatenated with the length of the tweak plus the block length, and the length of the ciphertext.

Note that in case of XCBv2 the second inputs to both the hashes are already padded, and the length parameter is only added in the second hash (line 109).

According to the authors this change in the formatting and order of the hash inputs helps in their analysis, but given that the original definition of the hash (in Eq. (4.1)) already includes a length parameter of the inputs, it is not clear why the authors propose to input the padded messages thus nullifying the effect of the lengths in the first hash and again adding the extra length parameter in case of the second hash computation. We will see later this makes XCBv2 insecure for certain types of messages.

4. **Plaintext is formatted differently:** XCBv1 uses the first n bits of the plaintext to compute the value CC while XCBv2 uses the last n bits of the plaintext to compute the value of the same variable. According to the authors, this change makes the construction amenable to a pipelined implementation.

Changes 1 and 3 give rise to some special properties in the hash function H and the variables CC and MM. These properties are central to the security proof of XCBv2 provided in [MF07]. The properties are as follows:

Theorem 4.1.1 (H is linear)**.** *For any $h \in \{0,1\}^n$ and any T, T', P and P' such that $|T| = |T'|$ and $|P| = |P'|$,*

$$H_h(T,P) \oplus H_h(T',P') = H_h(T \oplus T', P \oplus P') \oplus (\mathsf{bin}_{\frac{n}{2}}(|T|)||\mathsf{bin}_{\frac{n}{2}}(|P|))h.$$

Theorem 4.1.2 (CC and MM have a simple relation)**.**

$$CC = MM \oplus H_h(Z, E_{K_c}(\mathsf{inc}^0(S))|| \ldots ||\mathsf{drop}_r(E_{K_c}(\mathsf{inc}^{m-2}(S))))h$$

where $Z = (T||0^n) \oplus (0^n||T)$.

Proofs of the above theorems are given in Appendix A.

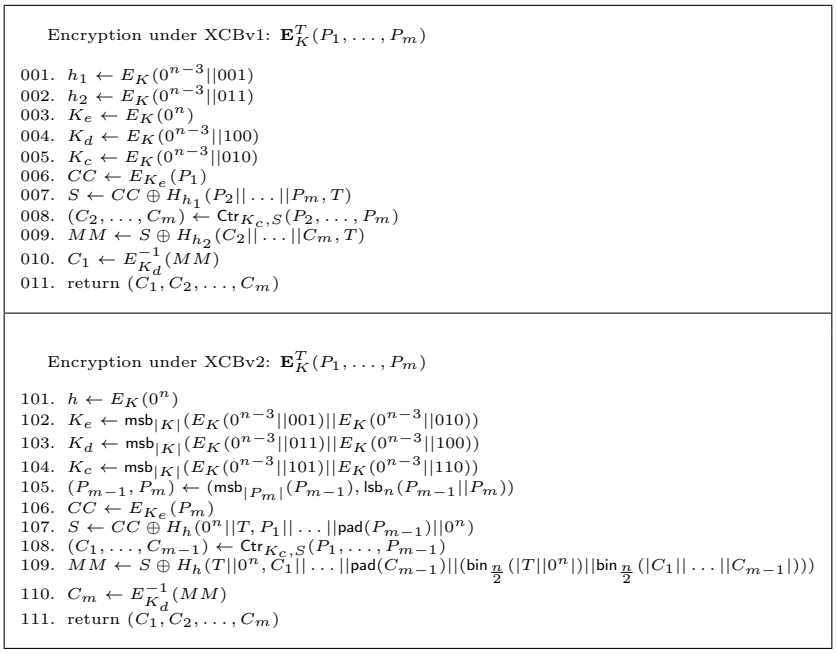

Figure 4.1: Encryption using XCBv1 and XCBv2.

4.2 Security Claims in [MF07]

The security bound for XCBv2 as claimed in [MF07] is summarized in the following theorem.

Theorem 4.2.1. *Fix n, σ to be positive integers and an n-bit block cipher $E : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$. Then*

$$\mathbf{Adv}^{\pm\widetilde{\mathrm{prp}}}_{\mathrm{XCBv2}[E]}(\sigma, t) \leq \frac{q^2\lceil \ell/n + 2\rceil^2 2^3 + 22}{2^n} + \mathbf{Adv}^{\pm\mathrm{prp}}_{E}(\sigma, t')$$

where $t' = t + O(\sigma)$ and ℓ is an upper bound on the total number of bits in the tweak and the plaintext/ciphertext.

The bound stated in the above theorem is based on the following result:

Theorem 4.2.2 (H is unlikely to collide with $\mathsf{inc}^s(H)$)**.** *For any T, T', P, P', D, D' where either $T' \neq T$ or $P' \neq P$ or both inequalities hold, and any index s,*

$$\Pr[h \xleftarrow{\$} \mathcal{K} : H_h(T, P) \oplus D = \mathsf{inc}^s(H_h(T', P') \oplus D')] \leq \frac{\lceil \ell/n + 2\rceil}{2^n},$$

whenever the inputs T and P are restricted to so that the sum of their lengths is ℓ or fewer bits.

In the following two sections we show that both these Theorems are wrong by showing counter examples.

4.3 Distinguishing Attack on XCBv2

Here we give a distinguishing attack on XCBv2 which violates Theorem 4.2.1.

Previously we mentioned that a TES is secure if no adversary can distinguish between the scheme and a random permutation, or the probability that the latter happens is small. A distinguishing attack on a scheme consists of the construction of an adversary which can distinguishing between the scheme and a random permutation with high probability. The existence of such adversary implies the insecurity of the scheme.

The distinguishing attack on XCBv2 that we show here consists of generating the same counter (value of the variable S in the encryption procedure) for two different messages.

We assume an adversary which makes two encryption queries $(T^{(1)}, P^{(1)})$ and $(T^{(2)}, P^{(2)})$, where $T^{(1)} = T^{(2)} = T \in \mathcal{T}$ is an arbitrary tweak and $P^{(1)} = 0^{2n+1}$, $P^{(2)} = 0^{3n}$. As a response to these queries the adversary gets back $C^{(1)}$ and $C^{(2)}$. If the first n-bits of $C^{(1)}$ and $C^{(2)}$ are equal then the adversary concludes that it's oracle is that of XCBv2 and otherwise it concludes that it's oracle is a random permutation.

Now, we explain why this attack works. For the first query $(T^{(1)}, P^{(1)})$, the internal variable $S^{(1)}$ (see line 107) would be,

$$\begin{aligned} S^{(1)} &= E_{K_e}(0^n) \oplus H_h(0^n||T, 0^n||\mathsf{pad}(0)||0^n) \\ &= E_{K_e}(0^n) \oplus H_h(0^n||T, 0^{3n}). \end{aligned}$$

The first block of ciphertext for this query would be

$$C_1^{(1)} = 0^n \oplus E_{K_c}(\mathsf{inc}^0(S^{(1)})). \tag{4.2}$$

For the second query the variable $S^{(2)}$ would be

$$\begin{aligned} S^{(2)} &= E_{K_e}(0^n) \oplus H_h(0^n||T, 0^{2n}||0^n) \\ &= E_{K_e}(0^n) \oplus H_h(0^n||T, 0^{3n}). \end{aligned}$$

Note that for computing $S^{(2)}$ no padding is required, as the plaintext is a multiple of n. Similarly, we have the first block of the ciphertext for the second query as

$$C_1^{(2)} = 0^n \oplus E_{K_c}(\mathsf{inc}^0(S^{(2)})). \tag{4.3}$$

Now, as $S^{(1)} = S^{(2)}$, hence it would always be the case that $C_1^{(1)} = C_1^{(2)}$. Thus the success probability of the adversary is 1 which contradicts the bound given in Theorem 4.2.1.

We can have a general description of this attack. Let us consider two messages $P^{(1)}$ and $P^{(2)}$ as

$$\begin{aligned} P^{(1)} &= P_1||\ldots||P_{m-1}||P_m, \\ P^{(2)} &= P_1||\ldots||P_{m-1}||0^{n-p}||P_m, \end{aligned}$$

where $|P_{m-1}| = p < n$ and $|P_i| = n$ for $1 \leq i \leq m$ and $i \neq m-1$. Note that the length of $P^{(2)}$, is a multiple of the block length n, which is not the case for $P^{(1)}$. Any two messages of this form will result in the same value of S, and hence the first $(m-2)$ blocks of ciphertext produced by this pair of different messages with the same tweak would be equal.

We can vary the size of P_{m-1} and obtain many different messages which will give the same value S. In [P1611], where XCBv2 has been standardized, the length in bits of the message is restricted to be a multiple of 8 bits, it is easy to see that with this restriction also our distinguishing attack works.

4.3.1 Some comments about the attack

1. The attack works because of the way the padding is applied. As the hash function takes as input the zero padded message, hence the length of the original message has no effect on the value of the first hash.

2. The attack does no depend on the padding function. If any other deterministic padding function is applied (say instead of padding zeros, one pads one) this attack can be easily modified to make it work.

3. It seems that the specific way in which the padding is applied has something to do with the proof of the scheme. We show in Appendix A that if the padding is applied in a different way then Theorem 4.1.2 does not hold.

4. An easy way to bypass this attack is to restrict the message space to contain only messages whose lengths are multiples of the block length. In such messages explicit padding would be not required and hence at least this attack would not work in case of XCBv2.

4.4 Collisions in the Increment Function

In the previous section we showed a distinguishing attack which shows that the main security Theorem (Theorem 4.2.1) of XCBv2 is false. We also mentioned that if we restrict ourselves to only messages whose lengths are multiples of the block length of the block cipher then our distinguishing attack does not work. Here we show that even if we restrict ourselves to messages whose length are multiples of the block length, then too the proof of Theorem 4.2.1 as provided in [MF07] is not correct.

The proof of Theorem 4.2.1 as provided in [MF07] depends on the result stated in Theorem 4.2.2. We show that this Theorem is also false by giving some counterexamples.

The result in Theorem 4.2.2 has also been used to argue about the security of an authenticated encryption scheme GCM. Recently, Iwata, Ohashi and Minematsu [IOM12] pointed out an error in the security bound of GCM, this error also

stems from the falsity of Theorem 4.2.2. The counter example that we present here is highly motivated by [IOM12], but our examples are different.

For the discussion that follows, we shall treat an n-bit string $a = a_{n-1}a_{n-2}\ldots a_1a_0$ as a polynomial $a(x) = a_{n-1} \oplus a_{n-2}x \oplus \cdots \oplus a_1x^{n-2} \oplus a_0x^{n-1}$.

Let us consider $D = D' = T = T' = 0^{128}$, $P = 0^{384}$ and $P' = 0^{640}$. Then, according to Theorem 4.2.2

$$p_s = \Pr[h \xleftarrow{\$} \{0,1\}^{128} : H_h(T,P) = \mathsf{inc}^s(H_h(T',P'))] \leq \frac{8}{2^{128}},$$

for any index s. We show that $p_1 \geq 16/2^{128}$, $p_2 \geq 16/2^{128}$ and $p_4 \geq 15/2^{128}$, thus invalidating the theorem.

For our choice of P, P', T, T' and the description of the hash function in equation (4.1) we have $H_h(T,P) = L_1h$ and $H_h(T',P') = L_2h$, where $L_1 = x^{56} + x^{119} + x^{120}$ and $L_2 = x^{56} + x^{118} + x^{120}$. We were able to find 16 distinct values of h which satisfies

$$L_1h \oplus \mathsf{inc}(L_2h) = 0^{128}. \tag{4.4}$$

Also we found 16 distinct values of h satisfying $L_1h \oplus \mathsf{inc}^2(L_2h) = 0^{128}$, and 15 distinct values of h satisfying $L_1h \oplus \mathsf{inc}^4(L_2h) = 0^{128}$. These values are listed in Tables 4.1, 4.2 and 4.3. This suggests that $p_1 \geq 16/2^{128}$, $p_2 \geq 16/2^{128}$ and $p_4 \geq 15/2^{128}$.

0xBE7FFFFFFFFFFFFFFFFFFFFFFFFFFFFF	0xBF7FFFFFFFFFFFFFFFFFFFFFFFFFFFFF
0xBB7FFFFFFFFFFFFFFFFFFFFFFFFFFFFF	0xAB7FFFFFFFFFFFFFFFFFFFFFFFFFFFFF
0xEB7FFFFFFFFFFFFFFFFFFFFFFFFFFFFF	0x297FFFFFFFFFFFFFFFFFFFFFFFFFFFFE
0xA57FFFFFFFFFFFFFFFFFFFFFFFFFFFF8	0xD37FFFFFFFFFFFFFFFFFFFFFFFFFFFE3
0xC97FFFFFFFFFFFFFFFFFFFFFFFFFFF8E	0xA17FFFFFFFFFFFFFFFFFFFFFFFFFFE3A
0xC37FFFFFFFFFFFFFFFFFFFFFFFFF8EB	0x897FFFFFFFFFFFFFFFFFFFFFFFFE3AE
0x637FFFFFFFFFFFFFFFFFFFFFFF8EBB	0x4F7FFFFFFFFFFFFFFFFFFFFFFE3AED
0xFF7FFFFFFFFFFFFFFFFFFFFFF8EBB5	0x797FFFFFFFFFFFFFFFFFFFFFE3AED6

Table 4.1: List of solutions for $L_1h \oplus \mathsf{inc}(L_2h) = 0^{128}$.

0xBEFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF	0xBCFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
0xB4FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF	0x94FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
0x14FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF	0x52FFFFFFFFFFFFFFFFFFFFFFFFFFFFFC
0x88FFFFFFFFFFFFFFFFFFFFFFFFFFFFF1	0x64FFFFFFFFFFFFFFFFFFFFFFFFFFFFC7
0x50FFFFFFFFFFFFFFFFFFFFFFFFFFFF1D	0x80FFFFFFFFFFFFFFFFFFFFFFFFFFFC75
0x44FFFFFFFFFFFFFFFFFFFFFFFFFFF1D7	0xD0FFFFFFFFFFFFFFFFFFFFFFFFFFC75D
0xC6FFFFFFFFFFFFFFFFFFFFFFFFFF1D76	0x9EFFFFFFFFFFFFFFFFFFFFFFFFFC75DA
0x3CFFFFFFFFFFFFFFFFFFFFFFFFFF1D76B	0xF2FFFFFFFFFFFFFFFFFFFFFFFFFC75DAC

Table 4.2: List of solutions for $L_1h \oplus \mathsf{inc}^2(L_2h) = 0^{128}$.

Next, we describe the method we used to find the solutions for equation (4.4). Similar methods can be used to find solutions for $L_1h \oplus \mathsf{inc}^2(L_2h) = 0^{128}$ and $L_1h \oplus \mathsf{inc}^4(L_2h) = 0^{128}$.

Our method is based on the following simple observation.

Definition 4.4.1. *For $a \in \{0,1\}^{128}$, i.e, $a(x) = a_{127} \oplus a_{126}x \oplus a_{125}x^2 \oplus \cdots a_1x^{126} \oplus a_0x^{127}$, where each $a_j \in \{0,1\}$. Define, $\mathsf{lbit}(a) = 127 - i$, where i is the smallest integer such that $0 \leq i \leq 127$, and $a_i = 0$. If, such a i does not exist (such a i will not exist if all bits of a are 1) or $i > 31$, then we fix $\mathsf{lbit}(a) = 96$.*

0xBFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF	0xBBFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
0xABFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF	0xEBFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
0x29FFFFFFFFFFFFFFFFFFFFFFFFFFFFFE	0xA5FFFFFFFFFFFFFFFFFFFFFFFFFFFFF8
0xD3FFFFFFFFFFFFFFFFFFFFFFFFFFFFE3	0xC9FFFFFFFFFFFFFFFFFFFFFFFFFFFF8E
0xA1FFFFFFFFFFFFFFFFFFFFFFFFFFFE3A	0xC3FFFFFFFFFFFFFFFFFFFFFFFFFFF8EB
0x89FFFFFFFFFFFFFFFFFFFFFFFFFFE3AE	0x63FFFFFFFFFFFFFFFFFFFFFFFFFF8EBB
0x4FFFFFFFFFFFFFFFFFFFFFFFFFFE3AED	0xFFFFFFFFFFFFFFFFFFFFFFFFFFF8EBB5
0x79FFFFFFFFFFFFFFFFFFFFFFFFE3AED6	

Table 4.3: List of solutions for $L_1 h \oplus \mathsf{inc}^4(L_2 h) = 0^{128}$.

Proposition 4.4.1. *Fix $a \in \{0,1\}^{128}$, if $\mathsf{lbit}(a) = j$, then $\mathsf{inc}(a) = a(x) \oplus x^j \oplus x^{j+1} \oplus \cdots \oplus x^{127}$.*

If we assume that a solution h of Eq. (4.4) is such that $\mathsf{lbit}(L_2 h) = j$, then by Proposition 4.4.1 we have

$$h = \frac{x^j \oplus x^{j+1} \oplus \cdots \oplus x^{127}}{L_1 \oplus L_2}. \tag{4.5}$$

Using $96 \leq j \leq 127$, in Eq. (4.5) we obtain 32 values of h. We substitute these values in Eq. (4.4) to check whether the value obtained is really a solution. It turned out that sixteen of the 32 values obtained satisfied Eq. (4.4), and these values are reported in Table 4.1. It is easy to characterize $\mathsf{inc}^2(a)$ and $\mathsf{inc}^4(a)$ along the lines of Proposition 4.4.1, and using these characterizations we generate the data for Tables 4.2 and 4.3.

4.5 The Security of XCBv2

In this section we give an updated theorem regarding the security of XCB, and its proof.

The following theorem specifies the security of XCBv2.

Theorem 4.5.1. *Fix n, σ to be positive integers and an n-bit block cipher $E : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$. Then when removing the padding function from XCBv2 i.e., the input message is a multiple of n,*

$$\mathbf{Adv}^{\pm\widetilde{\mathrm{prp}}}_{\mathrm{XCBv2}[\mathrm{Perm}(n),\mathrm{Perm}(n),\mathrm{Perm}(n)]}(\sigma) \leq (2^{22} q^2 \ell_{\max}^2 + 2\sigma^2)/2^{n-1}. \tag{4.6}$$

$$\mathbf{Adv}^{\pm\widetilde{\mathrm{prp}}}_{\mathrm{XCBv2}[E]}(\sigma, t) \leq (2^{22} q^2 \ell_{\max}^2 + 2\sigma^2)/2^{n-1} + \mathbf{Adv}^{\pm\mathrm{prp}}_{E}(\sigma, t') \tag{4.7}$$

where $t' = t + O(\sigma)$ and $\ell_{\max}$ is the maximum query length.

Eq. (4.6) represents the information theoretic bound, i.e., here we are considering that the three main block-ciphers (in lines 106, 108 and 110 of Fig. 4.1) are replaced by random permutations. Thus, the bound in Eq. (4.6) represents an ideal scenario. Eq. (4.7) represents the complexity theoretic bound, thus here the advantage of an adversary includes the distinguishing advantage of the block cipher itself. As it has

been noted before (for example in [HR03], [CN08]), passing from Eq. (4.6) to Eq. (4.7) is quite standard and so we will not provide details of the proof of Eq. (4.7) here. We provide the detailed proof of Eq. (4.7) in the next section. Our proof does not essentially contain any new idea. The structure of the proof closely follows [CN08], and it uses a combinatorial result previously stated in [IOM12].

Note that Theorem 4.5.1 is different from the original claimed security in [MF07] in two aspects. Firstly, Theorem 4.5.1 considers a restricted message space, and secondly, the bound in Theorem 4.5.1 is quite different.

4.6 Proof of Theorem 4.5.1

In this section, we derive the information theoretic security bound for XCBv2. For deriving this bound we use the assumption that the underlying block cipher E is a secure pseudorandom permutation. We replace the block cipher calls $E_{K_c}()$, $E_{K_d}()$ and $E_{K_e}()$ by three permutations π_1, π_2 and π_3 chosen uniformly at random from $\mathrm{Perm}(n)$. We also choose the hash key h uniformly at random from $\{0,1\}^n$.

The notation XCBv2$[E]$ denotes a tweakable enciphering scheme, where the n-bit block cipher E is used in the manner specified by XCBv2. We will use the notation $\mathbf{E}_{\pi_1,\pi_2,\pi_3}$ as a shorthand for XCBv2[Perm(n),Perm(n),Perm(n)] and $\mathbf{D}_{\pi_1,\pi_2,\pi_3}$ will denote the inverse of $\mathbf{E}_{\pi_1,\pi_2,\pi_3}$. Thus, the notation $A^{\mathbf{E}_{\pi_1,\pi_2,\pi_3},\mathbf{D}_{\pi_1,\pi_2,\pi_3}}$ will denote an adversary interacting with the oracles $\mathbf{E}_{\pi_1,\pi_2,\pi_3}$ and $\mathbf{D}_{\pi_1,\pi_2,\pi_3}$.

