Indian Statistical Institute, Kolkata

Inverse Free HCTR: A Length Preserving Tweakable Enciphering Mode

by

Mohammad Chharchhodawala

A report submitted in partial fulfillment for the
degree of Master of Technology in Computer Science

Guided by
Prof. Debrup Chakraborty
CSR UNIT

July 2018

Declaration

I hereby declare that the dissertation report entitled **"Inverse Free HCTR: A Length Preserving Tweakable Enciphering Mode"** submitted to Indian Statistical Institute, Kolkata, is a *bona fide* record of work carried out in partial fulfilment for the award of the degree of **Master of Technology in Computer Science**. The work has been carried out under the guidance of **Prof. Debrup Chakraborty**, Associate Professor, CSRU, Indian Statistical Institute, Kolkata.

I further declare that this work is original, composed by myself. The work contained herein is my own except where stated otherwise by reference or acknowledgement, and that this work has not been submitted to any other institution for award of any other degree or professional qualification.

Place : Kolkata
Date : July 2018

Mohammad Chharchhodawala
Roll No: CS-1618

CERTIFICATE

This is to certify that the dissertation entitled **"Inverse Free HCTR: A Length Preserving Tweakable Enciphering Mode"** submitted by **Mohammad Chharchhodawala** to Indian Statistical Institute, Kolkata, in partial fulfillment for the award of the degree of **Master of Technology in Computer Science** is a *bona fide* record of work carried out by him under my supervision and guidance. The dissertation has fulfilled all the requirements as per the regulations of this institute and, in my opinion, has reached the standard needed for submission.

Prof. Debrup Chakraborty
Associate Professor & Head,
Cryptology and Security Research Unit,
Indian Statistical Institute,
Kolkata-700108, India.

Abstract

Inverse Free HCTR (IFHCTR) is a length-preserving encryption scheme, which provides a tweakable strong pseudorandom permutation. IFHCTR is modification of HCTR scheme in which inverse of block cipher is not required. IFHCTR supports arbitrary variable input length with the minore restriction that data should be at least $2n$ bits long, where n is the block length of block cipher. IFHCTR also supports variable length tweaks. We prove that IFHCTR is a strong tweakable pseudorandom permutation (sprp), when the underlying blockcipher is a pseudorandom function (prf). IFHCTR can be used in disk sector encryption, and other applications where length-preserving encryptions are required, especially when size of the message is not multiple of n bits.

Keywords: Inverse free schemes, Length-preserving enciphering scheme, Tweakable enciphering scheme, HCTR

Acknowledgements

I would like to thank my thesis advisor Prof. Debrup Chakraborty. The door of his office was always open whenever I ran into a trouble spot or had a question about my work or writing.

I would be failing in my duties if I don't mention friends and people who mattered and conversing with whom did help me during two years at ISI Kolkata. In this period I have been enriched and nurtured in numerous ways by interaction with many people. Especially I would like to express my gratitude to research scholars of CSRU & ASU and Manish for keeping a positive workspace.

Lastly, about Deepayan Sanyal and Neilutpal Saha, whose friendships, I have cherished the most during two years at ISI Kolkata. I am indebted to them for their help during my trying times. We three enjoyed good times. Hope our friendship continues with the same fervor.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Data privacy is usually achieved by encryption. A block cipher is a vital primitive to design encryption schemes. However, in most application environments, only block ciphers can't provide the required security. In such cases block ciphers are used in a special way, which is called a block cipher mode of operation, to achieve the required functionality and security. Among others, this case occurs in case of security of stored data, especially in the application of disk sector encryption.

A well accepted solution for encryption of sector/block oriented storage devices is low level disc encryption or in-place disc encryption. Low level disk encryption is encryption at the hardware level which converts data on a hard disks, USB sticks etc into a form that cannot be understood by anyone who doesn't have the key to "undo" the conversion. Without the proper key, even if the hard drive is removed and placed in another machine, the data remains inaccessible. In low level disc encryption, the encryption and decryption algorithms reside in the disk-controller. These algorithms have no knowledge about the high-level logical partitions of the disk, like files and directories but have access only to the disk sectors. The disk-controller performs encryption of data before it writes a sector, and performs decryption of data which is stored in sector before sending it to the operating system.

An important property required for low level disc encryption is length preservation. We call an encryption scheme an enciphering scheme when it is *length-preserving*, i.e., when the length of the plaintext matches the length of the ciphertext. Another important property required of schemes to be used for low level disc encryption is ciphertext variability. In simple terms, it means that, the encryption of the same data which resides in different disc sectors should be unrelated and different. If an adversary swap the data of two disc sectors, then their decryption should give random looking data which is unlikely to be meaningful. The security requirement of these schemes is that the ciphertext should look random and a one bit change in the ciphertext should yield a random plaintext on decryption. All these properties are provided by a cryptographic object called *tweakable enciphering scheme (TES)*.

A tweakable enciphering scheme takes as input a tweak t, a key K, and a message M, and outputs a ciphertext C. Here, the tweak is an extra input, generally it may be an initial value, a state, a position, a mark, a file name, or something else. As mentioned earlier, the disk is encrypted sector-wise. Each sector is 4096 bytes and has a sector address. To perform encryption of sector data the requirement is to use a *tweakable enciphering scheme* with the sector address as the tweak and the plaintext as the 4096 byte data in the sector.

In the last two decades several *tweakable enciphering schemes* have been proposed. The existing constructions can be broadly classified into two categories; ones which use only block ciphers and the others use both block ciphers and some kind of polynomial hash functions. Some notable examples in the former category are EME[1], EME*[2], CMC[3], Fmix[4] and AEZ-core[5] etc. Whereas some examples in the later category are PEP[6], TET[7], HEH[8], HCTR[9], XCB[10], FAST[11] etc. STES[12] is a construction related to the later category but it uses stream ciphers instead of block ciphers.

Efficiency in both software and hardware is a major design goal for TES. Thus schemes with lesser operation counts, better options for parallel implementation and small footprint when implemented in hardware are preferred. Another design goal is to reduce the number of keys required for the construction. As keys are to be stored securely and storing more keys securely is more difficult.

We call a block cipher based TES an *inverse free scheme* if the construction depends only on the encryption function of the blockcipher both while encrypting and decrypting, thus never needing to call decryption function of block cipher. These designs have various advantages, like requiring just a a least stringent assumption on the security of the underlying blockcipher and substantial savings in hardware as the decryption module of the block cipher is not required to be implemented, Especially in disc encryption where encryption algorithm resides just above the disk controller). A hardware implementation would be preferred for a typical deployment and thus inverse free schemes which occupies less area in hardware are well suited. This scheme is also advantageous when decryption function takes more time to execute than encryption function or vice versa in some enciphering scheme. As per our knowledge the first inverse free tweakable enciphering scheme was proposed in [13]. Later proposed constructions are FAST[11], Fmix[4] and AEZ-core[5].

Our Contribution: We present a new inverse free TES which has simpler description compared to the existing ones. Also it is comparable to the existing ones in the terms of security and efficiency.

Our construction is obtained as a modification of HCTR. HCTR is a scheme which uses two layers of polynomial hashes along with a block cipher counter mode as it's main components. Other than these, HCTR also has a "single" block cipher call whose inverse is required to be computed at decryption. We remove this "single" block cipher call in our construction and obtain an *inverse free* cipher. We call our scheme as IFHCTR, i.e, inverse free HCTR.

The rest of the document is organized as follows. In Chapter 2 we fix some notations and introduce some basic cryptographic objects which we extensively use in the later chapters. Chapter 3 forms the most important part of this report where we describe the construction of IFHCTR and prove its security. Chapter 4 provides some preliminary experimental results on performance of IFHCTR when implemented in modern Intel Processors equipped with the AES-NI instruction set.

Chapter 2

Preliminaries

Notation: We denote the concatenation of two strings X and Y by $X||Y$. By $|X|$ we shall mean the length of X in bits. By $|X|_n$ we mean $\lceil \frac{|X|}{n} \rceil$, which we call the number of n-bit blocks in X. $\mathsf{bin}_n(\ell)$ will denote the n bit binary representation of an integer ℓ, where $0 \leq \ell < 2^n$. $\{0,1\}^n$ denotes set of all binary strings of the length n. $\mathsf{pad}(X)$ will denote the string $X||0^i$, where i is the minimum nonnegative integer such that $|X||0^i|$ is divisible by n. $\mathsf{drop}_r(X)$ drops last r bits of X and gives a string $|X| - r$ bits.