For proving Eq. (4.6), we need to consider an adversary's advantage in distinguishing a tweakable enciphering scheme $\mathbf{E}$ from an oracle which simply returns random bit strings. This advantage is defined in the following manner.

$$\begin{aligned}\mathbf{Adv}^{\pm\mathrm{rnd}}_{\mathrm{XCBv2[Perm}(n),\mathrm{Perm}(n),\mathrm{Perm}(n)]}(A) = \\ \Big|\Pr\Big[\pi_1,\pi_2,\pi_3 \xleftarrow{\$} \mathrm{Perm}(n): \quad & A^{\mathbf{E}_{\pi_1,\pi_2,\pi_3},\mathbf{D}_{\pi_1,\pi_2,\pi_3}} \Rightarrow 1\Big] \\ & - \Pr\left[A^{\$(.,.),\$(.,.)} \Rightarrow 1\right]\Big|, \qquad (4.8)\end{aligned}$$

where $\$(.,M)$ and $\$(.,C)$ returns independently distributed random bits of length $|M|$ and $|C|$ respectively.

The basic idea of proving Eq. (4.6) is as follows.

$$\begin{aligned}
\mathbf{Adv}^{\pm\widetilde{\mathrm{prp}}}_{\mathrm{XCBv2[Perm}(n),\mathrm{Perm}(n),\mathrm{Perm}(n)]}(A) \\
&= \Big(\Pr\Big[\pi_1,\pi_2,\pi_3 \xleftarrow{\$} \mathrm{Perm}(n) : A^{\mathbf{E}_{\pi_1,\pi_2,\pi_3},\mathbf{D}_{\pi_1,\pi_2,\pi_3}} \Rightarrow 1\Big] \\
&\quad - \Pr\Big[\boldsymbol{\pi} \xleftarrow{\$} \mathrm{Perm}^{\mathcal{T}}(\mathcal{M}) : A^{\boldsymbol{\pi}(.,.),\boldsymbol{\pi}^{-1}(.,.)} \Rightarrow 1\Big]\Big) \\
&= \Big(\Pr\Big[\pi_1,\pi_2,\pi_3 \xleftarrow{\$} \mathrm{Perm}(n) : A^{\mathbf{E}_{\pi_1,\pi_2,\pi_3},\mathbf{D}_{\pi_1,\pi_2,\pi_3}} \Rightarrow 1\Big] \\
&\quad - \Pr\left[A^{\$(.,.),\$(.,.)} \Rightarrow 1\right]\Big) \\
&\quad + \Big(\Pr\left[A^{\$(.,.),\$(.,.)} \Rightarrow 1\right] - \Pr\Big[\boldsymbol{\pi} \xleftarrow{\$} \mathrm{Perm}^{\mathcal{T}}(\mathcal{M}) : A^{\boldsymbol{\pi}(.,.),\boldsymbol{\pi}^{-1}(.,.)} \Rightarrow 1\Big]\Big) \\
&\leq \mathbf{Adv}^{\pm\mathrm{rnd}}_{\mathrm{XCBv2[Perm}(n),\mathrm{Perm}(n),\mathrm{Perm}(n)]}(A) + \binom{q}{2}\frac{1}{2^n}, \qquad (4.9)
\end{aligned}$$

where q is the number of queries made by the adversary. For a proof of the last inequality see [HR03]. Thus, the main task of the proof now reduces to obtaining an upper bound on $\mathbf{Adv}^{\pm\mathrm{rnd}}_{\mathrm{XCBv2[Perm}(n),\mathrm{Perm}(n),\mathrm{Perm}(n)]}(\sigma)$. In Section 4.6.1 we prove that

$$\mathbf{Adv}^{\pm\mathrm{rnd}}_{\mathrm{XCBv2[Perm}(n),\mathrm{Perm}(n),\mathrm{Perm}(n)]}(\sigma) \;\leq\; (2^{22}q^2\ell^2_{\max} + 1.5\sigma^2)/2^{n-1}. \quad (4.10)$$

Using Eq. (4.10) and Eq. (4.9) we obtain Eq. (4.6).
We will devote the next two sections in proving the bound in Eq. (4.10).

4.6.1 The game sequence

We shall model the interaction of the adversary with XCBv2 by a sequence of games. We shall start with the game XCB1 which describes the mode XCBv2, and with small changes we shall reach the game RAND2 which will represent an oracle which returns just random strings and we shall bound the advantage of an adversary in distinguishing between the games XCB1 and RAND1. Note that in the games we remove the pad function because we are assuming the length of the input message is multiple of the length of the block. Next we describe the games.

Game XCB1: In XCB1, the adversary interacts with $\mathbf{E}_{\pi_1,\pi_2,\pi_3}$ and $\mathbf{D}_{\pi_1,\pi_2,\pi_3}$ where each π_i $(i = 1, 2, 3)$ is a randomly chosen permutation from $\mathrm{Perm}(n)$. Instead of initially choosing π_i, we build up π_i in the following manner.

Initially π_i is assumed to be undefined everywhere. When $\pi_i(X)$ is needed, but the value of π_i is not yet defined at X, then a random value is chosen among the available range values. Similarly when $\pi_i^{-1}(Y)$ is required and there is no X yet defined for which $\pi_i(X) = Y$, we choose a random value for $\pi_i^{-1}(Y)$ from the available domain values.

The domain and range of π_i are maintained in two sets $Domain_i$ and $Range_i$, and $\overline{Domain_i}$ and $\overline{Range_i}$ are the complements of $Domain_i$ and $Range_i$ relative to $\{0,1\}^n$. The game XCB1 is shown in Figure 4.2. The figure shows the subroutines Ch-π_i, Ch-π_i^{-1}, the initialization steps and how the game responds to a encipher/decipher query of the adversary. The j^{th} query of the adversary depends on its previous queries, the responses to those queries and on some coins of the adversary.

The game XCB1 accurately represents the attack scenario, and by our choice of notation, we can write

$$\Pr[A^{\mathbf{E}_{\pi_1,\pi_2,\pi_3},\mathbf{D}_{\pi_1,\pi_2,\pi_3}} \Rightarrow 1] = \Pr[A^{\text{XCB1}} \Rightarrow 1]. \tag{4.11}$$

Game RAND1: We modify XCB1 by deleting the boxed entries in XCB1 and call the modified game as RAND1. By deleting the boxed entries it cannot be guaranteed that π_i $(i = 1, 2, 3)$ are permutations as though we do the consistency checks but we do not reset the values of Y (in Ch-π_i) and X (in Ch-π_i^{-1}). Thus, the games XCB1 and RAND1 are identical apart from what happens when the bad flag is set. By the fundamental lemma of game-playing technique (Lemma 2.2.1), we obtain

$$|\Pr[A^{\text{XCB1}} \Rightarrow 1] - \Pr[A^{\text{RAND1}} \Rightarrow 1]| \leq \Pr[A^{\text{RAND1}} \text{ sets } \mathsf{bad}]. \tag{4.12}$$

Another important thing to note is that in RAND1 in line 104, for an encryption query CC^s (and MM^s for a decryption query) gets set to a random n bit string. Similarly in line 108, Z_i^s gets set to random values, and in line 112 for an encryption query $C^s_{m^s}$ (and $P^s_{m^s}$ for a decryption query) gets set to a random n bit string. Thus the the adversary gets random strings in response to both his encryption and decryption queries. Hence,

$$\Pr[A^{\text{RAND1}} \Rightarrow 1] = \Pr[A^{\$(.,.),\$(.,.)} \Rightarrow 1]. \tag{4.13}$$

So using Eq. (4.8), Eq. (4.12) and Eq. (4.13) we get

$$\begin{aligned}
\mathbf{Adv}^{\pm\text{rnd}}_{\text{XCBv2}[\text{Perm}(n),\text{Perm}(n),\text{Perm}(n)]}(A) &= |\Pr[A^{\mathbf{E}_{\pi_1,\pi_2,\pi_3},\mathbf{D}_{\pi_1,\pi_2,\pi_3}} \Rightarrow 1] \\
&\quad - \Pr[A^{\$(.,.),\$(.,.)} \Rightarrow 1]| \qquad (4.14)\\
&= |\Pr[A^{\text{XCB1}} \Rightarrow 1] \\
&\quad - \Pr[A^{\text{RAND1}} \Rightarrow 1]| \\
&\leq \Pr[A^{\text{RAND1}} \text{ sets } \mathsf{bad}]. \qquad (4.15)
\end{aligned}$$

Game RAND2: Now we make some subtle changes in the game RAND1 to get a new game RAND2 which is described in Figure 4.3. In game RAND1 the permutation was not maintained and a call to the permutation was responded by returning random strings, so in Game RAND2 we no more use the subroutines Ch-π_i and Ch-π_i^{-1}. Here we immediately return random strings to the adversary in response to his encryption

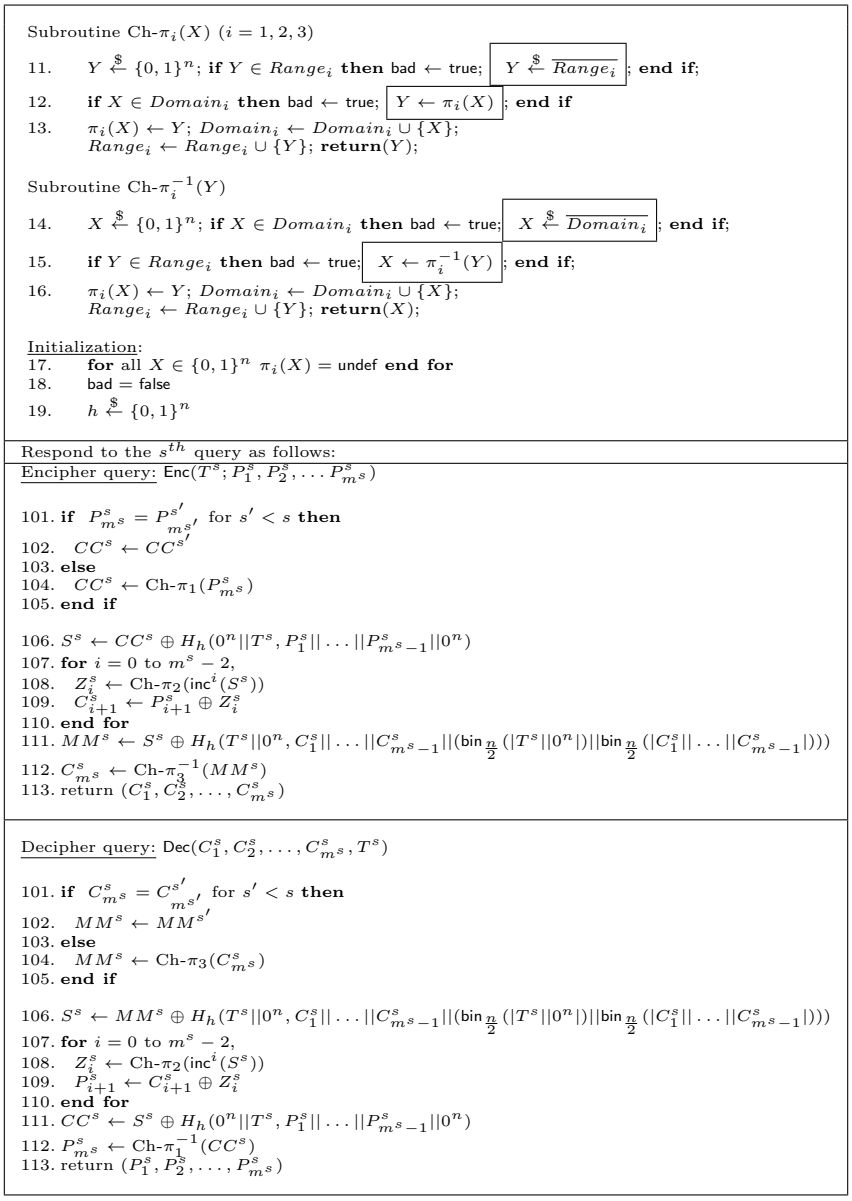

Figure 4.2: Games XCB1 and RAND1: In RAND1 the boxed entries are removed.

Respond to the s^{th} adversary query as follows:

ENCIPHER QUERY $\mathsf{Enc}(T^s; P_1^s, P_2^s, \ldots, P_{m^s}^s)$

11. $ty^s = \mathsf{Enc}$
12. $C_1^s||C_2^s||\ldots||C_{m^s-1}^s||C_{m^s}^s \xleftarrow{\$} \{0,1\}^{nm^s}$
13. **return** $C_1^s||C_2^s||\ldots||C_{m^s}^s$

DECIPHER QUERY $\mathsf{Dec}(T^s; C_1^s, C_2^s, \ldots, C_{m^s}^s)$

21. $ty^s = \mathsf{Dec}$
22. $P_1^s||P_2^s||\ldots||P_{m_s-1}^s||P_{m^s}^s \xleftarrow{\$} \{0,1\}^{nm^s}$
23. **return** $P_1^s||P_2^s||\ldots||P_{m^s}^s$

Finalization:

001. $h \xleftarrow{\$} \{0,1\}^n$
for $s = 1$ to q,
if $ty^s = \mathsf{Enc}$ **then**
101. **if** $P_{m^s}^s = P_{m^{s'}}^{s'}$ for $s' < s$ **then**
102. $CC^s \leftarrow CC^{s'}$
103. **else**
104. $CC^s \xleftarrow{\$} \{0,1\}^n$
105. $\mathcal{D}_1 \leftarrow \mathcal{D}_1 \cup \{P_{m^s}^s\}$
106. $\mathcal{R}_1 \leftarrow \mathcal{R}_1 \cup \{CC^s\}$
107. **end if**
108. $S^s \leftarrow CC^s \oplus H_h(0^n||T^s, P_1^s||\ldots||P_{m^s-1}^s||0^n)$
109. $MM^s \leftarrow S^s \oplus H_h(T^s||0^n, C_1^s||\ldots||C_{m^s-1}^s||(\mathsf{bin}_{\frac{n}{2}}(|T^s||0^n|)||\mathsf{bin}_{\frac{n}{2}}(|C_1^s||\ldots||C_{m^s-1}^s|)))$
110. $\mathcal{D}_3 \leftarrow \mathcal{D}_3 \cup \{C_{m^s}^s\}$
111. $\mathcal{R}_3 \leftarrow \mathcal{R}_3 \cup \{MM^s\}$
112. **for** $i = 0$ to $m^s - 2$,
113. $Y_i^s \leftarrow C_{i+1}^s \oplus P_{i+1}^s$
114. $\mathcal{D}_2 \leftarrow \mathcal{D}_2 \cup \{\mathsf{inc}^i(S^s)\}$
115. $\mathcal{R}_2 \leftarrow \mathcal{R}_2 \cup \{Y_i^s\}$
116. **end for**
else if $ty^s = \mathsf{Dec}$ **then**
201. **if** $C_{m^s}^s = C_{m^{s'}}^{s'}$ for $s' < s$ **then**
202. $MM^s \leftarrow MM^{s'}$
203. **else**
204. $MM^s \xleftarrow{\$} \{0,1\}^n$
205. $\mathcal{D}_3 \leftarrow \mathcal{D}_3 \cup \{C_{m^s}^s\}$
206. $\mathcal{R}_3 \leftarrow \mathcal{R}_3 \cup \{MM^s\}$
207. **end if**
208. $S^s \leftarrow MM^s \oplus H_h(T^s||0^n, C_1^s||\ldots||C_{m^s-1}^s||(\mathsf{bin}_{\frac{n}{2}}(|T^s||0^n|)||\mathsf{bin}_{\frac{n}{2}}(|C_1^s||\ldots||C_{m^s-1}^s|)))$
209. $CC^s \leftarrow S^s \oplus H_h(0^n||T^s, P_1^s||\ldots||P_{m^s-1}^s||0^n)$
210. $\mathcal{D}_1 \leftarrow \mathcal{D}_1 \cup \{P_{m^s}^s\}$
211. $\mathcal{R}_1 \leftarrow \mathcal{R}_1 \cup \{CC^s\}$
212. **for** $i = 0$ to $m^s - 2$,
213. $Y_i^s \leftarrow C_{i+1}^s \oplus P_{i+1}^s$
214. $\mathcal{D}_2 \leftarrow \mathcal{D}_2 \cup \{\mathsf{inc}^i(S^s)\}$
215. $\mathcal{R}_2 \leftarrow \mathcal{R}_2 \cup \{Y_i^s\}$
216. **end for**
end if
end for

SECOND PHASE
bad = false;
if (some value occurs more than once in $\mathcal{D}_i, i = 1, 2, 3$) **then** bad = true **end if**;
if (some value occurs more than once in $\mathcal{R}_i, i = 1, 2, 3$) **then** bad = true **end if**.

Figure 4.3: Game RAND2.

or decryption queries. Later in the finalization step we adjust variables and maintain multi sets $\mathcal{D}_i$ and $\mathcal{R}_i$ where we list the elements that were supposed to be inputs and outputs of the permutation. In the second phase of the finalization step, we check for collisions in the multi sets $\mathcal{D}_i$ and $\mathcal{R}_i$, and in the event of a collision we set the bad flag to true.

Game RAND1 and Game RAND2 are indistinguishable to the adversary, as in both cases he gets random strings in response to his queries. Also, the probability with which RAND1 sets bad is same as the probability with which RAND2 sets bad. Thus we get:

$$\Pr[A^{\text{RAND1}} \text{ sets } \mathsf{bad}] = \Pr[A^{\text{RAND2}} \text{ sets } \mathsf{bad}]. \tag{4.16}$$

Thus from Eq. (4.15) and Eq. (4.16) we obtain

$$\mathbf{Adv}^{\pm\text{rnd}}_{\text{XCBv2}[\text{Perm}(n),\text{Perm}(n),\text{Perm}(n)]}(A) \leq \Pr[A^{\text{RAND2}} \text{ sets } \mathsf{bad}]. \tag{4.17}$$

Now our goal would be to bound $\Pr[A^{\text{RAND2}} \text{ sets } \mathsf{bad}]$. We notice that in Game RAND2 the bad flag is set when there is a collision in either of the sets $\mathcal{D}_i$ or $\mathcal{R}_i$. So if COLLD_i and COLLR_i represent the event of a collision in $\mathcal{D}_i$ and $\mathcal{R}_i$ respectively then

$$\Pr[A^{\text{RAND2}} \text{ sets } \mathsf{bad}] \leq \sum_{i=1}^{3} \left(\Pr[\mathsf{COLLR}_i] + \Pr[\mathsf{COLLD}_i]\right).$$

We devote the next section in bounding the collision probabilities in $\mathcal{D}_i$ and $\mathcal{R}_i$.

4.6.2 Bounding collision probability in $\mathcal{D}_i$ and $\mathcal{R}_i$

In the analysis we consider the sets $\mathcal{D}_i$ and $\mathcal{R}_i$ to consist of the formal variables instead of their values. For example, whenever we set $\mathcal{D} \leftarrow \mathcal{D} \cup \{X\}$ for some variable X we think of it as setting $\mathcal{D} \leftarrow \mathcal{D} \cup \{\text{“}X\text{”}\}$ where “X” is the name of that formal variable. This is the same technique as used in [HR03]. Our goal is to bound the probability that two formal variables in the sets $\mathcal{D}$ and $\mathcal{R}$ take the same value. After q queries of the adversary where the s^{th} query has m^s blocks of plaintext or ciphertext and t^s blocks of tweak, then the sets $\mathcal{D}_i$ and $\mathcal{R}_i$ can be written as follows:

$\mathcal{D}_1 = \{P^s_{m^s} : 1 \leq s \leq q\}$,
$\mathcal{D}_2 = \bigcup_{s=1}^{q} \{\mathsf{inc}^j(S^s) : 0 \leq j \leq m^s - 2\}$,
$\mathcal{D}_3 = \{C^s_{m^s} : 1 \leq s \leq q\}$.

$\mathcal{R}_1 = \{CC^s : 1 \leq s \leq q\}$,
$\mathcal{R}_2 = \bigcup_{s=1}^{q} \{Y^s_j = C^s_{j+1} \oplus P^s_{j+1} : 0 \leq j \leq m^s - 2\}$,
$\mathcal{R}_3 = \{MM^s : 1 \leq s \leq q\}$.

Before we present the collision analysis let us identify the random variables based on which the probability of collision would be computed. In game RAND2 the hash key h is selected uniformly from the set $\{0,1\}^n$. The outputs that the adversary receives are also uniformly distributed, and are independent of the previous queries supplied by the adversary and the outputs obtained by the adversary. The i^{th} query supplied by the adversary may depend on the previous outputs obtained by the adversary, but as the output of GAME2 is not dependent in any way on the hash key h thus the queries supplied by the adversary are independent of h.

We consider T^s as t^s n-bit blocks. Thus, for any s, H_h either in line 108, 109, 208 or 209 of game RAND2 has degree at most m^s+t^s+2. Let us write $\sigma = \sum_s t^s + \sum_s m^s$ and $\ell^{s,s'} = \max\{t^s + m^s, t^{s'} + m^{s'}\} + 2$. Since $\ell^{s,s'} \leq m^s + m^{s'} + t^s + t^{s'} + 2$, we have the following inequality

$$\begin{aligned}\sum_{1\leq s<s'\leq q} \ell^{s,s'} &\leq \sum_{1\leq s<s'\leq q} (t^s + t^{s'}) + \sum_{1\leq s<s'\leq q} (m^s + m^{s'}) + \binom{q}{2}2 \\ &\leq (q-1)(\sigma + q).\end{aligned}$$

As we already noted the response of encryption or decryption query are completely independent of h (the poly hash key). Thus, inputs of $H_h(\cdot)$ for each query are independent of h. So we can use the fundamental theorem of algebra to claim that the probability that h is a root of a d degree polynomial is at most $d/2^n$ where h is chosen uniformly and independently from the coefficient of the polynomial, which is true in case of H_h in RAND2 game.