Binary Strings and Finite Field: We sometimes see n-bit strings to be elements in a finite field of size 2^n, i.e we view $\{0,1\}^n$ as $GF(2^n)$. Thus, we think of an n-bit string $L = L_{n-1} \ldots L_1 L_0 \in \{0,1\}^n$ as the polynomial $L(x) = L_{n-1}x^{n-1} + \ldots + L_1 x + L_0$. For $A, B \in GF(2^n)$, $A \oplus B$ and AB will denote addition and multiplication in the field respectively. To add two points A, B, we take their bit wise xor. To multiply two elements we must fix an irreducible polynomial $P_n(x)$ having binary coefficients and degree n: say the lexicographically first polynomial among the irreducible degree-n polynomials having a minimum number of nonzero coefficients. For $n = 128$, the indicated pentanominal polynomial is $P_{128}(x) = x^{128} + x^7 + x^2 + x + 1$. Now multiply $A(x)$ and $B(x)$ by forming the degree $2n - 2$ (or less) polynomial that is their product and taking the remainder when this polynomial is divided by $P_n(x)$. $B(x)$ is called as an *inverse* of $A(x)$ if we multiply $A(x)$ and $B(x)$ and it gives reminder 1 when their product is divided by $P_n(x)$. Inverse of $A(x)$ is denoted by $A(x)^{-1}$.

Block Cipher: A conventional *block cipher* takes two inputs. a *key* $K \in \{0,1\}^k$ and a *message(plaintext)* $M \in \{0,1\}^n$; and produces a single output- a *ciphertext* $C \in \{0,1\}^n$. The signature for a block cipher is $E : \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n$. We generally write $E_K(M)$ instead of $E(K, M)$. We call k as the key length and n as the size of block throughout this report. Formally, a *block cipher* is seen as family of permutations where the *key* selects a particular permutation from that family.

Tweakable Block Cipher: A *tweakable block cipher* takes three inputs a *key* $K \in \{0,1\}^k$, *tweak* $T \in \{0,1\}^t$ and a *message (plaintext)* $M \in \{0,1\}^n$; and produces a single output- a *ciphertext* $C \in \{0,1\}^n$. The signature for a tweakable block cipher is $\tilde{E} : \{0,1\}^k \times \{0.1\}^t \times \{0,1\}^n \rightarrow \{0,1\}^n$. We generally write $\tilde{E}_K^T(M)$ instead of $\tilde{E}(K,T,M)$. With a *tweakable block cipher* both *key* and *tweak* are used to select a permutation.

Tweakable Enciphering Scheme: A *tweakable enciphering scheme* is a function $\mathbb{E} : \mathcal{K} \times \mathcal{T} \times \mathcal{M} \rightarrow \mathcal{M}$ where $\mathcal{M} = \cup_{i \geq 1}\{0,1\}^i$ is the *message space*, $\mathcal{K} \neq \phi$ is the *key space* and $\mathcal{T} \neq \phi$, is the *tweak space*. We require that for every $K \in \mathcal{K}$ and $T \in \mathcal{T}$ we have that $\mathbb{E}(K,T,.) = \mathbb{E}_K^T(.)$ is a length-preserving permutation on $\mathcal{M}$. The inverse of an enciphering scheme $\mathbb{E}$ is the enciphering scheme $\mathbb{D} = \mathbb{E}^{-1}$ where $X = \mathbb{D}_K^T(Y)$ if and only if $\mathbb{E}_K^T(X) = Y$. A *block cipher* is the special case of a *tweakable enciphering scheme* where the message space is $\mathcal{M} \in \{0,1\}^n$ (for some fixed $n \geq 1$) and the tweak space is $\mathcal{T} = \{\epsilon\}$ (the empty string). A *tweakable block cipher* is a TES with message space $\mathcal{M} \in \{0,1\}^n$ (for some $n \geq 1$).