Claim 4.6.1. *The upper bound for the probability of collision in $\mathcal{D}_1$ is $\binom{q}{2}/2^n$.*

Proof. When the adversary does an encryption query we have no collision in $\mathcal{D}_1$ because of the condition in line 101 of RAND2. Otherwise when the adversary does a decryption query $\mathcal{D}_1$ has q elements and each element is chosen uniformly at random from $\{0,1\}^n$. Thus,

$$\Pr[P^s_{m^s} = P^{s'}_{m^{s'}} : \text{ for some } 1 \leq s < s' \leq q] \leq \tbinom{q}{2}/2^n. \tag{4.18}$$

□

For the collisions in $\mathcal{D}_2$, we need to find the probability of $\mathsf{inc}^j(S^s) = \mathsf{inc}^{j'}(S^{s'})$ for some $1 \leq s < s' \leq q$, $0 \leq j \leq m^s - 2$ and $0 \leq j' \leq m^{s'} - 2$. Let $\mathrm{COLL}_h(r, S^s, S^{s'})$ denote the event

$$\mathsf{inc}^r(S^s) \oplus S^{s'} = 0^n, \text{ where } r = j - j', \tag{4.19}$$

without loss of generality, let us assume that $j - j' \geq 0$, since $m^s, m^{s'} \leq 2^{32}$ when the block size of the underlying block cipher is 128 bits, the range of r is as $0 \leq r \leq 2^{32}-2$. Then we have that

$$\Pr\left[\mathsf{inc}^j(S^s) = \mathsf{inc}^{j'}(S^{s'})\right] = \Pr\left[h \xleftarrow{\$} \{0,1\}^n : \mathrm{COLL}_h(r, S^s, S^{s'})\right]. \tag{4.20}$$

In [IOM12], Iwata, Ohashi and Minematsu introduced a combinatorial problem in order to derive an upper bound on $\Pr\left[h \xleftarrow{\$} \{0,1\}^n : \mathrm{COLL}_h(r, S^s, S^{s'})\right]$, here we use the same combinatorial problem.

Problem 4.6.1 (Combinatorial problem)**.** *For* $0 \leq r \leq 2^{32}-1$*, let*

$$\mathbb{Y}_r = \{\mathsf{bin}_{32}(\mathsf{int}(Y) + r \ mod\ 2^{32}) \oplus Y | Y \in \{0,1\}^{32}\}. \tag{4.21}$$

We also let $\alpha_r = |\mathbb{Y}_r|$ *denote the cardinality of* $\mathbb{Y}_r$*, and* $\alpha_{\max} = \max\{\alpha_r | 0 \leq r \leq 2^{32}-1\}$*. Find* $\alpha_{\max}$*.*

The following lemma shows the relationship of α_r with our desired bound on $\Pr\left[h \xleftarrow{\$} \{0,1\}^n : \mathrm{COLL}_h(r, S^s, S^{s'})\right]$.

Lemma 4.6.1. *For any* $0 \leq r \leq 2^{32}-1$*,* S^s *and* $S^{s'}$ *polynomials of degree at most* $\ell^{s,s'}$*,* $S^s \neq S^{s'}$*,* $\Pr\left[h \xleftarrow{\$} \{0,1\}^n : \mathrm{COLL}_h(r, S^s, S^{s'})\right] \leq \alpha_r \ell^{s,s'}/2^n$*.*

Proof. The proof is exactly the same as the proof of Lemma 2 in [IOM12], we present it here for completeness. Fix r, then the set $\mathbb{Y}_r$ has α_r many elements

$$\mathbb{Y}_r = \{Y_1, Y_2, \ldots, Y_{\alpha_r}\}.$$

From the definition of the set $\mathbb{Y}_r$ we can see that each element of $\mathbb{Y}_r$ is a class representative of an equivalence class.

$$[Y_j] = \{Y \in \{0,1\}^{32} | \mathsf{bin}_{32}(\mathsf{int}(Y) + r \bmod 2^{32}) \oplus Y = Y_j\}, 1 \leq j \leq \alpha_r.$$

Let $\mathcal{Y}_j = [Y_j]$, as each $\mathcal{Y}_j$ is an equivalence class we have $\mathcal{Y}_1 \cup \cdots \cup \mathcal{Y}_{\alpha_r} = \{0,1\}^{32}$, and $\mathcal{Y}_j \cap \mathcal{Y}_{j'} = \emptyset$ for $1 \leq j \leq j' \leq \alpha_r$.

Observe that if $Y \in \mathcal{Y}_j$, then $\mathsf{bin}_{32}(\mathsf{int}(Y) + r \bmod 2^{32})$ can be replaced with $Y \oplus Y_j$. For $1 \leq j \leq \alpha_r$, let D_j be the event $\mathrm{COLL}_h(r, S^s, S^{s'}) \wedge \mathsf{lsb}_{32}(S^s) \in \mathcal{Y}_j$. Since $\mathrm{D}_1, \ldots, \mathrm{D}_{\alpha_r}$ are disjoint events, we have

$$\Pr\left[h \xleftarrow{\$} \{0,1\}^n : \mathrm{COLL}_h(r, S^s, S^{s'})\right] = \sum_{1 \leq j \leq \alpha_r} \Pr\left[\mathrm{D}_j\right]. \tag{4.22}$$

Recall that $\mathrm{COLL}_h(r, S^s, S^{s'})$ denotes the event $\mathsf{inc}^r(S^s) \oplus S^{s'} = 0^n$, and since $\mathsf{lsb}_{32}(S^s) \in \mathcal{Y}_j$, $\mathsf{inc}^r(S^s)$ can be replaced with $S^s \oplus (0^{n-32} || Y_j)$, implying that the event D_j is equivalent to

$$S^s \oplus S^{s'} \oplus \left(0^{n-32} || Y_j\right) = 0^n \tag{4.23}$$

and $\mathsf{lsb}_{32}(S^s) \in \mathcal{Y}_j$. We see that Eq. (4.23) is a non-trivial equation in h of degree at most $\ell^{s,s'}$ over $\mathrm{GF}(2^n)$, and hence it has at most $\ell^{s,s'}$ solutions. From Eq. (4.22), we obtain the lemma. □

In [IOM12] a method was also given to compute α_r and $\alpha_{\max}$, where it was found that $\alpha_{\max} = 3524578 \leq 2^{22}$. Thus by Lemma 4.6.1 we have,

$$\Pr\left[h \xleftarrow{\$} \{0,1\}^n : \mathrm{COLL}_h(r, S^s, S^{s'})\right] \leq \frac{2^{22}\ell^{s,s'}}{2^n}. \tag{4.24}$$

With this we are ready to compute the collision probability in $\mathcal{D}_2$.

Claim 4.6.2. *The upper bound for the probability of collision in $\mathcal{D}_2$ is*

$$\binom{\sum_s m^s - q}{2}/2^n + \binom{q}{2}\frac{2^{22}\ell_{\max}^2}{2^{n-1}}.$$

Proof. In line 108 of RAND2 we observe that S^s is obtained from $CC^s \oplus H_h(\cdot)$ for an encryption query (and from $MM^s \oplus H_h(\cdot)$ for a decryption query), given the condition in lines 101 and 201 we have two cases:

- If $P^s_{m^s} \neq P^{s'}_{m^{s'}}$ or $C^s_{m^s} \neq C^{s'}_{m^{s'}}$ for $s' < s$ then CC^s or MM^s is chosen at random, and so the probability of collision among S^s and $S^{s'}$ for $s \neq s'$ is $1/2^n$. Thus,

$$\begin{aligned}\Pr\ \ [&\mathsf{inc}^j(S^s) = \mathsf{inc}^{j'}(S^{s'}) : \text{ for some } 1 \leq s' < s \leq q,\\ &0 \leq j \leq m^s - 2, 0 \leq j' \leq m^{s'} - 2] \leq \binom{\sum_s m^s - q}{2}/2^n.\end{aligned} \tag{4.25}$$

- If $P^s_{m^s} = P^{s'}_{m^{s'}}$ or $C^s_{m^s} = C^{s'}_{m^{s'}}$ for $s' < s$, then we need to compute the probability of collision between $\mathsf{inc}^j(S^s)$ and $\mathsf{inc}^{j'}(S^{s'})$ for $1 \leq s \leq s' \leq q$, $0 \leq j \leq m^s - 2$ and $0 \leq j' \leq m^{s'} - 2$.

  The collision between $\mathsf{inc}^j(S^s)$ and $\mathsf{inc}^{j'}(S^{s'})$ happens whenever any of the following two events occurs.

$$S^s \oplus \mathsf{inc}^{j'}(S^{s'}), 0 \leq r \leq m^{s'} - 2$$

  and

$$\mathsf{inc}^j(S^s) \oplus S^{s'}, 0 \leq r \leq m^s - 2.$$

  By Lemma 4.6.1 we obtain the probability of $\mathsf{inc}^j(S^s) = \mathsf{inc}^{j'}(S^{s'})$ for fixed S^s and $S^{s'}$.

$$\Pr[\mathsf{inc}^j(S^s) = \mathsf{inc}^{j'}(S^{s'})] \leq \sum_{r=1}^{m^s+m^{s'}-4} \frac{\alpha_r \ell^{s,s'}}{2^n} < \frac{\alpha_{\max}\ell^{s,s'}(m^s + m^{s'})}{2^n}.$$

  Now we are able to compute the probability of collision for $1 \leq s < s' \leq q$,

$$\begin{aligned}\Pr[h \xleftarrow{\$} \{0,1\}^n : \mathrm{COLL}_h(r, S^s, S^{s'})] &\leq \sum_{1\leq s<s'\leq q} \frac{\alpha_{\max}\ell^{s,s'}(m^s + m^{s'})}{2^n}\\ &\leq \binom{q}{2}\left(\frac{2^{22}\ell^{s,s'}}{2^n}\right)\left(2\ell^{s,s'}\right)\\ &\leq \binom{q}{2}\frac{2^{22}\ell_{\max}^2}{2^{n-1}}.\end{aligned} \tag{4.26}$$

  From Eq. (4.25) and Eq. (4.26) we obtain the bound.

□

Claim 4.6.3. *The upper bound for the probability of collision in $\mathcal{D}_3$ is $\binom{q}{2}/2^n$.*

The proof of Claim 4.6.3 is similar to that of Claim 4.6.1.

Claim 4.6.4. *The upper bound for the probability of collision in $\mathcal{R}_1$ is $\binom{q}{2}/2^n$.*

Proof. For the collision in $\mathcal{R}_1$ we have two cases: one for an encryption query and another one for a decryption query.

- When the adversary does an encryption query, CC^s is chosen uniformly at random from $\{0,1\}^n$. Thus,

$$\Pr[CC^s = CC^{s'} : \text{ for some } 1 \leq s < s' \leq q] \leq \binom{q}{2}/2^n. \tag{4.27}$$

- For the case of a decryption query, from RAND2 line 208 we have

$$\begin{aligned} CC^s &\leftarrow MM^s \oplus H_h(0^n||T^s, P_1^s||\ldots||P_{m^s-1}^s||0^n) \\ &\oplus H_h(T^s||0^n, C_1^s||\ldots||C_{m^s-1}^s||(\mathsf{bin}_{\frac{n}{2}}(|T^s||0^n|)||\mathsf{bin}_{\frac{n}{2}}(|C_1^s||\ldots||C_{m^s-1}^s|))). \end{aligned}$$

$P_1^s, \ldots, P_{m^s}^s$ are chosen uniformly at random from $\{0,1\}^n$ in line 22 of RAND2. Thus

$$\Pr[CC^s = CC^{s'} : \text{ for some } 1 \leq s < s' \leq q] \leq \binom{q}{2}/2^n. \tag{4.28}$$

From Eq. (4.27) and Eq. (4.28) we obtain the bound.

□

Claim 4.6.5. *The upper bound for the probability of collision in $\mathcal{R}_2$ is $\binom{\sum_s m^s - q}{2}/2^n$.*

Proof. We consider collision among Y_i^s, $0 \leq i \leq m^s - 2$, $1 \leq s \leq q$. For the pairs $(Y_i^s, Y_{i'}^{s'})$ with $s' \leq s$ and $(s,i) \neq (s',i')$, the collision probability is $1/2^n$, since either P^s or C^s is chosen uniformly and independently from the rest of the variables. There are $\binom{\sum_s m^s - q}{2}$ pairs of this form. Thus,

$$\begin{aligned} \Pr[Y_i^s = Y_{i'}^{s'} : \text{ for some } 1 \leq s \leq s' \leq q, 1 &\leq i \leq m^s, 1 \leq i' \leq m^{s'}, (s,i) \neq (s',i')] \\ &\leq \binom{\sum_s m^s - q}{2}/2^n. \end{aligned} \tag{4.29}$$

□

Claim 4.6.6. *The upper bound for the probability of collision in $\mathcal{R}_3$ is $\binom{q}{2}/2^n$.*

The proof of Claim 4.6.6 is similar to that of Claim 4.6.4.
From Claim 4.6.1 to Claim 4.6.6 we obtain

$$\Pr[A^{\text{RAND2}} \text{ sets } \mathsf{bad}] \leq (2^{22}q^2\ell_{\max}^2 + 1.5\sigma^2)/2^{n-1}.$$

Using the above equation and Eq. (4.17) we obtain Eq. (4.10), which completes the proof.

4.6.3 Real world scenario

We consider the situation in which the adversary has access to a 1Tb hard drive. We give the adversary the power to choose either the plaintext or the ciphertext for each sector of the hard drive. We consider each query the adversary does as a message of 4096 bytes, as this is the size of a sector for the currently available hard drives. We consider the tweak to be 16 bytes.

With the above data we can compute the following information for a block length of $n = 128$ bits.

- Each sector contains 2^8 blocks.
- The size of the tweak is equal to 1 block.
- The adversary can ask at most $q = 2^{28}$ queries.
- The complexity of the queries asked by the adversary is $\sigma = 2^{28}(1 + 2^8)$.

As we note in the proof of Claim 4.6.2 the upper bound for XCBv2 depends on α_r where r ranges from 0 to the number of blocks in the message. For this scenario we do not consider $\alpha_{\max} = 2^{22}$ which is the general case when $0 \leq r \leq 2^{32} - 1$ but instead we consider $\alpha_{\max} = 2^{11}$ as this is the case when $0 \leq r \leq 2^8$ in Problem 4.6.1.

Now we can compute the advantage of the adversary in distinguishing XCBv2.

$$\begin{aligned} \mathbf{Adv}^{\pm\widetilde{\text{prp}}}_{\text{XCBv2}[E]}(\sigma, t) &\leq (2^{11} q^2 \ell^2_{\max} + 2\sigma^2)/2^{n-1} \\ &\leq (2^{84} + 2^{64})/2^{127}. \end{aligned}$$

4.7 The Security of XCBv1

When XCBv1 was proposed no proof of its security was given. In this section we give a security bound for XCBv1.

The following theorem specifies the security of XCBv1.

Theorem 4.7.1. *Fix n, σ to be positive integers and an n-bit block cipher $E : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$. Then*

$$\mathbf{Adv}^{\pm\widetilde{\text{prp}}}_{\text{XCBv1}[\text{Perm}(n),\text{Perm}(n),\text{Perm}(n)]}(\sigma) \leq (2^{22} q^2 \ell^2_{\max} + 2\sigma^2)/2^{n-1}. \tag{4.30}$$

$$\mathbf{Adv}^{\pm\widetilde{\text{prp}}}_{\text{XCBv1}[E]}(\sigma, t) \leq (2^{22} q^2 \ell^2_{\max} + 2\sigma^2)/2^{n-1} + \mathbf{Adv}^{\pm\text{prp}}_{E}(\sigma, t') \tag{4.31}$$

where $t' = t + O(\sigma)$ and $\ell_{\max}$ is the maximum query length.

The analysis of XCBv1 is very similar to that of XCBv2. In this section we just mention the differences. For the game sequence we just write the final game RAND2 and the bounds for the collision probabilities of the sets $\mathcal{D}_i$ and $\mathcal{R}_i$.

As in Eq. (4.9) we have

$$\begin{aligned}\mathbf{Adv} \quad & \overset{\pm\widetilde{\text{prp}}}{\text{XCBv1[Perm}(n)\text{,Perm}(n)\text{,Perm}(n)]}(A) \\ & \leq \mathbf{Adv}^{\pm\text{rnd}}_{\text{XCBv1[Perm}(n)\text{,Perm}(n)\text{,Perm}(n)]}(A) + \binom{q}{2}\frac{1}{2^n}.\end{aligned}$$

Now, the main task of the proof reduces to obtaining an upper bound on

$$\mathbf{Adv}^{\pm\text{rnd}}_{\text{XCBv1[Perm}(n)\text{,Perm}(n)\text{,Perm}(n)]}(\sigma).$$

4.7.1 The game sequence

The interaction of the adversary with XCBv1 is modeled by a sequence of games. We start with a game which describes the mode XCBv1 and with small changes we reach the game RAND2 which represents an oracle which returns random strings. Steps taken to reach game RAND2 in XCBv1 are the same of those of XCBv2. Game RAND2 for XCBv1 is shown in Fig. 4.4.

Note that game RAND2 in Fig. 4.4 slightly differs from that of Fig. 4.4. The differences are in the inputs of the hash function. The order of the inputs change and also their lengths. For XCBv2 we add a block of zeros to the inputs. Another difference is that in XCBv1 we have two different keys (h_1,h_2) for the computation of the hash function while in XCBv2 we only have one key.

Recall from Section 4.6.1 that

$$\mathbf{Adv}^{\pm\text{rnd}}_{\text{XCBv1[Perm}(n)\text{,Perm}(n)\text{,Perm}(n)]}(A) \leq \Pr[A^{\text{RAND2}} \text{ sets } \mathsf{bad}]. \tag{4.32}$$

Now our goal is to bound $\Pr[A^{\text{RAND2}} \text{ sets } \mathsf{bad}]$. From Claim 4.7.1 to Claim 4.7.6 we obtain

$$\Pr[A^{\text{RAND2}} \text{ sets } \mathsf{bad}] \leq (2^{22}q^2\ell^2_{\max} + 1.5\sigma^2)/2^{n-1}.$$

We notice that in Game RAND2 the bad flag is set when there is a collision in either of the sets $\mathcal{D}_i$ or $\mathcal{R}_i$. So if COLLD_i and COLLR_i represent the event of a collision in $\mathcal{D}_i$ and $\mathcal{R}_i$ respectively then

$$\Pr[A^{\text{RAND2}} \text{ sets } \mathsf{bad}] \leq \sum_{i=1}^{3} (\Pr[\mathsf{COLLR}_i] + \Pr[\mathsf{COLLD}_i])$$

Bounding collision probability in $\mathcal{D}_i$ and $\mathcal{R}_i$

After q queries of the adversary where the s^{th} query has m^s blocks of plaintext or ciphertext and t^s blocks of tweak, then the sets $\mathcal{D}_i$ and $\mathcal{R}_i$ can be written as follows:

Respond to the s^{th} adversary query as follows:

ENCIPHER QUERY $\mathsf{Enc}(T^s; P_1^s, P_2^s, \ldots, P_{m^s}^s)$

$ty^s = \mathsf{Enc}$
$C_1^s||C_2^s||\ldots||C_{m^s-1}^s||D_{m^s}^s \xleftarrow{\$} \{0,1\}^{nm^s}$
$C_{m^s}^s \leftarrow \mathsf{drop}_{n-r^s}(D_{m^s})$
return $C_1^s||C_2^s||\ldots||C_{m^s}^s$

DECIPHER QUERY $\mathsf{Dec}(T^s; C_1^s, C_2^s, \ldots, C_{m^s}^s)$

$ty^s = \mathsf{Dec}$
$P_1^s||P_2^s||\ldots||P_{m^s-1}^s||V_{m^s}^s \xleftarrow{\$} \{0,1\}^{nm^s}$
$P_{m^s}^s \leftarrow \mathsf{drop}_{n-r^s}(V_{m^s})$
return $P_1^s||P_2^s||\ldots||P_{m^s}^s$

Finalization:

$h \xleftarrow{\$} \{0,1\}^n$
for $s = 1$ to q,
if $ty^s = \mathsf{Enc}$ **then**
if $P_1^s = P_1^{s'}$ for $s' < s$ **then**
$CC^s \leftarrow CC^{s'}$
else
$CC^s \xleftarrow{\$} \{0,1\}^n$
$\mathcal{D}_1 \leftarrow \mathcal{D}_1 \cup \{P_1^s\}$
$\mathcal{R}_1 \leftarrow \mathcal{R}_1 \cup \{CC^s\}$
end if
$S^s \leftarrow CC^s \oplus H_{h_1}(P_2^s||\ldots||P_{m^s}^s, T^s)$
$MM^s \leftarrow S^s \oplus H_{h_2}(C_2^s||\ldots||C_{m^s}^s, T^s)$
$\mathcal{D}_3 \leftarrow \mathcal{D}_3 \cup \{C_1^s\}$
$\mathcal{R}_3 \leftarrow \mathcal{R}_3 \cup \{MM^s\}$
for $i = 0$ to $m^s - 3$,
$Y_i^s \leftarrow C_{i+2}^s \oplus P_{i+2}^s$
$\mathcal{D}_2 \leftarrow \mathcal{D}_2 \cup \{\mathsf{inc}^i(S^s)\}$
$\mathcal{R}_2 \leftarrow \mathcal{R}_2 \cup \{Y_i^s\}$
end for
$Y_{m^s-2}^s \leftarrow C_{m^s}^s \oplus P_{m^s}^s$
$\mathcal{D}_2 \leftarrow \mathcal{D}_2 \cup \{\mathsf{inc}^{m^s-2}(S^s)\}$
$\mathcal{R}_2 \leftarrow \mathcal{R}_2 \cup \{Y_{m^s-2}^s\}$
else if $ty^s = \mathsf{Dec}$ **then**
if $C_1^s = C_1^{s'}$ for $s' < s$ **then**
$MM^s \leftarrow MM^{s'}$
else
$MM^s \xleftarrow{\$} \{0,1\}^n$
$\mathcal{D}_3 \leftarrow \mathcal{D}_3 \cup \{C_1^s\}$
$\mathcal{R}_3 \leftarrow \mathcal{R}_3 \cup \{MM^s\}$
end if
$S^s \leftarrow MM^s \oplus H_{h_2}(C_2^s||\ldots||C_{m^s}^s, T^s)$
$CC^s \leftarrow S^s \oplus H_{h_1}(P_2^s||\ldots||P_{m^s}^s, T^s)$
$\mathcal{D}_1 \leftarrow \mathcal{D}_1 \cup \{P_1^s\}$
$\mathcal{R}_1 \leftarrow \mathcal{R}_1 \cup \{CC^s\}$
for $i = 0$ to $m^s - 3$,
$Y_i^s \leftarrow C_{i+2}^s \oplus P_{i+2}^s$
$\mathcal{D}_2 \leftarrow \mathcal{D}_2 \cup \{\mathsf{inc}^i(S^s)\}$
$\mathcal{R}_2 \leftarrow \mathcal{R}_2 \cup \{Y_i^s\}$
end for
$Y_{m^s-2}^s \leftarrow C_{m^s}^s \oplus P_{m^s}^s$
$\mathcal{D}_2 \leftarrow \mathcal{D}_2 \cup \{\mathsf{inc}^{m^s-2}(S^s)\}$
$\mathcal{R}_2 \leftarrow \mathcal{R}_2 \cup \{Y_{m^s-2}^s\}$
end if
end for

SECOND PHASE
bad = false;
if (some value occurs more than once in $\mathcal{D}_i, i = 1, 2, 3$) **then** bad = true **end if**;
if (some value occurs more than once in $\mathcal{R}_i, i = 1, 2, 3$) **then** bad = true **end if**.

Figure 4.4: Game RAND2 for XCBv1.

$\mathcal{D}_1 = \{P_1^s : 1 \le s \le q\}$,
$\mathcal{D}_2 = \bigcup_{s=1}^{q} \{\mathsf{inc}^j(S^s) : 0 \le j \le m^s - 2\}$,
$\mathcal{D}_3 = \{C_1^s : 1 \le s \le q\}$.

$\mathcal{R}_1 = \{CC^s : 1 \le s \le q\}$,
$\mathcal{R}_2 = \bigcup_{s=1}^{q} \{Y_j^s = C_{j+2}^s \oplus P_{j+2}^s : 0 \le j \le m^s - 2\}$,
$\mathcal{R}_3 = \{MM^s : 1 \le s \le q\}$.

We consider T^s as t^s n-bit blocks. Thus, for any s, $H_{h_1}(P_2^s||\cdots||P_{m^s}^s, T^s)$ or $H_{h_2}(C_2^s||\cdots||C_{m^s}^s, T^s)$ has degree at most $m^s + t^s$. Let us write $\sigma = \sum_s t^s + \sum_s m^s$ and $\ell^{s,s'} = \max\{m^s + t^s, m^{s'} + t^{s'}\}$. Since $\ell^{s,s'} \le m^s + m^{s'} + t^s + t^{s'}$, we have the following inequality

$$\begin{aligned}\sum_{1\le s<s'\le q} \ell^{s,s'} &\le \sum_{1\le s<s'\le q} (t^s + t^{s'}) + \sum_{1\le s<s'\le q} (m^s + m^{s'}) \\ &\le (q-1)(\sigma - \sum_s m^s) + (q-1)(\sigma - \sum_s t^s) \\ &\le (q-1)\sigma.\end{aligned}$$

We also note that the response of encryption or decryption query are completely independent of h_1 and h_2 (the poly hash keys). Thus, inputs of $H_{h_1}(\cdot)$ for each query are independent with h_1. So we can use the fundamental theorem of algebra to claim that the probability that h_1 is a root of a d degree polynomial is at most $d/2^n$ where h_1 is chosen uniformly and independently from the coefficient of the polynomial, which is true in case of H_{h_1} in RAND2 game (this scenario also applies to H_{h_2} and h_2).

The following claims except Claim 4.7.4 and Claim 4.7.6 are similar to those of Section 4.6.2.

Claim 4.7.1. *The upper bound for the probability of collision in $\mathcal{D}_1$ is $\binom{q}{2}/2^n$.*

Claim 4.7.2. *The upper bound for the probability of collision in $\mathcal{D}_2$ is*

$$\binom{\sum_s m^s - q}{2}/2^n + \binom{q}{2}\frac{2^{22}\ell_{\max}^2}{2^{n-1}}.$$

Claim 4.7.3. *The upper bound for the probability of collision in $\mathcal{D}_3$ is $\binom{q}{2}/2^n$.*

Claim 4.7.4. *The upper bound for the probability of collision in $\mathcal{R}_1$ is $\binom{q}{2}/2^n$.*

Proof. For the collision in $\mathcal{R}_1$ we have two cases: one for an encryption query and another one for a decryption query.

- When the adversary does an encryption query, CC^s is chosen at random from $\{0,1\}^n$. Thus,

$$\Pr[CC^s = CC^{s'} : \text{ for some } 1 \le s < s' \le q] \le \binom{q}{2}/2^n. \tag{4.33}$$

- For the case of a decryption query, from RAND2 line 109 we have

$$CC^s \leftarrow MM^s \oplus H_{h_1}(P_2^s||\ldots||P_{m^s}^s, T^s) \oplus H_{h_2}(C_2^s||\ldots||C_{m^s}^s, T^s). \tag{4.34}$$

  Similar to the proof of Claim 4.6.2 we have two cases derived from the condition in line 101.

  – If $C_1^s \neq C_1^{s'}$ for $s' < s$ then MM^s is chosen at random, thus

$$\Pr[CC^s = CC^{s'} : \text{ for some } 1 \leq s < s' \leq q] \leq \binom{q}{2}/2^n. \tag{4.35}$$

    This case is covered in Eq. (4.33).

  – Let $H_{h_1}^s$ denote $H_{h_1}(P_2^s||\ldots||P_{m^s}^s, T^s)$, $H_{h_2}^s$ denote $H_{h_2}(C_2^s||\ldots||C_{m^s}^s, T^s)$, $H_{h_1}^{s'}$ denote $H_{h_1}(P_2^{s'}||\ldots||P_{m^{s'}}^{s'}, T^{s'})$ and $H_{h_2}^{s'}$ denote $H_{h_2}(C_2^{s'}||\ldots||C_{m^{s'}}^{s'}, T^{s'})$. For the case $C_1^s = C_1^{s'}$ for $s' < s$, both MM^s and $MM^{s'}$ are equal and then $CC^s \oplus CC^{s'} = H_{h_1}^s \oplus H_{h_2}^s \oplus H_{h1}^{s'} \oplus H_{h_2}^{s'}$. Recall that $P_2^s, \ldots, P_{m^s}^s$ and $P_2^{s'}, \ldots, P_{m^{s'}}^{s'}$ are chosen independently at random (line 22 of RAND2), thus we consider $H_{h_1}^s$ and $H_{h_1}^{s'}$ as chosen independently at random and $\Pr[CC^s = CC^{s'}] = 1/2^n$, hence

$$\Pr[CC^s = CC^{s'} : \text{ for some } 1 \leq s < s' \leq q] \leq \binom{q}{2}/2^n. \tag{4.36}$$

    This case is also covered in Eq. (4.33).

□

Claim 4.7.5. *The upper bound for the probability of collision in $\mathcal{R}_2$ is $\binom{\sum_s m^s - q}{2}/2^n$.*

Claim 4.7.6. *The upper bound for the probability of collision in $\mathcal{R}_3$ is $\binom{q}{2}/2^n$.*

The proof of Claim 4.7.6 is similar to that of Claim 4.7.4.
Using Claims 4.7.1 to 4.7.6 and the union bound we have

$$\sum_{i=1}^{3} (\Pr[\mathsf{COLLR}_i] + \Pr[\mathsf{COLLD}_i]) \leq (2^{22} q^2 \ell_{\max}^2 + 1.5\sigma^2)/2^{n-1},$$

which give us an upper bound for $\Pr[A^{\mathrm{RAND2}}$ sets $\mathsf{bad}]$ in Eq. (4.32), completing the proof.

5. Security of Encryption Schemes on Key Dependent Messages

> I have not received any meaningful response to the issue of grandma storing her keys on an encrypted drive.
>
> Email archives IEEE P1619

In this chapter we investigate the strange scenario where an encryption algorithm encrypts its own key. This phenomenon has been named in the literature as a *key cycle*, and it has always been seen as an abuse of encryption. Thus, traditionally it has been implicitly assumed that an encryption algorithm will never encrypt its own key. As such an assumption is widely accepted thus there have been very limited analysis of encryption schemes in this scenario. In recent days, this question has gained some attention in the context of the formulation of the standard IEEE Std 1619.2-2010, which is a standard for disk encryption. The phrase in the epigraph of this chapter has been quoted from an email archive of the discussions of the working group responsible for the standard. The working group of P1619 considered key cycles to be a real threat, and it is supposed that they deleted one scheme called LRW from their list of possible schemes in IEEE Std 1619-2007 as there was a glaring insecurity of LRW if it was used to encrypt its own key.

The preliminary definitional work involving this paradigm was initiated in [BRS02], where security of encryption schemes when queried on key dependent messages was formalized. In [HK07] the security of tweakable enciphering schemes on key dependent messages were defined, and the key dependent message insecurity of LRW was also pointed out. Recently in [BCK11] some generic schemes were discussed to make any secure TES secure in the KDM sense.

In this chapter we discuss the basic notions of KDM security following the work of [BRS02] and [HK07]. We also demonstrate KDM insecurity of two TES namely HCTR and XCB, these attacks were not known before. Finally we discuss two transforms proposed in [BCK11] to convert a secure TES to a KDM secure TES.

5.1 Definitions and Security Notions

Black, Rogaway and Shrimpton gave the notion of KDM in [BRS02]. The following discussion of KDM is based on their work.

First we give an informal description of the KDM security notion. Consider an encryption scheme $\mathcal{E}$ instantiated by a key K. We allow an adversary to obtain encryptions of $g(K)$, for any function g chosen by the adversary. We call $\mathcal{E}$ to be KDM secure if the adversary cannot infer anything meaningful regarding the cipher given this extra information. Note that the adversary here has access to some information regarding the key through its encryption, and it is infeasible for any adversary to obtain this information in the normal chosen plaintext attack scenario.

We formalize the notion of KDM security in the following definition.

Definition 5.1.1. *[IND-KDM] [BRS02] Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme and let $\mathcal{A}$ be an adversary. For $K \in \mathcal{K}$,*

- *Let Real_K be the oracle that on input g returns $\mathcal{E}_K(g(K))$*
- *Let Fake_K be the oracle that on input g returns $\mathcal{E}_K(0^{|g(K)|})$*

The IND-KDM advantage of an adversary A is defined as

$$\mathbf{Adv}^{\mathrm{kdm}}_{\Pi}(A) = \left|\Pr\left[K \xleftarrow{\$} \mathcal{K} : A^{\mathsf{Real}_K} \Rightarrow 1\right] - \Pr\left[K \xleftarrow{\$} \mathcal{K} : A^{\mathsf{Fake}_K} \Rightarrow 1\right]\right|.$$

Π is KDM-secure if $\mathbf{Adv}^{\mathrm{kdm}}_{\Pi}(A)$ is "small" for all "efficient" adversaries A.

The function g in the above definition is called a plaintext construction function. For the above definition to be meaningful, it is required that the length of $g(K)$ (denoted by $|g(K)|$) does not depend on K, i.e., the plaintext construction function g is required to be of fixed length. When this is not the case, KDM-secure schemes would not exist. To see this, consider, for example, the function $g_i(K)$, defined as $g_i(K) = 1$ if the i-th bit of K is 1 and $g_i(K) = 00$ if the i-th bit of K is zero. If the encryption function $\mathcal{E}$ reveals the length of the plaintext (which most encryption algorithms do) then an adversary can ask for $g_i(K)$-values as a way to learn any key, bit by bit.

KDM is a strong notion of security, it implies security in other senses, for example given an encryption scheme that is prp-secure, one can trivially modify it to construct a similar prp-secure encryption scheme that is completely insecure in the KDM sense. Let $\mathcal{E}_K()$ be a prp secure encryption scheme. We construct $\tilde{\mathcal{E}}_K()$ from $\mathcal{E}_K()$ as

$$\tilde{\mathcal{E}}_K(X) = \begin{cases} 0||\mathcal{E}_K(M) & \text{if } M \neq K \\ 1||K & \text{otherwise.} \end{cases}$$

$\tilde{\mathcal{E}}_K()$ is trivially insecure in the KDM sense though it is prp-secure. This shows that KDM security implies prp-security but KDM security is not implied by prp-security.

Black, Rogaway and Shrimpton worked on KDM security in the context of randomized encryption, their study does not include deterministic encryption schemes such as TES. In a later work, Halevi and Krawczyk [HK07] extend the study of KDM in the context of deterministic symmetric schemes and rename the security notion as Key Dependent Inputs (KDI) security. The renaming was necessary as KDI does not only consider key dependent messages (which KDM does) but it also considers key dependent ciphertexts.

An important observation in [HK07] is that for the KDI notion to be meaningful in case of deterministic schemes, one needs to put a restriction on the functions g which an adversary can query. In particular they proved the following

Theorem 5.1.1 (No single construction for all g [HK07])**.** *There exists no deterministic construction of a pseudo-random function family that is KDI-secure with respect to all functions g.*

In practical terms, Theorem 5.1.1 implies that an application must restrict the types of information on the key that can be securely encrypted under the key itself.

The definition of KDI considered by Halevi and Krawczyk is more general than the KDM notion in Definition 5.1.1. In particular the KDI notion also encompasses the case of key dependent chosen ciphertext. As we mainly concentrate our study on TES, hence we state the definition of KDI security for TES.

Definition 5.1.2 (KDI-secure tweakable SPRPs)**.** *[HK07] A tweakable enciphering scheme $E : \mathcal{K} \times \mathcal{T} \times \mathcal{M} \to \mathcal{M}$ is KDI-secure with respect to a class Φ of functions if no feasible adversary A can distinguish with non-negligible advantage between the following two cases:*

1. *The key $K \in \mathcal{K}$ is chosen at random, and for any $T \in \mathcal{T}$, $X \in \mathcal{M}$, $g \in \Phi$ the oracles are set as, $\mathsf{Fn}(T, X) = E_K(T, X)$, $\mathsf{Fn}^{-1}(T, X) = E_K^{-1}(T, X)$, $\mathsf{KDFn}(T, g) = E_K(T, g(K))$ and $\mathsf{KDFn}^{-1}(T, g) = E_K^{-1}(T, g(K))$ where $\mathcal{T}$ is the tweak space and $\mathcal{M}$ is the message space;*

2. *The key $K \in \mathcal{K}$ is chosen at random, for every $T \in \mathcal{T}$ we set $\mathsf{Fn}(T, \cdot)$ to a random permutation and $\mathsf{Fn}^{-1}(T, \cdot)$ to its inverse, and then $\mathsf{KDFn}(T, g) = \mathsf{Fn}(T, g(K))$, and $\mathsf{KDFn}^{-1}(T, g) = \mathsf{Fn}^{-1}(T, g(K))$.*

In this work we will not consider security against key dependent ciphertexts. When we will be talking about KDM security, we will be referring to Φ-KDM security. We can think of Φ-KDM security as KDI security without the KDFn^{-1} oracle.

For convenience and compatibility with our notation, we would denote Φ-KDM advantage of an adversary A in distinguishing a TES E as

$$\begin{aligned}\mathbf{Adv}^{\pm\widetilde{\mathrm{prp}}}_{E,\Phi}(A) &= \Pr\left[K \xleftarrow{\$} \mathcal{K} : A^{E_K(\cdot,\cdot),E_K^{-1}(\cdot,\cdot),\mathsf{KDFn}_{E_K}(\cdot,\cdot)} \Rightarrow 1\right] \\ &\quad - \Pr\left[\boldsymbol{\pi} \xleftarrow{\$} \mathrm{Perm}^{\mathcal{T}}(\mathcal{M}) : A^{\boldsymbol{\pi}(\cdot,\cdot),\boldsymbol{\pi}^{-1}(\cdot,\cdot),\mathsf{KDFn}_{\boldsymbol{\pi}}(\cdot,\cdot)} \Rightarrow 1\right], \quad (5.1)\end{aligned}$$

where $\mathsf{KDFn}_{E_K}(T, g)$ returns $E_K(T, g(K))$ and $\mathsf{KDFn}_{\boldsymbol{\pi}}(T, g)$ returns $\boldsymbol{\pi}(T, g(K))$. Note g is queried by A and it is required that $g \in \Phi$.

5.2 KDM Attacks on some TES

As stated in the previous section, prp-security does not imply KDM security. In this section we show attacks on some encryption schemes that have been proved secure in the sense of SPRP but are KDM insecure. Concrete attacks in the KDM sense are rare. Though KDM insecurity of TES have received some attention and also in [BCK11] some transforms have been designed which makes an arbitrary secure TES, KDM secure. But concrete attacks on existing TES were not known. Here for the first time we show attacks against HCTR and XCB. The attack on LRW described here is taken from [HK07].

5.2.1 LRW attack

Recall from Section 3.1 that LRW is a tweakable block cipher proposed by Liskov, Rivest and Wagner. LRW was considered as a possible encryption standard for the IEEE P1619 standard group, but some members objected, citing an attack that can be mounted when the scheme is applied to its own secret key. The group rejected LRW when it was informed that the implementation of disk encryption in Windows Vista can store to the disk an encryption of its own secret keys in some situations. Consequently the group switched to a different scheme, for which the particular attack in question does not seem to apply.

Halevi and Krawczyk [HK07] described the attack on LRW. Let E be a block cipher, LRW construction is defined as follows:

$$\tilde{E}_{K_1,K_2}(T, M) = E_{K_1}(M \oplus h_{K_2}(T)) \oplus h_{K_2}(T).$$

Liskov et al. [LRW02] proved that $\tilde{E}$ is a secure tweakable cipher when h is an Almost Xor Universal (AXU) hash function. In P1619 the following instantiation of h was used. Let $h_{K_2}(T) = TK_2$, where $T, K_2 \in GF(2^n)$. With this instantiation LRW can be described as

$$\tilde{E}_{K_1,K_2}(T, M) = E_{K_1}((TK_2) \oplus M) \oplus (TK_2).$$

In [HK07] it was shown that the above construction is not KDM-secure with respect to the function $g(K_1, K_2) = K_2$.

The adversary can query $\mathsf{KDFn}(0, g)$ (i.e, using tweak value 0 and plaintext K_2) and also $\mathsf{Fn}(1, 0)$ (i.e, tweak value 1 and plaintext 0), thus getting

$$c_1 = \tilde{E}_{K_1,K_2}(0, K_2) = E_{K_1}(K_2) \text{ and } c_2 = \tilde{E}_{K_1,K_2}(1, 0) = E_{K_1}(K_2) \oplus K_2.$$

Next the adversary can compute $K_2 = c_1 \oplus c_2$ and then verify this value by asking to decrypt the value of $c_1 \oplus 2K_2$ with respect to the tweak value 2.

5.2.2 HCTR attack

HCTR uses two keys, denoted by K, h. K is used as a key for the underlying block cipher E and h the key for the polynomial hash is an n-bit string and it is treated as an element of $GF(2^n)$. HCTR is not KDM-secure with respect to the function $g(K,h) = X||0^n||h$, where $X \in \{0,1\}^n$ (i.e., when encrypting a message that depends on the element h of the secret key). The attacker can query $\mathsf{KDFn}(T, g)$, where $T \in \mathcal{T}$; and also $\mathsf{Fn}(T, Y)$, where $Y = X||\mathsf{bin}_n(1)||0^n$, thus getting:

In case of $\mathsf{KDFn}(T, g)$, from the construction of HCTR (see Fig. 3.5) we have:

$$
\begin{aligned}
&MM^{(1)} \leftarrow X \oplus H_h(0^n||h||T)\\
&CC^{(1)} \leftarrow E_K(X \oplus h^4 \oplus Th^2 \oplus \mathsf{bin}_n(3n)h)\\
&S^{(1)} \leftarrow (X \oplus h^4 \oplus Th^2 \oplus \mathsf{bin}_n(3n)h) \oplus E_K(X \oplus h^4 \oplus Th^2 \oplus \mathsf{bin}_n(3n)h)\\
&C_2^{(1)} \leftarrow E_K(S \oplus \mathsf{bin}_n(1))\\
&C_3^{(1)} \leftarrow h \oplus E_K(S \oplus \mathsf{bin}_n(2))\\
&C_1^{(1)} \leftarrow CC^{(1)} \oplus H_h(C_2^{(1)}||C_3^{(1)}||T)
\end{aligned}
$$

In case of $\mathsf{Fn}(T, Y)$, from the construction of HCTR we have:

$$
\begin{aligned}
&MM^{(2)} \leftarrow X \oplus H_h(\mathsf{bin}_n(1)||0^n||T)\\
&CC^{(2)} \leftarrow E_K(X \oplus h^4 \oplus Th^2 \oplus \mathsf{bin}_n(3n)h)\\
&S^{(2)} \leftarrow (X \oplus h^4 \oplus Th^2 \oplus \mathsf{bin}_n(3n)h) \oplus E_K(X \oplus h^4 \oplus Th^2 \oplus \mathsf{bin}_n(3n)h)\\
&C_2^{(2)} \leftarrow \mathsf{bin}_n(1) \oplus E_K(S \oplus \mathsf{bin}_n(1))\\
&C_3^{(2)} \leftarrow E_K(S \oplus \mathsf{bin}_n(2))\\
&C_1^{(2)} \leftarrow CC^{(2)} \oplus H_h(C_2^{(2)}||C_3^{(2)}||T)
\end{aligned}
$$

It is easy to see that $S^{(1)} = S^{(2)}$. Thus $C_3^{(1)} \oplus C_3^{(2)} = h$. Which shows that any adversary can obtain the hash key for HCTR with only two queries.