Adversary: An adversary A is a (possibly probabilistic) algorithm with access to some oracles. Oracles are written as superscripts. The notation $A^O \Rightarrow 1$ describes the event that the adversary A interacting with the oracle O outputs the bit one.

Pseudorandom Function(PRF): Let $\{F_K\}_{K \in \mathcal{K}}$ be a function family, such that for every $K \in \mathcal{K}$, $F_K : \{0,1\}^n \rightarrow \{0,1\}^n$. Let Func($n$) is a set of functions mapping n-bit strings to n-bit strings and A be an adversary. We define the *prf* advantage of A for F as:

$$\mathsf{Adv}_F^{\mathrm{prf}}(A) = |\Pr[K \xleftarrow{\$} \mathcal{K} : A^{F_K(.)} \Rightarrow 1] - \Pr[f \xleftarrow{\$} \mathrm{Func}(n) : A^{f(.)} \Rightarrow 1]|. \tag{2.1}$$

We say the family $\{F_K\}_{K \in \mathcal{K}}$ is a *pseudorandom function family* if for all *"efficient"* A, Adv is *"small"*.

Pseudorandom Permutation(PRP): Let $\{\Pi_K\}_{K \in \mathcal{K}}$ be a permutation family, such that for every $K \in \mathcal{K}$, $\Pi_K : \{0,1\}^n \rightarrow \{0,1\}^n$. Let Perm($n$) is a set of permutations mapping n-bit strings to n-bit strings and A be an adversary. We define the *prp* advantage of A for Π as

$$\mathsf{Adv}_\Pi^{\mathrm{prp}}(A) = |\Pr[K \xleftarrow{\$} \mathcal{K} : A^{\Pi_K(.)} \Rightarrow 1] - \Pr[\pi \xleftarrow{\$} \mathrm{Perm}(n) : A^{\pi(.)} \Rightarrow 1]|. \tag{2.2}$$

We say the family $\{\Pi_K\}_{K \in \mathcal{K}}$ is a *pseudorandom permutation family* if for all *"efficient"* A, Adv is *"small"*.

Strong Pseudorandom Permutation(SPRP): Let $\{\Pi_K\}_{K\in\mathcal{K}}$ be a permutation family, such that for every $K \in \mathcal{K}$, $\Pi_K : \{0,1\}^n \rightarrow \{0,1\}^n$. Let $\mathrm{Perm}(n)$ is a set of permutations mapping n-bit strings to n-bit strings and A be an adversary. We define the *prp* advantage of A for Π as

$$\mathsf{Adv}_{\Pi}^{\pm\mathrm{prp}}(A) = |\Pr[K \xleftarrow{\$} \mathcal{K} : A^{\Pi_K(.),\Pi_K^{-1}(.)} \Rightarrow 1] - \Pr[\pi \xleftarrow{\$} \mathrm{Perm}(n) : A^{\pi(.),\pi^{-1}(.)} \Rightarrow 1]|. \tag{2.3}$$

We say the family $\{\Pi_K\}_{K\in\mathcal{K}}$ is a *strong pseudorandom permutation family* if for all *"efficient"* A, Adv is *"small"*.

Block Cipher Security: The standard security assumption on a block cipher $E : \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n$ is that E_K is a strong pseudorandom permutation. In certain usage scenarios, say where the inverse of block cipher is never used, a weaker assumption like the block cipher is a secure prf may suffice. In the construction that we later propose we will never use the decryption functionality of the block cipher and thus for us assuming the block cipher to the prf secure would be enough.

Security Measure For Tweakable Enciphering Scheme: For a tweakable enciphering scheme $\mathbb{E} : \mathcal{K} \times \mathcal{T} \times \mathcal{M} \rightarrow \mathcal{M}$ we consider the advantage that the adversary A has in distinguishing $\mathbb{E}$ and its inverse from a random permutation and its inverse as

$$\mathsf{Adv}_{\mathbb{E}}^{\pm\widetilde{\mathrm{prp}}}(A) = |\Pr[K \xleftarrow{\$} \mathcal{K} : A^{\mathbb{E}_K(\cdot,\cdot),\mathbb{E}_K^{-1}(\cdot,\cdot)} \Rightarrow 1] - |\Pr[\pi \xleftarrow{\$} \mathrm{Perm}^{\mathcal{T}}(\mathcal{M}) : A^{\pi(\cdot,\cdot),\pi^{-1}(\cdot,\cdot)} \Rightarrow 1]|. \tag{2.4}$$