In general if we can make the value S the same for two different queries then we can mount an attack on HCTR. From the construction of HCTR we can see that it is sufficient to make the value of MM the same for two different queries. This can be easily done if a collision for H can be found. It is easy to find a collision on H if we allow the input to depend on the hash key h. Note that this is not related with the almost universality of the hash function H.

Now, we give a general way to find a collision in H.

Recall the definition of H from Section 3.4.2.

$$H_h(X) = X_1 h^{m+1} \oplus X_2 h^m \oplus \ldots \oplus X_m h^2 \oplus \mathsf{bin}_n(|X|)h.$$

For $X, X' \in \mathcal{M}$ such that $X \neq X'$ and $|X| = |X'| = m$. Let $X = X_1||\ldots||X_m$ and $X' = X_1'||\ldots||X_m'$, then we have

$$H_h(X) \oplus H_h(X') = (X_1 \oplus X_1')h^{m+1} \oplus (X_2 \oplus X_2')h^m \oplus \ldots \oplus (X_m \oplus X_m')h^2.$$

Fix any i, such that $2 \leq i \leq m$. Set $X_i = h$ and $X_{i-1} = 1$, and $X_j = 0$ for $1 \leq j \leq m$ and $j \neq i$, $j \neq i-1$. Set $X'_j = 0$ for $1 \leq j \leq m$. Then we have

$$H_h(X) \oplus H_h(X') = X_{i-1}h^{m+3-i} \oplus X_i h^{m+2-i} = h^{m+3-i} \oplus h^{m+3-i} = 0.$$

Now, we can mount an attack on HCTR using X and X'. By querying $\mathsf{KDFn}(T, X)$ and $\mathsf{Fn}(T, X')$, and then xoring the i-th block of both answers we can recover the key h.

5.2.3 XCB attack

In the description of XCBv1 and XCBv2 it is defined just one input key whose purpose is to derive the key material used by the scheme. This key material can be precomputed and stored in some place avoiding the processes of deriving it every time an encryption or decryption operation needs to be performed. With this scenario in mind we allow the adversary get access to the key h through the function $g \in \Phi$, mounting a similar attack to that of HCTR.

Recall the hash function used in XCB from Section 3.4.1. In case of XCBv1 we have

$$\begin{aligned} H_h(X, T) =& X_1 h^{m+p+1} \oplus X_2 h^{m+p} \oplus \ldots \oplus X_m h^{p+2} \oplus T_1 h^{p+1} \\ & \oplus T_2 h^p \oplus \ldots \oplus T_p h^2 \oplus (\mathsf{bin}_{\frac{n}{2}}(|X|)||\mathsf{bin}_{\frac{n}{2}}(|T|))h \end{aligned}$$

and in case of XCBv2 we have

$$\begin{aligned} H'_h(T, X) =& T_1 h^{p+m+1} \oplus T_2 h^{p+m} \oplus \ldots \oplus T_p h^{m+2} \oplus X_1 h^{m+1} \\ & \oplus X_2 h^m \oplus \ldots \oplus X_m h^2 \oplus (\mathsf{bin}_{\frac{n}{2}}(|T|)||\mathsf{bin}_{\frac{n}{2}}(|X|))h. \end{aligned}$$

Let X_i be equal to h and $X_{i-1} = 1$ for $2 \leq i \leq m$, $X_j = 0$, $j \neq i$; then

$$H_h(X, 0^n) = (\mathsf{bin}_{\frac{n}{2}}(|X|)||\mathsf{bin}_{\frac{n}{2}}(n))h$$

and

$$H'_h(0^n, X) = (\mathsf{bin}_{\frac{n}{2}}(n)||\mathsf{bin}_{\frac{n}{2}}(|X|))h.$$

Let $X' = 0^{|X|}$, we have

$$H_h(X', 0^n) = (\mathsf{bin}_{\frac{n}{2}}(|X|)||\mathsf{bin}_{\frac{n}{2}}(n))h$$

and

$$H'_h(0^n, X') = (\mathsf{bin}_{\frac{n}{2}}(n)||\mathsf{bin}_{\frac{n}{2}}(|X|))h.$$

As in the case of HCTR the attack would consist of the queries $\mathsf{KDFn}(0^n, X)$ and $\mathsf{Fn}(0^n, X')$. Obtaining the key h as a result of this attack.

$\underline{F(K,T,M)}$
$\quad \lambda \leftarrow E(K,\gamma,\alpha)$
$\quad$ **if** $M = K$ **then** $Y \leftarrow E(K,T,\lambda)$
$\quad$ **else if** $M = \lambda$ **then** $Y \leftarrow E(K,T,K)$
$\quad$ **else** $Y \leftarrow E(K,T,M)$
$\quad$ **return** Y

$\underline{F^{-1}(K,T,C)}$
$\quad Y \leftarrow E^{-1}(K,T,C)$
$\quad \lambda \leftarrow E(K,\gamma,\alpha)$
$\quad$ **if** $Y = K$ **then** $M \leftarrow \lambda$
$\quad$ **else if** $Y = \lambda$ **then** $M \leftarrow K$
$\quad$ **return** M

Figure 5.1: Swap then Encipher transform.

5.3 Ciphers that Securely Encipher their own Keys

Bellare, Cash and Keelveedhi [BCK11] in 2011 proposed two transformations in order to turn a secure tweakable SPRP into a secure tweakable SPRP for KDM. One transformation is performed in the narrow block case, and the other one is performed in the wide block case. Let $E : \mathcal{K} \times \mathcal{T} \times \{0,1\}^{mn} \to \{0,1\}^{mn}$ be a TES, we say that the E is narrow block if $m = 1$ and wide block if $m > 1$.

5.3.1 KDM security for narrow block case

In narrow block case, Bellare, Cash and Keelveedhi provide a transform **StE** (Swap then Encipher) that turns a given secure TES into a secure {id}-KDM TES. Where {id} denotes the identity function on the key, i.e, the transformation guarantees the security of the TES when the only allowed key dependent message is equal to the key.

The idea of the transformation is to swap the key for a "hidden point" (λ in Fig. 5.1), determined by encryption of a constant under the key. The hidden point is defined via a tweak not used for anything else.

Given a TES $E : \{0,1\}^n \times \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ assumed to have SPRP security. We pick an arbitrary $\gamma \in \mathcal{T}$ and also an arbitrary point $\alpha \in \{0,1\}^n$. Both α and γ are public and known to the adversary. The **StE** transform, described in Fig. 5.1, turns E into another TES $F : \{0,1\}^n \times \mathcal{T} \setminus \{\gamma\} \times \{0,1\}^n \to \{0,1\}^n$ whose tweakspace $\mathcal{T} \setminus \{\gamma\}$ is that of E with γ removed, meaning γ is not allowed as a tweak for F. The following theorem gives the security for **StE** transform.

Theorem 5.3.1 ([BCK11])**.** *Let $E : \{0,1\}^n \times \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ be a tweakable block cipher. Let $\gamma \in \mathcal{T}$ and $\alpha \in \{0,1\}^n$. Let $F = \mathbf{StE}_{\gamma,\alpha}[E] : \{0,1\}^n \times \mathcal{T} \setminus \{\gamma\} \times \{0,1\}^n \to \{0,1\}^n$ be a tweakable block cipher associated to E via the $\mathbf{StE}_{\gamma,\alpha}$ transform as defined above. Let $\Phi = \{\mathrm{id}\}$ consist of the identity function. Let A be an adversary making at most q oracle queries. Then there exists an adversary B such that*

$$\mathbf{Adv}^{\pm\widetilde{\mathrm{prp}}}_{F,\Phi}(A) < 2 \cdot \mathbf{Adv}^{\pm\widetilde{\mathrm{prp}}}_{E}(B) + \frac{3q}{2^n - 1}.$$

$\underline{J(K,T,M)}$	$\underline{J^{-1}(K,T,Y)}$
$X \leftarrow \mathrm{ECB}[F(K,\gamma_1,\cdot)](M)$	$X \leftarrow \mathbf{E}^{-1}[F^{-1}(K,\gamma_2,\cdot)](T,Y)$
$Y \leftarrow \mathbf{E}[F(K,\gamma_2,\cdot)](T,X)$	$M \leftarrow \mathrm{ECB}[F^{-1}(K,\gamma_1,\cdot)](X)$
return Y	**return** X

Figure 5.2: ECB then Encipher transform.

5.3.2 KDM security for wide block case

EtE (ECB then Encipher) transform turns a secure SPRP TES into a secure $\mathcal{ID}_m$-KDM TES. Where $\mathcal{ID}_m$ refers to the class of functions which when given K as input returns a message where one or more blocks contain the key. The key can be present in multiple blocks but not overlapping between blocks.

We formally define the class $\mathcal{ID}_m$ next.

The class $\mathcal{ID}_m$. We formally define the class capturing occurrence of the key in any block of the message. Let $\mathcal{ID}_m$ be the class of functions which maps an n-bit key to a message of m blocks. Each function id_M in $\mathcal{ID}_m$ is indexed by M, where M is a vector with m components where each component is either in $\{0,1\}^n$ or is $\perp$. Let $M = (M_1, M_2, \ldots, M_m)$ where either $M_i \in \{0,1\}^n$ or $M_i = \perp$. Hence $\mathrm{id}_M : \{0,1\}^n \to \{0,1\}^{mn}$ maps the input key to m many n-bit blocks in the following manner:

$$\mathrm{id}_M(K) = (M'_1, M'_2, \ldots, M'_m)$$

where

$$M'_i = \begin{cases} M_i \text{ if } M_i \in \{0,1\}^n \\ K \text{ if } M_i = \perp \end{cases}$$

EtE makes an ECB pass through the data using $F = \mathbf{StE}[E]$ under one tweak. Then applies a secure wide block SPRP TES $\mathbf{E}[F]$ with another tweak. By $\mathbf{E}[F]$ we denote TES $\mathbf{E}$ using F as underlying block cipher. **EtE** transform associates to $\mathbf{E}$ and F a tweakable wide block cipher $J = \mathbf{EtE}[\mathbf{E}, F]$ where $J : \{0,1\}^n \times \mathcal{T} \times \{0,1\}^{mn} \to \{0,1\}^{mn}$. Fig. 5.2 describes **EtE** transform. The following theorem gives the security for **EtE** transform.

Theorem 5.3.2 ([BCK11]). *Let $\mathbf{E}$ be a wide block tweakable block cipher, with tweakspace $\mathcal{T}$, blocklength n and number of blocks m. Let $F : \{0,1\}^n \times \{\gamma_1, \gamma_2\} \times \{0,1\}^n \to \{0,1\}^n$ be a secure $\{\mathrm{id}\}$-KDM narrow tweakable block cipher. Let $J = \mathbf{EtE}[\mathbf{E}, F]$ be the wide block tweakable cipher associated to $\mathbf{E}$ and F via the **EtE** transform as defined above. Let $\Phi = \mathcal{ID}_m$. Let A be an adversary making at most q oracle queries with the tweak argument in each query having length at most nt. Then*

there is an adversary B such that

$$\mathbf{Adv}^{\pm\widetilde{\text{prp}}}_{J,\Phi}(A) < 2 \cdot \mathbf{Adv}^{\pm\widetilde{\text{prp}}}_{F,\{\text{id}\}}(B) + 2 \cdot \mathbf{Adv}^{\pm\widetilde{\text{prp}}}_{E}(q,n,m,t) + \frac{2q^2(m^2+2)}{2^{n+2}}.$$

5.3.3 Drawbacks of StE and EtE

StE and EtE transformations consider that the underlying scheme uses only one key. This can be inconvenient because it limits the use of the transformations to those TES where there is only a single key. TES with this feature are the minority of the all known TES.

For example, one can imagine that if we apply StE transformation to LRW then it will be KDM secure. The problem is that LRW uses two keys and it is not clear which key is going to be swapped with the hidden point. A natural choice could be to pick the key of the underlying block cipher to be swapped. We show that if we do so the scheme remains KDM insecure.

Recall from Section 5.2.1 the LRW construction when instantiated with $h_{K_2}(T) = TK_2$, where $T, K_2 \in GF(2^n)$.

$$\tilde{E}_{K_1,K_2}(T,M) = E_{K_1}(M \oplus h_{K_2}(T)) \oplus h_{K_2}(T).$$

If we apply StE to LRW we obtain the following scheme F

$\underline{F_{K_1,K_2}(T,M)}$
$H \leftarrow \tilde{E}_{K_1,K_2}(\gamma,\alpha)$
if $M = K_1$ then $Y \leftarrow \tilde{E}_{K_1,K_2}(T,H)$
else if $M = H$ then $Y \leftarrow \tilde{E}_{K_1,K_2}(T,K_1)$
else $Y \leftarrow \tilde{E}_{K_1,K_2}(T,M)$
return Y

Then F is not KDM-secure with respect to the function $g(K_1,K_2) = K_2$. We use the same attack presented in Section 5.2.1. The adversary queries $\mathsf{KDFn}(0,g)$ (i.e, tweak value 0 and plaintext K_2) and also $\mathsf{Fn}(1,0)$ (i.e, tweak value 1 and plaintext 0), thus getting

$$c_1 = F_{K_1,K_2}(0,K_2) = E_{K_1}(K_2) \text{ and } c_2 = F_{K_1,K_2}(1,0) = E_{K_1}(K_2) \oplus K_2.$$

Next the adversary computes $K_2 = c_1 \oplus c_2$ and then verify this value by asking to decrypt the value of $c_1 \oplus 2K_2$ with respect to the tweak value 2.

In case of EtE it is not easy to see an attack when we apply this transformation to a TES which uses more than one key. The latter is due to the ECB layer added by EtE. Despite this fact, we must remark that the design of EtE and its security proof consider that the underlying TES uses only one key.

In the following chapter we show a transformation which solves this problem of StE and EtE. This transformation considers that the underlying TES uses a set $\mathbf{K}$ of keys.

6. MStE: A Generic KDM Secure TES

> Few false ideas have more firmly gripped the minds of so many intelligent men than the one that, if they just tried, they could invent a cipher that no one could break.
>
> The Codebreakers, David Kahn

In this chapter we present a transformation that turns a SPRP-secure TES into a KDM-secure TES. We call this transformation as Meta Swap-then-Encipher (MStE), it receives its name because it first swaps the underlying secret keys of the TES and then it continues with the enciphering procedure as specified by the TES.

First in Section 6.1 we show the construction of MStE, then in Section 6.2 we present the proof of security for MStE and finally in Section 6.3 we compare MStE with EtE and StE.

6.1 Construction of MStE

MStE is based on the work of Bellare, Cash and Keelveedhi [BCK11], particularly in their Swap-then-Encipher (StE) transformation. MStE converts a tweakable block cipher into another one which can securely encipher its own key.

As stated by Bellare et al. [BCK11], the idea behind StE is a suggestion of Boneh, Halevi, Hamburg and Ostrovsky [BHHO08] to securely encrypt keys with a *randomized* encryption scheme by exchanging the key with another point. To make this work for deterministic encryption the key is exchanged with a "hidden" point determined by encryption of a constant under the key. A crucial idea is to use tweaks, defining the hidden point via a tweak which is not used for anything else.

We consider a set $\mathbf{K}$ of underlying keys (we think of a set of keys because the assumption that there is just a single key would seem to entail a loss of generality). Let $\mathbf{K} = \{K_1, K_2, \ldots, K_p\}$ be a set of keys which consists of p many different elements. By $\mathsf{siz}(\mathbf{K})$ we denote the sum of the lengths of each element of $\mathbf{K}$, such that $\mathsf{siz}(\mathbf{K}) =$

$|K_1| + |K_2| + \ldots + |K_p|$. for the scheme we assume $|K_1| = |K_2| = \cdots = |K_p| = n$ where n is the block length of the block cipher.

Given a SPRP-secure TES $\mathbf{E} : \mathcal{K} \times \mathcal{T} \times \mathcal{M} \to \mathcal{M}$, MStE first picks an arbitrary tweak $\gamma \in \mathcal{T}$ and also an arbitrary point $\alpha \in \{0,1\}^{\mathsf{siz}(\mathbf{K})}$. The length of α is determined by the sum of the lengths of the underlying keys used by $\mathbf{E}$. Both α and γ are public and known to the adversary. MStE transformation turns $\mathbf{E}$ into another TES $\mathbf{F} : \mathcal{K} \times \mathcal{T} \setminus \{\gamma\} \times \mathcal{M} \to \mathcal{M}$ whose tweak space is $\mathcal{T} \setminus \{\gamma\}$, meaning γ is not allowed as a tweak for $\mathbf{F}$.

Encryption and decryption operations using MStE are described in Fig. 6.1. In Fig. 6.1 the sets $\boldsymbol{\lambda}$ and $\mathbf{K}$ have the same number of elements and $\mathsf{siz}(\boldsymbol{\lambda}) = \mathsf{siz}(\mathbf{K})$.

Encryption$^T_{\mathbf{K}}(P_1, \ldots, P_m)$
1. $\lambda_1 || \lambda_2 || \ldots || \lambda_p \leftarrow \mathbf{E}^{\gamma}_{\mathbf{K}}(\alpha)$; $\boldsymbol{\lambda} = \{\lambda_1, \lambda_2, \ldots, \lambda_p\}$
2. **for** $i = 1$ to m,
3. **if** $P_i = K_j$ for some $K_j \in \mathbf{K}$ **then**
4. $P_i \leftarrow \lambda_j$
5. **else if** $P_i = \lambda_j$ for some $\lambda_j \in \boldsymbol{\lambda}$ **then**
6. $P_i \leftarrow K_j$
7. **end if**
8. **end for**
9. $C_1, \ldots, C_m \leftarrow \mathbf{E}^T_{\mathbf{K}}(P_1, \ldots, P_m)$
10. **return** $C_1, \ldots, C_m$

Decryption$^T_{\mathbf{K}}(C_1, \ldots, C_m)$
$\lambda_1 || \lambda_2 || \ldots || \lambda_p \leftarrow \mathbf{E}^{\gamma}_{\mathbf{K}}(\alpha)$; $\boldsymbol{\lambda} = \{\lambda_1, \lambda_2, \ldots, \lambda_p\}$
$P_1, \ldots, P_m \leftarrow \mathbf{E}^{T\,-1}_{\mathbf{K}}(C_1, \ldots, C_m)$
for $i = 1$ to m,
if $P_i = K_j$ for some $K_j \in \mathbf{K}$ **then**
$P_i \leftarrow \lambda_j$
else if $P_i = \lambda_j$ for some $\lambda_j \in \boldsymbol{\lambda}$ **then**
$P_i \leftarrow K_j$
end if
end for
return $P_1, \ldots, P_m$

Figure 6.1: Encryption and decryption using MStE[$\mathbf{E}$], $\mathbf{E}$ is a TES, $\mathbf{K}$ is a set of keys, $T \in \mathcal{T} \setminus \{\gamma\}$ the tweak, and α is a public parameter.

6.2 Security of MStE

MStE is $\mathcal{J}_m$-KDM secure. The class $\mathcal{J}_m$ is similar to the class $\mathcal{ID}_m$ discussed in Section 5.3.2. First we define the class $\mathcal{J}_m$.

The class $\mathcal{J}_m$: Let $\mathcal{J}_m$ be the class of functions which maps p many n bit keys to a message of m blocks. Each function id_P in $\mathcal{J}_m$ is indexed by P, where P is a vector with m components where each component is either in $\{0,1\}^n$ or in $\{\perp_1, \perp_2, \ldots, \perp_p\}$. Let $P = (P_1, P_2, \ldots, P_m)$ where either $P_i \in \{0,1\}^n$ or $P_i = \perp_j$ for $1 \leq j \leq p$. Hence $\mathrm{id}_P : \{0,1\}^{pn} \to \{0,1\}^{mn}$ maps p keys to m many n bit blocks in the following manner:

$$\mathrm{id}_P(K_1, \ldots, K_p) = (P'_1, P'_2, \ldots, P'_m)$$

where

$$P'_i = \begin{cases} P_i \text{ if } P_i \in \{0,1\}^n \\ K_j \text{ if } P_i = \perp_j,\ 1 \leq j \leq p \end{cases}$$

The following theorem specifies the security of MStE.

Theorem 6.2.1. *Fix n, σ to be positive integers. Let $\mathbf{E}$ be a tweakable enciphering scheme. Let $F = \mathrm{MStE}[\mathbf{E}]$ be the tweakable block cipher associated to $\mathbf{E}$ via the* MStE *transform as defined above. Then*

$$\mathbf{Adv}^{\pm\widetilde{\mathrm{prp}}}_{\mathrm{MStE}[\mathbf{E}],\mathcal{J}_m}(\sigma) \leq 2 \cdot \mathbf{Adv}^{\pm\widetilde{\mathrm{prp}}}_{\mathbf{E}}(\sigma) + \frac{p\sigma}{2^{n-1}}, \tag{6.1}$$

where p is the number of keys in the set $\mathbf{K}$.

6.2.1 Proof of theorem 6.2.1

The discussion in this section is based on [BCK11]. We prove Theorem 6.2.1 by modeling the interaction of an adversary with MStE by a sequence of games. We bound the advantage of an adversary in distinguishing between games G_0 and G_1. Game G_0 describes MStE and game G_1 represents a random tweak index permutation.

The advantage of an adversary in distinguishing $F = \mathrm{MStE}[\mathbf{E}]$ is defined in the following manner

$$\begin{aligned} \mathbf{Adv}^{\pm\widetilde{\mathrm{prp}}}_{F,\mathcal{J}_m}(A) \;=\; & \Pr\left[K \xleftarrow{\$} \{0,1\}^n : A^{F_K(\cdot,\cdot),F_K^{-1}(\cdot,\cdot),\mathsf{KDFn}_{F_K}(\cdot,\mathrm{id}_P)} \Rightarrow 1\right] \\ & - \Pr\left[\boldsymbol{\pi} \xleftarrow{\$} \mathrm{Perm}^{\mathcal{T}}(\mathcal{M}) : A^{\boldsymbol{\pi}(\cdot,\cdot),\boldsymbol{\pi}^{-1}(\cdot,\cdot),\mathsf{KDFn}_{\boldsymbol{\pi}}(\cdot,\mathrm{id}_P)} \Rightarrow 1\right], \end{aligned} \tag{6.2}$$

where $\mathrm{id}_P \in \mathcal{J}_m$.

Consider the games in Fig. 6.2. Games G_0 and G_1 accurately represent the following:

$$\Pr[A^{G_0}] = \Pr\left[K \xleftarrow{\$} \{0,1\}^n : A^{F_K(\cdot,\cdot),F_K^{-1}(\cdot,\cdot),\mathsf{KDFn}_{F_K}(\cdot,\mathrm{id}_P)} \Rightarrow 1\right].$$

$$\Pr[A^{G_1}] = \Pr\left[\boldsymbol{\pi} \xleftarrow{\$} \mathrm{Perm}^{\mathcal{T}}(\mathcal{M}) : A^{\boldsymbol{\pi}(\cdot,\cdot),\boldsymbol{\pi}^{-1}(\cdot,\cdot),\mathsf{KDFn}_{\boldsymbol{\pi}}(\cdot,\mathrm{id}_P)} \Rightarrow 1\right].$$

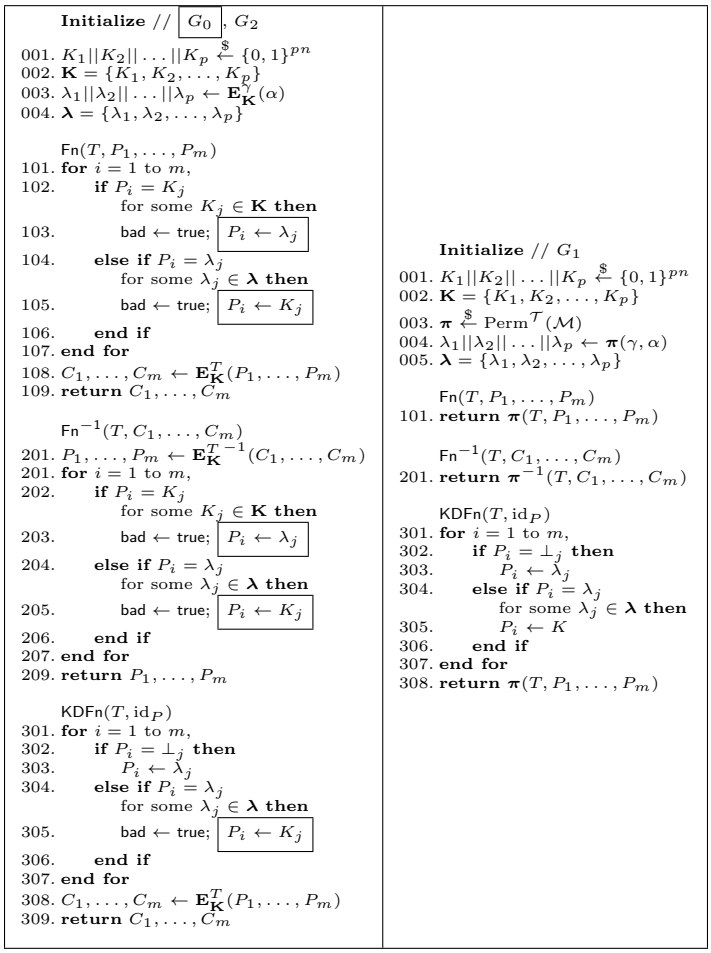

Figure 6.2: Games G_0, G_1 and G_2. In G_2 the boxed entries are removed.

Thus

$$\begin{aligned}\mathbf{Adv}^{\pm\widetilde{\mathrm{prp}}}_{F,\mathcal{J}_m}(A) &= \Pr\left[A^{G_0} \Rightarrow 1\right] - \Pr\left[A^{G_1} \Rightarrow 1\right] \\ &= \left(\Pr\left[A^{G_2} \Rightarrow 1\right] - \Pr\left[A^{G_1} \Rightarrow 1\right]\right) \\ &\quad + \left(\Pr\left[A^{G_0} \Rightarrow 1\right] - \Pr\left[A^{G_2} \Rightarrow 1\right]\right) \\ &\leq \left(\Pr\left[A^{G_2} \Rightarrow 1\right] - \Pr\left[A^{G_1} \Rightarrow 1\right]\right) + \Pr[A^{G_2} \text{ sets } \mathsf{bad}]. \quad (6.3)\end{aligned}$$

The inequality is by the fundamental lemma of game playing since G_0 and G_2 are identical until bad. Game G_2 does not include the boxed entries.

For bounding Eq. (6.3) we use the following Claims which we shall prove later in Section 6.2.1.

Claim 6.2.1. *There exists an adversary B_1 for $\mathbf{E}$ such that*

$$\Pr\left[A^{G_2} \Rightarrow 1\right] - \Pr\left[A^{G_1} \Rightarrow 1\right] \leq \mathbf{Adv}^{\pm\widetilde{\mathrm{prp}}}_{\mathbf{E}}(B_1). \quad (6.4)$$

From Eq. (6.3) and Claim 6.2.1 we have that

$$\mathbf{Adv}^{\pm\widetilde{\mathrm{prp}}}_{F,\mathcal{J}_m}(A) \leq \mathbf{Adv}^{\pm\widetilde{\mathrm{prp}}}_{\mathbf{E}}(B_1) + \Pr[A^{G_2} \text{ sets } \mathsf{bad}]. \quad (6.5)$$

Next we bound $\Pr[A^{G_2} \text{ sets } \mathsf{bad}]$, considering separately the probability of setting bad_1 and bad_2 in game G_3 of Fig. 6.3.

$$\Pr\left[A^{G_2} \text{ sets } \mathsf{bad}\right] \leq \Pr\left[A^{G_3} \text{ sets } \mathsf{bad}_1\right] + \Pr\left[A^{G_3} \text{ sets } \mathsf{bad}_2\right]. \quad (6.6)$$

For bounding $\Pr\left[A^{G_3} \text{ sets } \mathsf{bad}_1\right]$, we observe that each element of $\mathbf{K}$ is chosen uniformly at random, so

$$\Pr\left[A^{G_3} \text{ sets } \mathsf{bad}_1\right] \leq \frac{p\sigma}{2^n}. \quad (6.7)$$

For bounding $\Pr\left[A^{G_3} \text{ sets } \mathsf{bad}_2\right]$, we use the following Claim.

Claim 6.2.2. *There exists an adversary B_2 for $\mathbf{E}$ such that*

$$\Pr\left[A^{G_3} \textit{ sets } \mathsf{bad}_2\right] \leq \mathbf{Adv}^{\pm\widetilde{\mathrm{prp}}}_{\mathbf{E}}(B_2) + \frac{p\sigma}{2^n}. \quad (6.8)$$

From Eq. (6.5), Eq. (6.6), Eq. (6.7) and Claim 6.2.2 we obtain

$$\mathbf{Adv}^{\pm\widetilde{\mathrm{prp}}}_{\mathrm{MStE}[\mathbf{E}],\mathcal{J}_m}(\sigma) \leq 2 \cdot \mathbf{Adv}^{\pm\widetilde{\mathrm{prp}}}_{\mathbf{E}}(\sigma) + \frac{p\sigma}{2^{n-1}},$$

as desired. We are thus left with proving Claims 6.2.1 and 6.2.2 which we do next.

Proofs of claims 6.2.1 and 6.2.2

Proof of Claim 6.2.1. We design an adversary B_1 so that

$$\Pr\left[A^{G_2} \Rightarrow 1\right] - \Pr\left[A^{G_1} \Rightarrow 1\right] \leq \mathbf{Adv}_{\mathbf{E}}^{\pm\widetilde{\mathrm{prp}}}(B_1).$$

Adversary B_1 is an adversary for TES $\mathbf{E}$. Assume that B_1 has as its oracles O and O^{-1}. It runs A in the following manner. First it starts by letting $\lambda_1||\lambda_2||\ldots||\lambda_p \leftarrow O_(\gamma, \alpha)$. When A makes a query (T, P) to its Fn oracle, B_1 queries its oracle O with (T, P) and forwards the response to A. Similarly when A makes a query (T, C) to its Fn^{-1}, B_1 queries its O^{-1} oracle with (T, C) and forwards the response to A. When A queries its KDFn oracle with (T, id_P) adversary B_1 does the following:

for $i = 1$ to m,
  if $P_i = \perp_j$ **then**
    $P_i \leftarrow \lambda_j$
  end if
end for
return $\mathrm{O}(T, P_1, \ldots, P_m)$

When A halts with output b'. B_1 also halts and outputs b'. We have

$$\Pr\left[K \xleftarrow{\$} \{0,1\}^n : B_1^{\mathbf{E}_K(\cdot,\cdot),\mathbf{E}_K^{-1}(\cdot,\cdot)} \Rightarrow 1\right] = \Pr[A^{G_2} \Rightarrow 1]$$
$$\Pr\left[\boldsymbol{\pi} \xleftarrow{\$} \mathrm{Perm}^{\mathcal{T}}(\mathcal{M}) : B_1^{\boldsymbol{\pi}(.,.),\boldsymbol{\pi}^{-1}(.,.)} \Rightarrow 1\right] = \Pr[A^{\mathrm{G}_1} \Rightarrow 1].$$

We recall that

$$\begin{aligned}\mathbf{Adv}_{\mathbf{E}}^{\pm\widetilde{\mathrm{prp}}}(B_1) &= \Big|\Pr\left[K \xleftarrow{\$} \mathcal{K} : A^{\mathbf{E}_K(.,.),\mathbf{E}_K^{-1}(.,.)} \Rightarrow 1\right] \\ &\quad - \Pr\left[\boldsymbol{\pi} \xleftarrow{\$} \mathrm{Perm}^{\mathcal{T}}(\mathcal{M}) : A^{\boldsymbol{\pi}(.,.),\boldsymbol{\pi}^{-1}(.,.)} \Rightarrow 1\right]\Big|.\end{aligned}$$

So, we obtain

$$\Pr\left[A^{G_2} \Rightarrow 1\right] - \Pr\left[A^{G_1} \Rightarrow 1\right] \leq \mathbf{Adv}_{\mathbf{E}}^{\pm\widetilde{\mathrm{prp}}}(B_1).$$

□

Proof of Claim 6.2.2. As in the previous proof B_2 is an adversary for $\mathbf{E}$. We design adversary B_2 which has oracles O and O^{-1}. It starts by letting $\lambda_1||\lambda_2||\ldots||\lambda_p \leftarrow O(\gamma, \alpha)$ and $\boldsymbol{\lambda} = \{\lambda_1, \lambda_2, \ldots, \lambda_p\}$. It then runs A. When A makes a query (T, P) to its Fn oracle, B_2 does the following:

Initialize // G_3	**Initialize** // G_4
001. $K_1 \| \| K_2 \| \| \ldots \| \| K_p \xleftarrow{\$} \{0,1\}^{pn}$	001. $\boldsymbol{\pi} \xleftarrow{\$} \mathrm{Perm}^{\mathcal{T}}(\mathcal{M})$
002. $\mathbf{K} = \{K_1, K_2, \ldots, K_p\}$	002. $\lambda_1 \| \| \lambda_2 \| \| \ldots \| \| \lambda_p \leftarrow \boldsymbol{\pi}(\gamma, \alpha)$
003. $\lambda_1 \| \| \lambda_2 \| \| \ldots \| \| \lambda_p \leftarrow \mathbf{E}^{\gamma}_{\mathbf{K}}(\alpha)$	003. $\boldsymbol{\lambda} = \{\lambda_1, \lambda_2, \ldots, \lambda_p\}$
004. $\boldsymbol{\lambda} = \{\lambda_1, \lambda_2, \ldots, \lambda_p\}$	
$\mathsf{Fn}(T, P_1, \ldots, P_m)$	$\mathsf{Fn}(T, P_1, \ldots, P_m)$
101. **for** $i = 1$ to m,	101. **for** $i = 1$ to m,
102. **if** $P_i = K_j$	102. **if** $P_i = \lambda_j$
for some $K_j \in \mathbf{K}$ **then**	for some $\lambda_j \in \boldsymbol{\lambda}$ **then**
103. $\mathsf{bad}_1 \leftarrow \mathsf{true}$	103. $\mathsf{bad}_2 \leftarrow \mathsf{true}$
104. **else if** $P_i = \lambda_j$	104. **end if**
for some $\lambda_j \in \boldsymbol{\lambda}$ **then**	105. **end for**
105. $\mathsf{bad}_2 \leftarrow \mathsf{true}$	106. $C_1, \ldots, C_m \leftarrow \boldsymbol{\pi}(T, P_1, \ldots, P_m)$
106. **end if**	107. **return** $C_1, \ldots, C_m$
107. **end for**	
108. $C_1, \ldots, C_m \leftarrow \mathbf{E}^{T}_{\mathbf{K}}(P_1, \ldots, P_m)$	
109. **return** $C_1, \ldots, C_m$	
$\mathsf{Fn}^{-1}(T, C_1, \ldots, C_m)$	$\mathsf{Fn}^{-1}(T, C_1, \ldots, C_m)$
201. $P_1, \ldots, P_m \leftarrow \mathbf{E}^{T^{-1}}_{\mathbf{K}}(C_1, \ldots, C_m)$	201. $P_1, \ldots, P_m \leftarrow \boldsymbol{\pi}^{-1}(T, C_1, \ldots, C_m)$
201. **for** $i = 1$ to m,	202. **for** $i = 1$ to m,
202. **if** $P_i = K_j$	203. **if** $P_i = \lambda_j$
for some $K_j \in \mathbf{K}$ **then**	for some $\lambda_j \in \boldsymbol{\lambda}$ **then**
203. $\mathsf{bad}_1 \leftarrow \mathsf{true}$	204. $\mathsf{bad}_2 \leftarrow \mathsf{true}$
204. **else if** $P_i = \lambda_j$	205. **end if**
for some $\lambda_j \in \boldsymbol{\lambda}$ **then**	206. **end for**
205. $\mathsf{bad}_2 \leftarrow \mathsf{true}$	207. **return** $P_1, \ldots, P_m$
206. **end if**	
207. **end for**	
209. **return** $P_1, \ldots, P_m$	
$\mathsf{KDFn}(T, \mathrm{id}_P)$	$\mathsf{KDFn}(T, \mathrm{id}_P)$
301. **for** $i = 1$ to m,	301. **for** $i = 1$ to m,
302. **if** $P_i = \perp_j$ **then**	302. **if** $P_i = \perp_j$ **then**
303. $P_i \leftarrow \lambda_j$	303. $P_i \leftarrow \lambda_j$
304. **else if** $P_i = \lambda_j$	204. **else if** $P_i = \lambda_j$
for some $\lambda_j \in \boldsymbol{\lambda}$ **then**	for some $\lambda_j \in \boldsymbol{\lambda}$ **then**
305. $\mathsf{bad}_2 \leftarrow \mathsf{true}$	305. $\mathsf{bad}_2 \leftarrow \mathsf{true}$
306. **end if**	306. **end if**
307. **end for**	307. **end for**
308. $C_1, \ldots, C_m \leftarrow \mathbf{E}^{T}_{\mathbf{K}}(P_1, \ldots, P_m)$	308. $C_1, \ldots, C_m \leftarrow \boldsymbol{\pi}(T, P_1, \ldots, P_m)$
309. **return** $C_1, \ldots, C_m$	309. **return** $C_1, \ldots, C_m$

Figure 6.3: Games G_3 and G_4.

for $i = 1$ to m,
  if $P_i = \lambda_j$ for some $\lambda_j \in \boldsymbol{\lambda}$ **then**
    $\mathsf{bad}_2 \leftarrow \mathsf{true}$
  end if
end for
$C_1, \ldots, C_m \leftarrow O(T, P_1, \ldots, P_m)$
return $C_1, \ldots, C_m$

Similarly when A makes a query (T, C) to its Fn^{-1} oracle, B_2 does the following:

$P_1, \ldots, P_m \leftarrow O^{-1}(T, C_1, \ldots, C_m)$
for $i = 1$ to m,
  if $P_i = \lambda_j$ for some $\lambda_j \in \boldsymbol{\lambda}$ **then**
    $\mathsf{bad}_2 \leftarrow \mathsf{true}$
  end if
end for
return $P_1, \ldots, P_m$

When A queries its KDFn oracle with (T, id_P), adversary B_2 does the following:

for $i = 1$ to m,
  if $P_i = \perp_j$ **then**
    $P_i \leftarrow \lambda_j$
  else if $P_i = \lambda_j$ for some $\lambda_j \in \boldsymbol{\lambda}$ **then**
    $\mathsf{bad}_2 \leftarrow \mathsf{true}$
  end if
end for
$C_1, \ldots, C_m \leftarrow O(T, P_1, \ldots, P_m)$
return $C_1, \ldots, C_m$

When A halts with output b', adversary B_2 halts with output 1 if bad_2 has value true and 0 otherwise. Considering the games in Fig. 6.3 we have

$$\begin{aligned}\Pr\left[K \xleftarrow{\$} \{0,1\}^n : B_2^{\mathbf{E}_K(\cdot,\cdot),\mathbf{E}_K^{-1}(\cdot,\cdot)} \Rightarrow 1\right] &= \Pr[A^{G_3} \text{ sets } \mathsf{bad}_2] \\ \Pr\left[\boldsymbol{\pi} \xleftarrow{\$} \mathrm{Perm}^{\mathcal{T}}(\mathcal{M}) : B_2^{\boldsymbol{\pi}(\cdot,\cdot),\boldsymbol{\pi}^{-1}(\cdot,\cdot)} \Rightarrow 1\right] &= \Pr[A^{\mathrm{G}_4} \text{ sets } \mathsf{bad}_2]. \qquad (6.9)\end{aligned}$$

Thus

$$\begin{aligned}\Pr[A^{\mathrm{G}_3} \text{ sets } \mathsf{bad}_2] &= \Pr[A^{\mathrm{G}_3} \text{ sets } \mathsf{bad}_2] - \Pr[A^{\mathrm{G}_4} \text{ sets } \mathsf{bad}_2] \\ &\quad + \Pr[A^{\mathrm{G}_4} \text{ sets } \mathsf{bad}_2] \\ &\leq \mathbf{Adv}_{\mathbf{E}}^{\widetilde{\pm\mathrm{prp}}}(B_2) + \Pr\left[A^{G_4} \text{ sets } \mathsf{bad}_2\right].\end{aligned}$$

For bounding $\Pr\left[A^{G_4} \text{ sets } \mathsf{bad}_2\right]$ we move to game $G5$ in Fig. 6.4, where $\lambda_1, \lambda_2, \ldots, \lambda_p$ are not referred to in replying to adversary oracle queries. Since γ is not in the tweak

space of MStE[**E**], in game $G5$ $\lambda_1, \lambda_2, \ldots, \lambda_p$ are chosen uniformly at random. Thus we have

$$\begin{aligned} \Pr[A^{\mathrm{G}_4} \text{ sets } \mathsf{bad}_2] &= \Pr[A^{\mathrm{G}_5} \text{ sets } \mathsf{bad}_2] \\ &\leq \frac{p\sigma}{2^n}. \end{aligned} \tag{6.10}$$

This completes the proof. □

6.3 Comparison

In this section we compare MStE with the transformations StE and EtE, the last two resulted of the work of Bellare et al. [BCK11]. We begin with MStE and StE. For an easier comparison in Fig. 6.5 we show the encryption procedures of MStE and StE.