where, $\mathrm{Perm}^{\mathcal{T}}\ (\mathcal{M})$ is the set of all functions $\pi : \mathcal{T} \times \mathcal{M} \rightarrow \mathcal{M}$ where $\pi(\mathcal{T}, .)$ is a length-preserving permutation. In $\widetilde{\mathrm{prp}}$ the tilde serves as a reminder that the PRP is tweakable. In the above definition we assume some restrictions on the adversary. Without loss of generality we assume that an adversary never repeats an encrypt query, never repeats a decrypt query, never queries its decrypting oracle with (T, C) if it got C in response to some (T, M) encrypt query, and never queries its encrypting oracle with (T, M) if it earlier got M in response to some (T, C) decrypt query. We call such queries *pointless* because the adversary already "knows" the answer that it should receive.

Let $\mathbb{E} : \mathcal{K} \times \mathcal{T} \times \mathcal{M} \rightarrow \mathcal{M}$ be a *tweakable enciphering scheme* and let $\mathbb{D}$ be its inverse. Define the advantage of A in distinguishing $\mathbb{E}$ from random bits, $\mathsf{Adv}_{\mathbb{E}}^{\pm\mathrm{rnd}}$, by

$$\mathsf{Adv}_{\mathbb{E}}^{\pm\widetilde{\mathrm{rnd}}}(A) = |\Pr[K \xleftarrow{\$} \mathcal{K} : A^{\mathbb{E}_K(\cdot,\cdot),\mathbb{D}_K(\cdot,\cdot)} \Rightarrow 1] - |\Pr[A^{\$(\cdot,\cdot),\$^{-1}(\cdot,\cdot)} \Rightarrow 1]|. \tag{2.5}$$

where $\$(T, M)$ returns a random string of length $|M|$. We insist that A makes no pointless queries, regardless of oracle responses, and A asks no query (T, M) outside of $\mathcal{T} \times \mathcal{M}$.

Let A be an adversary and $\mathcal{R}$ be a list of resources for A and suppose $\mathsf{Adv}_{\Pi}^{\mathrm{xxx}}(A)$ already has been defined. We write $\mathsf{Adv}_{\Pi}^{\mathrm{xxx}}(\mathcal{R})$ for the maximal value of $\mathsf{Adv}_{\Pi}^{\mathrm{xxx}}(A)$ over all adversaries A that use resources at most $\mathcal{R}$. Resources of interest are the running time t and the number of oracle queries q and the query complexity σ_n (where $n \geq 1$ is a number). The query complexity σ_n specifies the number of blocks (each of size n-bits) of queries made by an adversary. We will be specially interested in query complexity of adversaries for tweakable enciphering schemes. In such cases, the query complexity of any one query (T, P) is $\lceil |P|/n \rceil + \lceil |T|/n \rceil$, and the query complexity of an adversary is the sum of the query complexity of all the queries.

Chapter 3

Inverse Free HCTR (IFHCTR)

HCTR was proposed by Wang, Feng and Wu in 2005 [9]. Later in 2008 a better security bound for HCTR was proved[14]. Inverse Free HCTR (IFHCTR) is the modification of HCTR scheme in which decryption module is not required. IFHCTR is a function $\mathbb{E} : \mathcal{K} \times \mathcal{T} \times \mathcal{M} \to \mathcal{M}$ where $\mathcal{M} = \cup_{i \geq 2n} \{0,1\}^i$ is the *message space* and $\mathcal{K} = \{0,1\}^n \times \{0,1\}^n \times \{0,1\}^n$ is the *key space* and $\mathcal{T} = \{0,1\}^*$ is the *tweak space*.

As in HCTR, IFHCTR uses a polynomial hash function and a counter mode of operation. Here we have modified the definition of hash function H_h, to incorporate use of arbitrary length tweaks. We remove "single" block cipher call in HCTR and replace it by a finite field multiplication. But for security we have to introduce two more block cipher calls, we encrypt outputs of the polynomial hashes before using them. Thus, for fixed length tweaks. IFHCTR requires one more block cipher call and one more finite field multiplication compared to HCTR. Also it uses one additional key. Thus IFHCTR requires, three keys, $m + 1$ block cipher calls and $2m + 2t + 1$ finite field multiplications for encrypting/decrypting a m blocks of message with t blocks of tweak.

3.1 Construction of IFHCTR

We first describe the two building blocks of IFHCTR, the hash function H and the counter mode Ctr.

Hash Function: In IFHCTR, we use hash a hash function $H : \{0,1\}^n \times \{0,1\}^i_{i \geq n} \times \mathcal{T} \to \{0,1\}^n$ defined as:

$$\begin{aligned} H_h(X,T) &= X_1 h^{m+t+1} \oplus X_2 h^{m+t} \oplus \ldots \oplus \mathsf{pad}_r(X_m) h^{t+2} \\ &\quad \oplus T_1 h^{t+1} \oplus \ldots \oplus \mathsf{pad}_r(T_t) h^2 \oplus \mathsf{bin}_{n/2}(|X|) || \mathsf{bin}_{n/2}(|T|) h \end{aligned}$$

FIGURE 3.1: IFHCTR algorithm

```
IFHCTR.Encrypt$^T_{K,h,\alpha}(P)$

1. $P_1||P_2||\dots||P_m \leftarrow P$
2. $MM \leftarrow P_1 \oplus E_K(H_h(P_2||\dots||P_m, T))$
3. $CC \leftarrow \alpha \times MM$
4. $S \leftarrow MM \oplus CC$
5. $(C_2, \dots, C_{m-1}, C_m)$
   $\leftarrow \mathsf{Ctr}_{K,S}(P_2, \dots, P_m)$
6. $C_1 \leftarrow CC \oplus E_K(H_h(C_2||\dots||C_m, T))$
return$(C_1||\dots||C_m)$;
```

```
IFHCTR.Decrypt$^T_{K,h,\alpha}(C)$

1. $C_1||C_2||\dots||C_m \leftarrow C$
2. $CC \leftarrow C_1 \oplus E_K(H_h(C_2||\dots||C_m, T))$
3. $MM \leftarrow \alpha^{-1} \times CC$
4. $S \leftarrow MM \oplus CC$
5. $(P_2, \dots, P_{m-1}, P_m)$
   $\leftarrow \mathsf{Ctr}_{K,S}(C_2, \dots, C_m)$
6. $P_1 \leftarrow MM \oplus E_K(H_h(P_2||\dots||P_m, T))$
return$(P_1||\dots||P_m)$;
```

where $h \in \{0,1\}^n$ is the hash key. $X = X_1||X_2||\dots||X_m$, where $|X_i| = n$ for $1 \leq i \leq m-1$ and $0 < |X_m| \leq n$. So, $|\mathsf{pad}_r(X_m)| = n$ and $T = T_1||T_2||\dots||T_t$, where $|T_i| = n$ for $1 \leq i \leq t-1$ and $0 < |T_t| \leq n$. We denote $H(h, X, T)$ by $H_h(X, T)$. H_h is different from the hash function of the original HCTR[9]. By the fact that we incorporate the tweak length in the definition to allow arbitrary length tweaks. Our definition of H_h is exactly the same as the hash function used in XCB[10].

COUNTER MODE: Counter mode which is used in case of IFHCTR defined as follows. Let $K, S \in \{0,1\}^n$ be given and $A_1, A_2, \dots, A_m \in \{0,1\}^n$ and A_m is a non-empty string of size at most n. Then we define.

$$\mathsf{Ctr}_{K,S}(A_1, \dots, A_m) = (A_1 \oplus E_K(S_1), \dots, A_m \oplus E_K(S_m)). \tag{3.1}$$

Where, S_i is defined as $S \oplus \mathsf{bin}_n(i)$. If last block is incomplete, i.e., if $A_m < n$, then $A_m \oplus E_K(S_m)$ is replaced by $A_m \oplus \mathsf{drop}_r(E_K(S_m))$ in Equation (3.1), where $r = n - |A_m|$.