StE was proposed for narrow block tweakable block cipher where the message is equal to one block, i.e, the message space $\mathcal{M} = \{0,1\}^n$. MStE is proposed for wide block tweakable block ciphers where the message is a bit string of arbitrary length, i.e, the message space $\mathcal{M} = \{0,1\}^*$. We can see MStE as a generalization of StE, if the underlying scheme uses only one key and we use the transformation for encrypting only one block then there is no difference between MStE and StE.

In Section 5.3.3 we show that when we apply StE to LRW the resulted scheme can remain KDM insecure. Here we show that the attack not longer holds when we apply MStE to LRW.

Recall from Section 5.2.1 the LRW construction when instantiated with $h_{K_2}(T) = TK_2$, $T, K_2 \in GF(2^n)$.

$$\tilde{E}_{K_1,K_2}(T, M) = E_{K_1}(M \oplus h_{K_2}(T)) \oplus h_{K_2}(T).$$

If we apply MStE to LRW we obtain the following scheme F.

$F^T_{K_1,K_2}(P)$
  $\boldsymbol{\lambda} \leftarrow \mathbf{E}^{\gamma}_{K_1,K_2}(\alpha)$
  if $P = K_1$ **then** $P \leftarrow \lambda_1$
  else if $P = \lambda_1$ **then** $P \leftarrow K_1$
  else if $P = K_2$ **then** $P \leftarrow \lambda_2$
  else if $P = \lambda_2$ **then** $P \leftarrow K_2$
  end if
  $C \leftarrow \mathbf{E}^{T}_{K_1,K_2}(P)$
  return C

The attack presented in Section 5.2.1 consisted of querying $\mathsf{KDFn}(0, \mathsf{g})$ where $g(K_1, K_2) = K_2$ and $\mathsf{Fn}(1, 0)$, getting

$$c_1 = F_{K_1,K_2}(0, K_2) = E_{K_1}(\lambda_2) \text{ and } c_2 = F_{K_1,K_2}(1, 0) = E_{K_1}(K_2) \oplus K_2.$$

Initialize // G_5
001. $\boldsymbol{\pi} \xleftarrow{\$} \mathrm{Perm}^{\mathcal{T}}(\mathcal{M})$
002. $S \leftarrow \emptyset$

$\mathsf{Fn}(T, P_1, \ldots, P_m)$
101. **for** $i = 1$ to m,
102. $\quad S \leftarrow S \cup \{P_i\}$
103. **end for**
104. $C_1, \ldots, C_m \leftarrow \boldsymbol{\pi}(T, P_1, \ldots, P_m)$
105. **return** $C_1, \ldots, C_m$

$\mathsf{Fn}^{-1}(T, C_1, \ldots, C_m)$
201. $P_1, \ldots, P_m \leftarrow \boldsymbol{\pi}^{-1}(T, C_1, \ldots, C_m)$
202. **for** $i = 1$ to m,
203. $\quad S \leftarrow S \cup \{P_i\}$
204. **end for**
205. **return** $P_1, \ldots, P_m$

$\mathsf{KDFn}(T, \mathrm{id}_P)$
301. **for** $i = 1$ to m,
302. $\quad$ **if** $P_i \neq \perp_j$ **then**
303. $\quad\quad S \leftarrow S \cup \{P_i\}$
304. $\quad$ **end if**
305. **end for**
306. $C_1, \ldots, C_m \leftarrow \boldsymbol{\pi}(T, P_1, \ldots, P_m)$
307. **return** $C_1, \ldots, C_m$

Finalize
401. **for** $i = 1$ to p,
402. $\quad \lambda_i \xleftarrow{\$} \{0,1\}^n$
403. $\quad$ **if** $\lambda_i \in S$ **then**
404. $\quad\quad \mathsf{bad}_2 \leftarrow \mathsf{true}$
405. $\quad$ **end if**
406. **end for**

Figure 6.4: Game G_5.

Encryption under $\mathbf{MStE}^T_{\mathbf{K}}(P_1, \ldots, P_m)$

1. $\boldsymbol{\lambda} \leftarrow \mathbf{E}^{\gamma}_{\mathbf{K}}(\alpha)$
2. **for** $i = 1$ to m,
3. **if** $P_i = K_j$ for some $1 \leq j \leq p$ s.t. $K_j \in \mathbf{K}$ **then**
4. $P_i \leftarrow \lambda_j$
5. **else if** $P_i = \lambda_j$ for some $1 \leq j \leq p$ s.t. $\lambda_j \in \boldsymbol{\lambda}$ **then**
6. $P_i \leftarrow K_j$
7. **end if**
8. **end for**
9. $C_1, \ldots, C_m \leftarrow \mathbf{E}^T_{\mathbf{K}}(P_1, \ldots, P_m)$
10. **return** $C_1, \ldots, C_m$

Encryption under $\mathbf{StE}^T_K(P)$

1. $H \leftarrow \mathbf{E}^{\gamma}_K(\alpha)$
2. **if** $P = K$ **then** $Y \leftarrow \mathbf{E}^T_K(H)$
3. **else if** $P = H$ **then** $Y \leftarrow \mathbf{E}^T_K(K)$
4. **else** $Y \leftarrow \mathbf{E}^T_K(P)$
5. **return** Y

Figure 6.5: Encryption using MStE and StE.

From c_1 and c_2 is not longer possible to obtain K_2. This is also true for similar attacks, this lies in the fact that it is not possible for the adversary to obtain the hidden points in $\boldsymbol{\lambda}$.

In [BCK11], Bellare et al. proposed an Encrypt-then-Encipher (EtE) transformation for TES. The conversion results in a TES $\mathbf{E} : \{0,1\}^n \times \mathcal{T} \times \{0,1\}^m \rightarrow \{0,1\}^m$, with the restriction that m must be a multiple of n. The latter is because of the ECB layer added at the beginning of the cipher. In MStE this is not the case, i.e., MStE can cipher arbitrary length messages.

The addition of the ECB layer in EtE impacts the performance of the TES. While in MStE it is sufficient to compute the $\boldsymbol{\lambda}$ set and then make the swap of the keys.

In both transformations, StE and EtE, it is only protected one secret key. MStE is more general in this sense because it considers a set $\mathbf{K}$ of underlying secret keys.

7. Implementation Results

> We reject: kings, presidents and voting. We believe in: rough consensus and running code.
>
> IETF, David D. Clark

In this chapter we show the results of implementing MStE. In Section 7.1 we give a brief description of the basic building blocks for TES. Then in Section 7.2 we give a description of the technology used for the implementation, Intel AES-NI. And then in Section 7.3 we continue with the implementation of HCTR, XCB and EME2, and the instantiation of MStE and EtE with those TES. We also show in Section 7.3.3 that MStE performs better than EtE.

7.1 Basic Building Blocks

The basic building blocks for TES are block ciphers. A TES can be instantiated with any block cipher. For our implementations we choose the Advanced Encryption Standard (AES) with 128 bit key. Other than the block cipher calls some TES uses some operators on a finite field. In particular XCB and HCTR require finite field multiplications for computation of the hash function, and EME2 requires a special operation called *xtimes*. In this section we briefly describe the required finite field operators and the way we implemented them. Also we give a brief description of the AES block cipher.

7.1.1 Field operations

We sometimes view an n-bit string as a polynomial in $GF(2^n)$. An n-bit string $a_{n-1} \dots a_1 a_0 \in \{0,1\}^n$ corresponds to a formal polynomial $A(x) = a_{n-1} + a_{n-2}x + \dots + a_1 x^{n-2} + a_0 x^{n-1}$.

Because it is possible to perform well defined operations on the elements of the field, encryption schemes often use the polynomial representation of bit strings to scramble the input messages.

Let $A(x), B(x) \in GF(2)[x]$ be elements of the field $GF(2^n)$. The common operations performed on the elements of the field are:

Karatsuba multiplication(A,B)
1. parse A as $(A_1||A_0)$ where A_1, A_0 are $64 - bit$ words
2. parse B as $(B_1||B_0)$ where B_1, B_0 are $64 - bit$ words
3. $(C_1||C0) \leftarrow A_1 B_1$
4. $(D_1||D_0) \leftarrow A_0 B_0$
5. $(E_1||E_0) \leftarrow (A_0 \oplus A_1)(B_0 \oplus B_1)$
6. $S \leftarrow C_1||(C_0 \oplus C_1 \oplus D_1 \oplus E_1)||(D_1 \oplus C_0 \oplus D_0 \oplus E_0)||D_0$
7. **return** S

Figure 7.1: Karatsuba multiplication.

- **Addition:** The addition is performed by xoring the string representations of $A(x)$ and $B(x)$.

- **Multiplication:** In order to perform the multiplication of two elements of the field, we must fix an n degree irreducible polynomial. For $n = 128$ we choose $p(x) = 1 + x + x^2 + x^7 + x^{128}$ as the irreducible polynomial, so the operation in this field is defined as $A(x)B(x) = A(x)B(x) \bmod p(x)$. In our implementation we do the multiplication using the Karatsuba technique. The specific algorithm that we use is presented in Fig. 7.1. The algorithm in Fig. 7.1 shows that the multiplication of two 128 bit strings is obtained by computing the multiplication of 64 bit strings. This specific partitioning is used because new Intel machines have a dedicated instruction to perform carry less multiplication of 64 bit strings. The result S produced by the algorithm in Fig. 7.1 can be 256 bit long, hence we need to reduce it with the irreducible polynomial. The reduction is performed using a method proposed by Gueron and Kounavis in [GK10]. The particularity of this reduction technique is that it uses just shifts and xor operations. The algorithm is presented in Fig. 7.2.

- **xtimes:** The operation xtimes is the multiplication of an element of the field times the polynomial representation in the field of the integer 2. Which will always have order $2^n - 1$ in the multiplicative group of $GF(2^n)$, meaning that $2, 2^2, 2^3, \ldots, 2^{2^n-1}$ are all distinct. In Fig. 7.3 is given an easy way to compute this operation. In our implementations we use 128 bit SSE registers where performing a 1 bit shift is not possible. Hence we use the algorithm shown in Fig. 7.4 to achieve this.

7.1.2 Advanced encryption standard

The Advanced Encryption Standard (AES) is a specification for the encryption of electronic data adopted by the U.S. government in 2001. AES is a symmetric block cipher used worldwide and thus a key ingredient in the implementations of many cryptographic systems.

Fast Reduction modulo $p(x)$(A)
1. parse A as $(A_3||A_2||A_1||A_0)$ where A_3, A_2, A_1, A_0 are $64-bit$ words
2. $X_0 \leftarrow A_3 >> 63$
3. $X_1 \leftarrow A_3 >> 62$
4. $X_2 \leftarrow A_3 >> 57$
5. $X_3 \leftarrow A_2 \oplus X_0 \oplus X_1 \oplus X_2$
6. $(B_1||B_0) \leftarrow (A_3||X_3) << 1$
7. $(C_1||C_0) \leftarrow (A_3||X_3) << 2$
8. $(D_1||D_0) \leftarrow (A_3||X_3) << 7$
9. $(E_1||E_0) \leftarrow (A_3 \oplus B_1 \oplus C_1 \oplus D_1)||(X_3 \oplus B_0 \oplus C_0 \oplus D_0)$
10. $S \leftarrow (A_1 \oplus E_1)||(A_0 \oplus E_0)$
11. **return** S

Figure 7.2: Reduction modulo $p(x) = 1 + x + x^2 + x^7 + x^{128}$.

xtimes(A)
1. $b \leftarrow \mathsf{msb}(A)$
2. $A \leftarrow A << 1$
3. **if** $b = 1$
4. $\quad A \leftarrow A \oplus 0\mathtt{x}87$
5. **return** A

Figure 7.3: The xtimes operation.

xtimes(A)
1. parse A as $(A_3||A_2||A_1||A_0)$ where A_3, A_2, A_1, A_0 are 32-bit words, and $A_i = a_{i,31}a_{i,30}\ldots a_{i,0}$
2. $R \leftarrow \underline{a_{3,31}}||\underline{a_{2,31}}||\underline{a_{1,31}}||\underline{a_{3,31}}$ where $\underline{a}$ means a repeated 32 times. (R can be obtained from A by the instruction `PSRAD`)
3. $S \leftarrow \underline{a_{1,31}}||\underline{a_{1,31}}||\underline{a_{1,31}}||\underline{a_{3,31}}$ (S can be obtained from R by using `PSHUFD` instruction)
4. $S \leftarrow S \wedge (0\mathtt{x}00||0\mathtt{x}01||0\mathtt{x}00||0\mathtt{x}87)$
5. $S \leftarrow S \oplus [(A_3||A_2) << 1||(A_1||A_0) << 1]$
6. **return** S

Figure 7.4: The xtimes operation with 128 bit registers.

AES algorithm is based on the substitution-permutation design and it can process data blocks of 128 bits using cipher keys of length of 128, 192 and 256 bits. AES is described in FIPS-197 [FIP01].

AES operates on a 4×4 matrix of bytes (128 bits) termed the *state*. For the cipher procedure, AES uses a round function that is composed of four different transformations or functions:

- SubBytes: A non-linear substitution function where each byte is replaced with another according to a lookup table.
- ShiftRows: A transposition function where the i-th row of the state is shifted left circular by $i - 1$ bytes.
- MixColumns: A mixing function which operates on the columns of the state, combining the four bytes in each column.
- AddRoundKey: Each byte of the state is combined with the round key using bitwise xor.

The key size used in AES specifies the numbers of transformation rounds will be applied. The number of rounds are as follows.

- For the 128 bits key length 10 rounds are applied.
- For the 192 bits key length 12 rounds are applied.
- For the 256 bits key length 14 rounds are applied.

Depending of the key length AES algorithm is referred as AES-128, AES-192 and AES 256. AES uses a KeyExpansion routine to generate a series of round keys from the cipher key.

A high level description of the cipher procedure using AES-128 is shown in Fig. 7.5.

The cipher transformations can be inverted and then implemented in reverse order to produce a straight forward inverse cipher for AES. The following functions are defined in case of inverse cipher.

- InvMixColumns. Inverse of MixColumns.
- InvShiftRows. Inverse of ShiftRows.
- InvSubBytes. Inverse of SubBytes.

However, due to properties of the functions SubBytes, ShiftRows and MixColumns in the AES algorithm; it is possible to have an Equivalent Inverse Cipher which offers a more efficient structure than the straightforward inverse cipher. The Equivalent Inverse Cipher is defined by reversing the order of the InvSubBytes and InvShiftRows transformations, and by reversing the order of the AddRoundKey and InvMixColumns transformations after first modifying the decryption key schedule using the InvMixColumns transformation. The first and last round keys of the decryption key schedule are not modified in this manner. A high level description of the Equivalent Inverse Cipher procedure using AES-128 is shown in Fig. 7.6.

Cipher

1. KeyExpansion
2. AddRoundKey
3. **for** $round = 1$ to 9,
4. SubBytes
5. ShiftRows
6. MixColumns
7. AddRoundKey
8. **end for**
9. SubBytes
10. ShiftRows
11. AddRoundKey

Figure 7.5: Cipher procedure using AES-128.

Equivalent Inverse Cipher

1. KeyExpansion
   //apply InvMixColumns to the round keys
2. **for** $i = 1$ to 9,
3. MixColumns
4. **end for**
5. AddRoundKey
6. **for** $round = 1$ to 9,
7. InvSubBytes
8. InvShiftRows
9. InvMixColumns
10. AddRoundKey
11. **end for**
12. InvSubBytes
13. InvShiftRows
14. AddRoundKey

Figure 7.6: Equivalent inverse cipher procedure using AES-128.

7.2 Using Intel SIMD instructions

There exists a classification of computers architectures based on the number of concurrent instructions and data streams available. Most modern computers implements the Single Instruction Multiple Data (SIMD) instruction set. Using these instructions one can perform the same operation simultaneously on a set of independent data. This amounts to a certain type of parallelization which is called data level parallelism.

Since 1999, Intel's processors implement the set of instructions *Streaming SIMD Extensions* (SSE) which gives the processors the capacity to perform the same operation on multiple data. This was achieved by adding sixteen new registers of 128 bits. This registers can store one value of 128 bits, 2 values of 64 bits, 4 values of 32 bits, 8 values of 16 bits or 16 values of 8 bits. The instructions performed on these registers can be seen as vectorial instructions, as these instructions are performed on consecutive memory locations. In our implementations we use the SSE instructions wherever possible.

7.2.1 Intel AES new instructions architecture

In the Intel Core processor family based on the 32nm micro-architecture named *Westmere*, Intel introduced a set of instructions for the computation of the AES algorithm.

The AES-NI instruction set [Gue10] is comprised of six new instructions that perform several parts of the AES algorithm. These instructions can be executed using significantly less clock cycles than a software solution. Four of the new instructions are for accelerating the encryption/decryption of a round and two new instructions are for round key generation. Decryption using AES-NI instructions is done via the Equivalent Inverse Cipher of the AES algorithm. The following is a description of the new instructions.

- AESENC: This instruction performs a single round of encryption. The instruction combines the four transformations of the AES algorithm (ShiftRows, SubBytes, MixColumns and AddRoundKey) into a single instruction.
- AESENCLAST: Instruction for the last round of encryption. Combines the ShiftRows, SubBytes and AddRoundKey transformations into one instruction.
- AESDEC: Instruction for a single round of decryption. This combines the four transformations of AES (InvShiftRows, InvSubBytes, InvMixColumns and AddRoundKey) into a single instruction.
- AESDECLAST: Performs last round of decryption. It combines InvShiftRows, InvSubBytes and AddRoundKey into one instruction.
- AESKEYGENASSIST: It is used for generating the round keys used for encryption.

- AESMIC: It is used for converting the encryption round keys to a form usable for decryption using the Equivalent Inverse Cipher.

Beyond improving performance, the AES instructions provide important security benefits. These instructions are resistant to side channel leakages. More over, using these instructions code sizes are drastically reduced.

PCLMULQDQ. The seventh instruction of AES-NI is called PCLMULQDQ. This instruction performs a carry free multiplication of two 64 bit integers, which can be used for efficient implementation of multipliers in $GF(2^n)$. We use this instruction to multiply in $GF(2^{128})$ with the help of the Karatsuba algorithm (described in Fig. 7.1) and the reduction technique (described in Fig. 7.2).

7.3 Implementation

We implemented XCB, HCTR and EME2 for message length of 4096 bytes and tweak length of 128 bits. We implemented for message length of 4096 because this is the size of a sector for the currently available hard disks.

For all the implementations we use AES-128 as the underlying block cipher, and the finite field operations are in the field $GF(2^{128})$. Implementations were developed in C using the Intel Intrinsics. Where ever possible we used the SSE instructions to achieve parallelization and also we use the AES-NI for AES and PCLMULQDQ for multiplication in the field.

Gueron [Gue09] shown that it is possible to increase the performance for encryption using the AES instructions by re-ordering the code. This is because the hardware that supports the four AES round instructions is pipelined. This allows independent AES instructions to be dispatched theoretically every 1-2 CPU clock cycles, if data can be provided sufficiently fast. As a result, the AES throughput can be significantly enhanced for parallel modes of operation, if the order of the loop is reversed: instead of completing the encryption of one data block and then continuing to the subsequent block, it is preferable to write software sequences that compute one AES round on multiple blocks, using one round key, and only then continue to computing the subsequent round on for multiple blocks (using another round key). This turns out to be very convenient when we use the AES block cipher in parallel modes of operation as the CTR mode in case of XCB and HCTR, or the ECB mode in case of EME2. For such optimization it is necessary to choose the number of blocks that will be processed in parallel. We process 4 blocks in parallel, in order to achieve high throughput.

7.3.1 System information

The constructions were implemented on a computer with the following specifications:

- **CPU**: Intel Core i5-661 @ 3.33 GHz, 4M Cache (2 cores, 4 logical threads)

```
#define BENCHMARK(x) \
    for(i = 0; i < WARMUP; i++){ \
        x \
    } \
    start_cycles = get_cycles(); \
    for(i = 0; i < BENCH; i++){ \
        x \
    } \
    end_cycles = get_cycles(); \
    total = (double)(end_cycles - start_cycles)/BENCH;
```

Figure 7.7: Basic benchmark procedure.

- **Memory**: 4GB DDR3
- **Operating System**: GNU/Linux Fedora release 16
- **Compilers**: GCC 4.6.2 and ICC 12.0.1

For icc compiler we use the following compiling options `-xSSE4.2 -finline-functions -fast`. In case of gcc we use `-finline-functions -maes -msse4 -mpclmul -O3`.

7.3.2 Testing methodology

A benchmark procedure for performance evaluation was developed in [Tre12]. We use the same procedure, its skeleton is shown in Fig. 7.7. Where `x` represents the set of instructions used for the implementation of a specific scheme. To obtain the cycle counts for a specific set of instructions it is important to reduce the *cache effects*, the effects of transition from memory to data cache and memory to instruction cache. The first loop in Fig. 7.7 handles these cache effects using a technique known as "cache warming". *Warming up* the cache requires passing through the entire data set which is going to be used, so that it will be moved into the cache. Warming the instruction cache requires a first pass through all instructions before the timing begins. After the warm up, the cycle counts are computed as an average over 1,000,000 runs of the set of instructions.

In [Tre12] it is also shown the code used to obtain the number of cycles, We used the same code, and it is shown in Fig. 7.8.