The encryption and decryption algorithms for IFHCTR are described in Figure [3.1] and schematic diagrams for encryption and decryption are shown in Figures 3.2 and 3.3 respectively.

3.2 Characteristics of the IFHCTR

1. INVERSE FREE: This construction is inverse free as the inverse of the block cipher is never required. So encryption function of block cipher is sufficient for this construction. Since inverse of block cipher is not required then we can take a PRF assumption on the encryption function of a block cipher than SPRP assumption on the block cipher. So security of construction is based on a PRF assumption

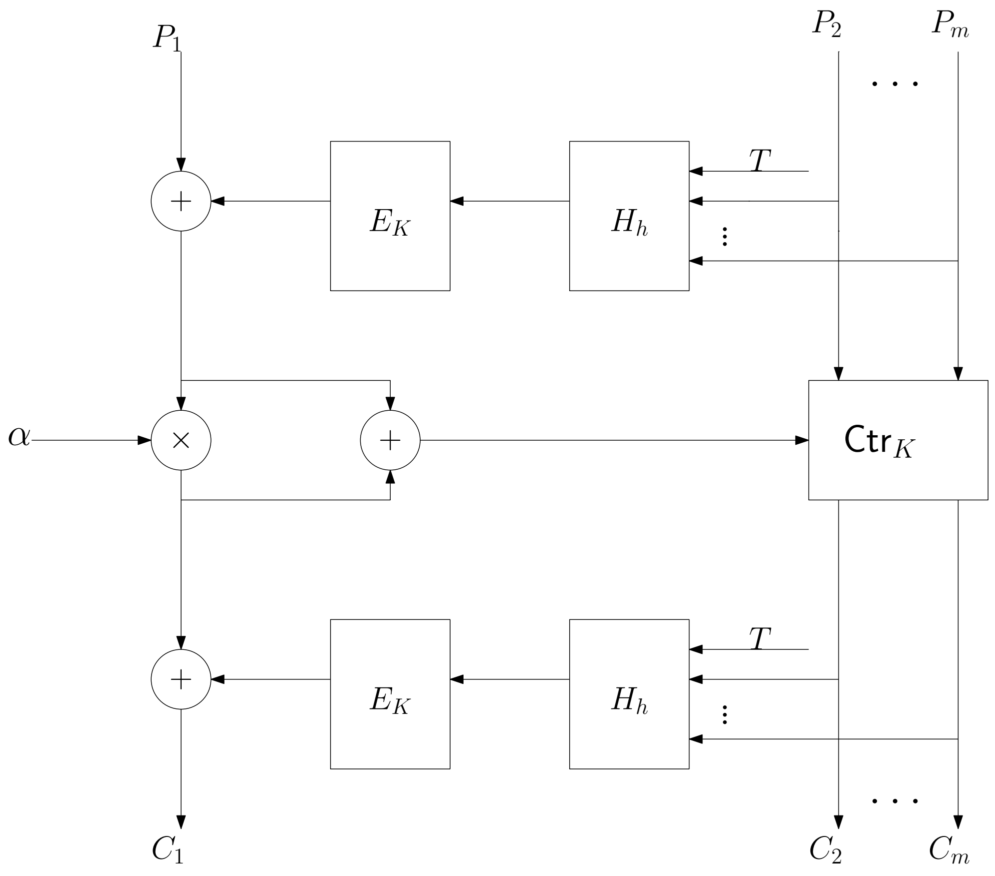

FIGURE 3.2: IFHCTR encryption

on the underlying block cipher, which is a weaker assumption than the SPRP assumption.

2. NUMBER OF FINITE FIELD MULTIPLICATIONS: For encrypting a m block message with a t block tweak, This construction requires $m + t$ finite field multiplications for evaluation of hash function in line 2 and line 6 and one more multiplication is required when α/α^{-1} is multiplied with MM/CC. Since in construction, evaluation of hash function is performed twice, it requires $2m + 2t + 1$ finite field multiplications.

3. NUMBER OF BLOCK CIPHER CALLS: This construction requires $m-1$ block cipher calls in Ctr mode and one block cipher calls after evaluation of hash function which is evaluated twice. So this construction requires $m + 1$ block cipher calls, while HCTR requires m block cipher calls.