7.3.3 Results

The results presented are an average over 1,000,000 calls to the encryption procedure. We present the performance comparison of the various modes in *cycles per bytes* (cpb). The number of CPU cycles needed to encrypt a message is divided by the length of the message to derive the cost per byte to encrypt messages of that length.

```
static __inline uint64_t get_cycles(void){
    uint64_t tmp;
    __asm__ volatile(
        "rdtsc\n\t\
        mov %%eax,(%0)\n\t\
        mov %%edx,4(%0)"::"rm"(&tmp):"eax","edx");
    return tmp;
}
```

Figure 7.8: Code used to measure time.

	gcc (cpb)	**icc** (cpb)
HCTR	8.27	7.97
XCBv2	9.71	9.25
EME2	6.92	5.08

Table 7.1: Encryption implementation results in cycles per byte, for a message of 4096 bytes and a tweak of 128 bits.

Performance results of the implementations of HCTR, XCBv2 and EME2 are shown in Table 7.1. We did not implement XCBv1 because of its close similarity with XCBv2. For all the implementation we consider the computation of the AES key expansion.

As we saw in Section 5.3.2, in page 62, EtE was proposed to secure TES when encrypting their own key at a cost of adding a layer of encryption. EtE requires that the underlying block cipher be secure when encrypting its own key, which adds some extra operations. In the proposal of EtE [BCK11], XEX a tweakable block cipher which uses AES at the bottom was used as the underlying block cipher.

For the sake of comparison we did not implement the full specification of EtE. Instead we implemented EtE*[1] which adds a layer of ECB encryption using AES as the underlying block cipher. In this sense EtE* has a better performance of that of EtE. The latter is also reported in the original description of EtE [BCK11]. Table 7.2 reports the performance of EtE* when instantiated with HCTR, XCBv2 and EME2.

EtE assumes that the TES which will be secured only use one key. However TES we are considering use more than one key. In the implementations we consider h as the key for HCTR and XCBv2, and K for EME2. These keys are also used to construct the message that will be encrypted using EtE*.

We also implemented MStE, our proposal to secure TES when encrypting their own set of keys. Results are shown in Table 7.3. For HCTR we considered two keys. For XCBv2 we considered four keys (although in the specification of XCBv2 is

[1]The name EtE* was not given by its proposers. We use this nomenclature to separate the difference with EtE.

	gcc (cpb)	**icc** (cpb)
EtE*[HCTR]	9.44	9.06
EtE*[XCBv2]	11.29	10.63
EtE*[EME2]	8.13	6.44

Table 7.2: Encryption implementation results when instantiated EtE* with HCTR, XCBv2 and EME2.

	No. keys	**gcc** (cpb)	**icc** (cpb)
MStE[HCTR]	2	8.93	8.45
MStE[XCBv2]	4	10.90	10.19
MStE[EME2]	3	7.46	5.68

Table 7.3: Encryption implementation results when instantiated MStE with HCTR, XCBv2 and EME2.

considered as an input only one key XCBv2 actually uses four keys, see Fig. 3.3) and for EME2 we considered three keys. For the implementations we considered messages of 4096 bytes where each block of 128 bits equals one of the keys of the respective TES.

Table 7.4 shows the comparison between MStE and EtE*. Shaded cells show the least overhead when instantiated either MStE or EtE*. In the experiments we obtained that MStE has a better performance than EtE in all the TES we considered.

As we can see from the results, the percentage the instance of MStE is slower compared with the implementation of the TES depends on the number of keys the TES use. We can see that the overhead of XCB for MStE is maximum, this is because XCB has four keys.

	MStE				**EtE***			
	gcc		**icc**		**gcc**		**icc**	
	cpb	%	cpb	%	cpb	%	cpb	%
HCTR	0.66	7.98	0.48	6.02	1.17	14.14	1.09	13.67
XCB	1.19	12.25	0.94	10.16	1.58	16.27	1.38	14.91
EME2	0.54	7.80	0.60	11.81	1.21	17.48	1.36	26.77

Table 7.4: Overhead involved in MStE and EtE* transformations.

8. Conclusions and Future Work

> I hope we'll be able to solve
> these problems before we leave.
>
> Paul Erdős

In this chapter we present our conclusions and summarize the contributions. At the end of the chapter we list some related topics of interest which were not considered in this work.

8.1 Conclusions and Summary of Contributions

In this thesis we considered two different problems related to security of TES.
Security of XCB: We carefully analyzed the two versions of the TES named XCB. The result of our analysis were the following:

1. XCBv2 as specified in [MF07] is not secure as a TES. We found an easy distinguishing attack on XCBv2. The attack works because of a faulty padding scheme, and there seems to be no easy way to fix this problem. However, if the inputs to XCBv2 are such that their lengths are multiples of the block length of the block cipher, then our attack does not work, and for this restricted message space XCBv2 is secure.

2. Even for the restricted message space, XCBv2 (possibly) does not have the security bound as claimed in [MF07]. This is due to the fact that the proof of the security theorem in [MF07] is wrong. The error stems from a faulty calculation of collision probabilities in the incr function. We point out the mistake by showing concrete examples where that the bound on the collision probabilities in the incr function as given in [MF07] are violated. We use heavily some combinatorial techniques and results presented in [IOM12] for constructing these examples.

3. We provide a corrected security bound for XCBv2 in the restricted message space scenario. Also we provide a detailed proof which demonstrate why the bound is correct. Our bound is worse than that claimed in [MF07].

4. XCBv1 does not suffer from the weaknesses as in XCBv2. The distinguishing attack which we present for XCBv2 does not work for XCBv1. XCBv1 (as specified in [MF04]) is a secure TES. There was no proof of the fact that XCBv1 is secure. We for the first time provide a proof of security for XCBv1 along with a concrete security bound.

5. XCBv2 was derived as a small modification of XCBv1. The authors said that the modifications were made to enable easy analysis [MF07]. Though it is not very clear to us, how these modifications help in the analysis. Our analysis reveals that any modification in an existing cryptographic scheme should be done with utmost care, even an innocent looking change may have a grave impact on the security of the scheme.

6. XCBv2 is a part of the standard IEEE Std 1619.2-2010. Our analysis puts into serious doubts the methodology adopted by the working group for formulating the standard. We are surprised that an international standardization committee for a cryptographic scheme overlooked some important security issues, which were not so difficult to detect. Thus, our analysis on XCB also says that outcomes of standards should also be critically analyzed before deploying them in a real application.

KDM security of TES: KDM security of TES is not well studied, we add on to the scant literature in the following ways:

1. We point out KDM insecurity in two concrete schemes HCTR and XCB. Prior to our work no specific attacks were known on existing TES in the KDM sense. We think that our attacks can be extended to other schemes which uses polynomial hashes.

2. There exists a transform EtE [BCK11], which converts a SPRP secure TES to a KDM secure TES. We point out some deficiencies in the construction. In particular, EtE can only be applied to those schemes which have a single key. Most TES reported in the literature uses multiple keys, hence EtE cannot be suitably applied in such cases.

3. We propose a new transformation MStE which converts a SPRP secure TES to a KDM secure TES. MStE does not suffer from the limitation of EtE. i.e., it can work for schemes which uses multiple keys. We also formally prove the security of MStE.

4. Finally, we implement MStE for some existing TES. In our implementation we use the special instruction set extensions of Intel for implementing AES and finite field multiplication. Also we use the SIMD instructions in our implementations wherever possible. We implement EtE also using the same paradigm, and compare the running times for EtE and MStE. Our experimental results suggest that MStE is faster than EtE.

8.2 Future Work

We mention some of the issues and problems that we did not consider in this thesis, but can be of immediate interest.

1. We showed new security bounds for both XCBv1 and XCBv2. But we are not sure whether these bounds are tight. One way to be sure about the tightness of the bounds is to search for some matching attacks for the schemes, i.e., to construct adversaries which would achieve the upper bound. We tried to think in this direction, but as of now we do not have any idea how such adversaries can be constructed for XCBv1 or XCBv2.

2. There are some secure TES, where there are no known attacks in the KDM sense. One such example is EME2. It would be an interesting project to prove that EME2 is secure (or insecure) in the KDM sense.

3. MStE is a generic scheme, and it works for all TES. Such a generic scheme is always interesting from a theoretical viewpoint. But in most cases, generic schemes are not efficient enough. KDM insecurity of TES should be analyzed on a case to case basis. Such an analysis may help to point out specific reasons for KDM insecurity for different TES and then fixes can also be designed on a case to case basis. It is expected that such particular fixes can be obtained with minor changes to the original construction and thus would be more efficient.

4. MStE has a restriction that it only works for schemes where the length of the key(s) is same as the block length of the block cipher. For example MStE cannot be used with say EME2 with AES-192 as the underlying block cipher. It would be nice to overcome this limitation. But again, it may need a scheme completely different from MStE to achieve this. We plan to explore in this direction.

A. Properties of the Hash Function in XCBv2

In this section we prove the properties of the hash function which were included in the changes made to XCBv2.

Proof of Theorem 4.1.1. From the definition of H

$$\begin{aligned} H_h(T,P) =& T_1 h^{p+m+1} \oplus T_2 h^{p+m} \oplus \ldots \oplus \mathsf{pad}\,(T_p)\, h^{m+2} \\ & \oplus P_1 h^{m+1} \oplus P_2 h^m \oplus \ldots \oplus \mathsf{pad}(P_m) h^2 \\ & \oplus (\mathsf{bin}_{\frac{n}{2}}(|T|)||\mathsf{bin}_{\frac{n}{2}}(|P|))h \end{aligned}$$

$$\begin{aligned} H_h(T',P') =& T'_1 h^{p+m+1} \oplus T'_2 h^{p+m} \oplus \ldots \oplus \mathsf{pad}\left(T'_p\right) h^{m+2} \\ & \oplus P'_1 h^{m+1} \oplus P'_2 h^m \oplus \ldots \oplus \mathsf{pad}(P'_m) h^2 \\ & \oplus (\mathsf{bin}_{\frac{n}{2}}(|T|)||\mathsf{bin}_{\frac{n}{2}}(|P|))h \end{aligned}$$

$$\begin{aligned} H_h(T,P) \oplus H_h(T',P') =& (T_1 \oplus T'_1) h^{p+m+1} \oplus (T_2 \oplus T'_2) h^{p+m} \oplus \ldots \\ & \oplus \mathsf{pad}\left(T_p \oplus T'_p\right) h^{m+2} \oplus (P_1 \oplus P'_1) h^{m+1} \oplus (P_2 \oplus P'_2) h^m \oplus \ldots \\ & \oplus \mathsf{pad}(P_m \oplus P'_m) h^2 \oplus (\mathsf{bin}_{\frac{n}{2}}(|T|)||\mathsf{bin}_{\frac{n}{2}}(|P|))h \\ & \oplus (\mathsf{bin}_{\frac{n}{2}}(|T'|)||\mathsf{bin}_{\frac{n}{2}}(|P'|))h, \end{aligned}$$

as $|T| = |T'|$ and $|P| = |P'|$

$$H_h(T,P) \oplus H_h(T',P') = H_h(T \oplus T', P \oplus P') \oplus (\mathsf{bin}_{\frac{n}{2}}(|T|)||\mathsf{bin}_{\frac{n}{2}}(|P|))h$$

□

In the proof of Theorem 4.1.2 it is implicit that the padding must be computed before the hash function. For proving Theorem 4.1.2 we need to show that

$$CC = MM \oplus H_h(Z, E_{K_c}(\mathsf{inc}^0(S))|| \ldots ||\mathsf{drop}(E_{K_c}(\mathsf{inc}^{m-2}(S))))h,$$

where $Z = (T||0^n) \oplus (0^n||T)$.

Proof of Theorem 4.1.2. From Algorithm 3.3

$$MM = S \oplus H_h(T||0^n, C_1|| \dots ||\mathsf{pad}(C_{m-1})||(\mathsf{bin}_{\frac{n}{2}}(|T||0^n|)||\mathsf{bin}_{\frac{n}{2}}(|C_1|| \dots ||C_{m-1}|)))$$

and

$$S = CC \oplus H_h(0^n||T, P_1|| \dots ||\mathsf{pad}(P_{m-1})||0^n).$$

Then using Theorem 4.1.1

$$\begin{array}{rrl} MM = & CC\oplus & H_h((T||0^n) \oplus (0^n||T), C_1 \oplus P_1|| \dots ||\mathsf{pad}(C_{m-1} \oplus P_{m-1}) \\ & & ||(\mathsf{bin}_{\frac{n}{2}}(|T||0^n|)||\mathsf{bin}_{\frac{n}{2}}(|C_1|| \dots ||C_{m-1}|))) \\ & \oplus & (\mathsf{bin}_{\frac{n}{2}}(|T||0^n|)||\mathsf{bin}_{\frac{n}{2}}(|P_1|| \dots ||P_{m-1}||0^n|))h. \end{array}$$

Recall that C_i is obtained from $E_{K_c}(\mathsf{inc}^{i-1}(S)) \oplus P_i$. Thus

$$\begin{array}{rrl} MM = & CC\oplus & H_h((T||0^n) \oplus (0^n||T), E_{K_c}(\mathsf{inc}^0(S))|| \dots ||\mathsf{drop}(E_{K_c}(\mathsf{inc}^{m-2}(S)) \\ & & ||(\mathsf{bin}_{\frac{n}{2}}(|T||0^n|)||\mathsf{bin}_{\frac{n}{2}}(|C_1|| \dots ||C_{m-1}|))) \\ & \oplus & (\mathsf{bin}_{\frac{n}{2}}(|T||0^n|)||\mathsf{bin}_{\frac{n}{2}}(|P_1|| \dots ||P_{m-1}||0^n|))h. \end{array}$$

The last two terms of the above hash function would be

$$(\mathsf{bin}_{\frac{n}{2}}(|T||0^n|)||\mathsf{bin}_{\frac{n}{2}}(|C_1|| \dots ||C_{m-1}|))h^2 \oplus (\mathsf{bin}_{\frac{n}{2}}(|T||0^n|)||\mathsf{bin}_{\frac{n}{2}}(|P_1|| \dots ||P_{m-1}||0^n|))h.$$

Then

$$MM = CC \oplus H_h((T||0^n) \oplus (0^n||T), E_{K_c}(\mathsf{inc}^0(S))|| \dots ||\mathsf{drop}(E_{K_c}(\mathsf{inc}^{m-2}(S))))h.$$

This is why we need the padding be computed before the hash function, otherwise the terms would not cancel. □

Bibliography

[BCK11] M. Bellare, D. Cash, and S. Keelveedhi. Ciphers that securely encipher their own keys. In *Proceedings of the 18th ACM conference on Computer and communications security*, CCS '11, pages 423–432, New York, NY, USA, 2011. ACM.

[BFK+12] R. Bardou, R. Focardi, Y. Kawamoto, L. Simionato, G. Steel, and J. Tsay. Efficient padding oracle attacks on cryptographic hardware. In *Advances in Cryptology–CRYPTO 2012*, pages 608–625. Springer, 2012.

[BHHO08] D. Boneh, S. Halevi, M. Hamburg, and R. Ostrovsky. Circular-secure encryption from decision Diffie-Hellman. In *Advances in Cryptology–CRYPTO 2008*, pages 108–125. Springer, 2008.

[BR04] M. Bellare and P. Rogaway. Code-based game-playing proofs and the security of triple encryption. Cryptology ePrint Archive, Report 2004/331, 2004. http://eprint.iacr.org/.

[BR06] M. Bellare and P. Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In *Advances in Cryptology - Eurocrypt 2006*, pages 409–426. Springer, 2006.

[BRS02] J. Black, P. Rogaway, and T. Shrimpton. Encryption-scheme security in the presence of key-dependent messages. In *In Selected Areas in Cryptography, volume 2595 of LNCS*, pages 62–75. Springer-Verlag, 2002.

[CN08] D. Chakraborty and M. Nandi. An improved security bound for hctr. In *Fast Software Encryption*, pages 289–302. Springer, 2008.

[CS06a] D. Chakraborty and P. Sarkar. HCH: A new tweakable enciphering scheme using the hash-encrypt-hash approach. In *INDOCRYPT*, pages 287–302, 2006. Extended version in http://eprint.iacr.org/2007/028.

[CS06b] D. Chakraborty and P. Sarkar. A new mode of encryption providing a tweakable strong pseudo-random permutation. In *Fast Software Encryption*, pages 293–309, 2006.

[Dwo01] M. Dworkin. Recommendation for block cipher modes of operation. methods and techniques. Technical report, DTIC Document, 2001.

[FIP01] Specification for the advanced encryption standard (AES). Federal Information Processing Standards Publication 197, 2001. http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf.

[GK10] S. Gueron and M.E. Kounavis. Intel® carry-less multiplication instruction and its usage for computing the GCM mode. *White paper, Intel Corporation*, 2(3):11, 2010.

[Gue09] S. Gueron. Intel's new AES instructions for enhanced performance and security. In *Fast Software Encryption*, pages 51–66. Springer, 2009.

[Gue10] S. Gueron. Intel advanced encryption standard (AES) instructions set. *Intel White Paper, Rev*, 3, 2010.

[Hal04] S. Halevi. EME*: Extending EME to handle arbitrary-length messages with associated data. In *INDOCRYPT*, pages 315–327. Springer, 2004.

[Hal07] S. Halevi. Invertible universal hashing and the tet encryption mode. In *CRYPTO*, volume 4622 of *Lecture Notes in Computer Science*, pages 412–429. Springer, 2007.

[HK07] S. Halevi and H. Krawczyk. Security under key-dependent inputs. In *Proceedings of the 14th ACM conference on Computer and communications security*, CCS '07, pages 466–475, New York, NY, USA, 2007. ACM.

[HR03] S. Halevi and P. Rogaway. A tweakable enciphering mode. In *Advances in Cryptology-CRYPTO 2003*, pages 482–499. Springer, 2003.

[HR04] S. Halevi and P. Rogaway. A parallelizable enciphering mode. In *Topics in Cryptology–CT-RSA 2004*, pages 292–304. Springer, 2004.

[IOM12] T. Iwata, K. Ohashi, and K. Minematsu. Breaking and repairing GCM security proofs. In *Advances in Cryptology - Crypto 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 31–49. Springer, 2012.

[Jou03] A. Joux. Cryptanalysis of the EMD mode of operation. In *Advances in Cryptology - Eurocrypt 2003*, pages 1–16. Springer, 2003.

[KL07] J. Katz and Y. Lindell. *Introduction to Modern Cryptography (Chapman & Hall/Crc Cryptography and Network Security Series)*. Chapman & Hall/CRC, 2007.

[LRW02] M. Liskov, R. L. Rivest, and D. Wagner. Tweakable block ciphers. In *Advances in Cryptology - CRYPTO 2002*, pages 31–46. Springer, 2002.

[MF04] D. A. McGrew and S. R. Fluhrer. The extended codebook (XCB) mode of operation. Cryptology ePrint Archive, Report 2004/278, 2004. http://eprint.iacr.org/.

[MF07] D. A. McGrew and S. R. Fluhrer. The security of the extended codebook (XCB) mode of operation. In Carlisle Adams, Ali Miri, and Michael Wiener, editors, *Selected Areas in Cryptography*, volume 4876 of *Lecture Notes in Computer Science*, pages 311–327. Springer Berlin Heidelberg, 2007.

[MLMI13] K. Minematsu, S. Lucks, H. Morita, and T. Iwata. Attacks and security proofs of EAX-prime. In *Pre-proceedings of Fast Software Encryption*, 2013.

[MV04] D. A. McGrew and J. Viega. Arbitrary block length mode, 2004. http://grouper.ieee.org/groups/1619/email/pdf00005.pdf.

[NR97] M. Naor and O. Reingold. A pseudo-random encryption mode. In *UNPUBLISHED*, 1997. http://www.wisdom.weizmann.ac.il/~naor.

[NR99] M. Naor and O. Reingold. On the construction of pseudo-random permutations. In *Journal of Cryptology*, pages 29–66, 1999.

[OST06] D. A. Osvik, A. Shamir, and E. Tromer. Cache attacks and countermeasures: the case of AES. In *Topics in Cryptology–CT-RSA 2006*, pages 1–20. Springer, 2006.

[P1611] IEEE Std 1619.2-2010: IEEE standard for wide-block encryption for shared storage media. IEEE Computer Society, March 2011. http://standards.ieee.org/findstds/standard/1619.2-2010.html.

[Sar07] P. Sarkar. Improving upon the TET mode of operation. In *INDOCRYPT*, volume 4817 of *Lecture Notes in Computer Science*, pages 180–192. Springer, 2007.

[Sar09] P. Sarkar. Efficient tweakable enciphering schemes from (block-wise) universal hash functions. *Information Theory, IEEE Transactions on*, 55(10):4749–4760, 2009.

[Sti91] D. R. Stinson. Universal hashing and authentication codes. In *Designs, Codes and Cryptography*, pages 74–85. Springer, 1991.

[Sti02] D. R. Stinson. Cryptography: Theory and practice. 2002, 2002.

[Tre12] N. I. G. Trejo. Efficient software implementations of disk encryption schemes using AES-NI support. CINVESTAV-IPN:MX, 2012. M.Sc. Thesis http://www.cs.cinvestav.mx/TesisGraduados/2012/TesisNallelyTrejo.pdf.

[WFW05] P. Wang, D. Feng, and W. Wu. Hctr: A variable-input-length enciphering mode. In *Information Security and Cryptology*, pages 175–188. Springer, 2005.