4. NUMBER OF KEYS: This construction requires three keys, K for the block cipher, h for the hash function and α. All these three keys must be chosen uniformly and independently from $\{0, 1\}^n$.

5. MESSAGE LENGTH RESTRICTIONS: This construction works for variable length messages which are not necessarily multiples of the block length of block cipher.

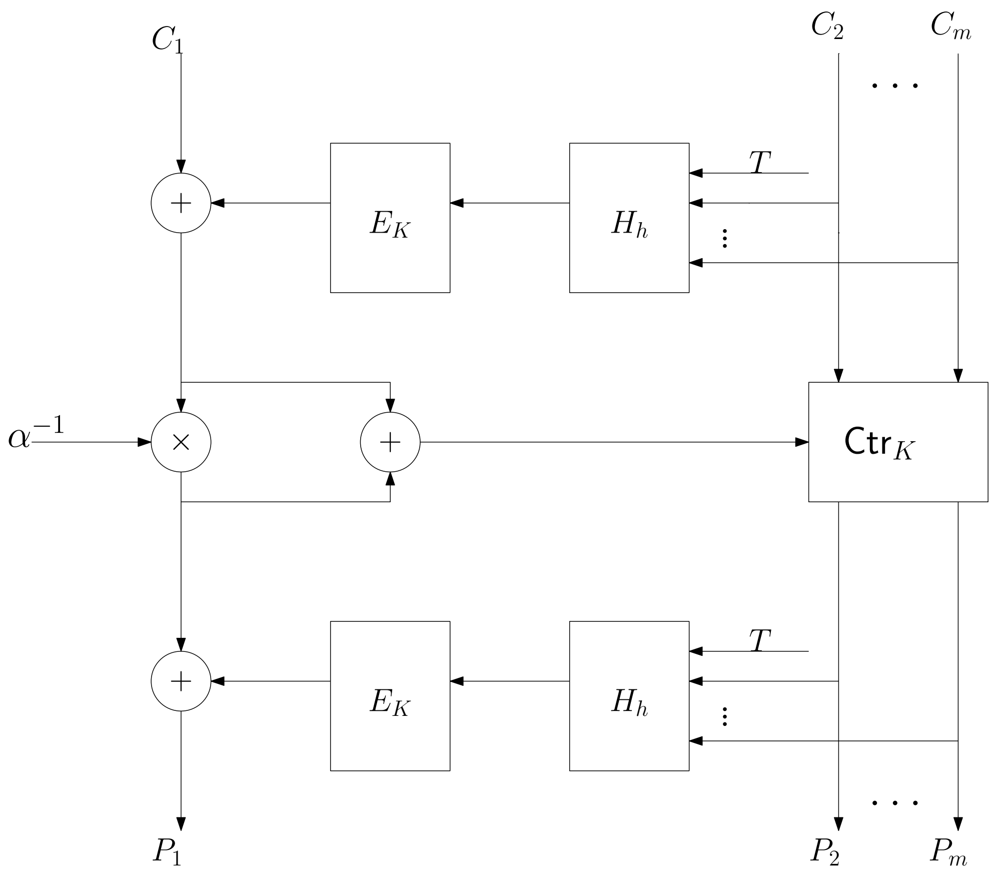

FIGURE 3.3: IFHCTR decryption

We are restricting message length should be at least $2n$ (at least two full blocks of message) to make the construction secure. There are attacks which have been found when message length is less than $2n$. Some of them are described in the Section 3.3.

6. TWEAK LENGTH: This construction works for variable length tweaks. We can query different length tweaks with message in different queries. While in HCTR if we fix the length of the tweak than we can't query different length tweak with message.

3.3 Some Insecure Constructions

We have thought about several constructions about IFHCTR before the finalization of construction shown in Figure 3.1, but they have turned to be insecure. We summarize some of the insecure versions with the hope that these can give some insights on the security of our final construction.

Here are few insecure constructions related HCTR/IFHCTR on which we have found out attacks. Because of those attacks we have imposed minimum query length restriction, used S as counter instead of CC and performed encryption after evaluating hash function.

length denotes $\mathsf{bin}_{n/2}(X)||\mathsf{bin}_{n/2}(T)$ where X is a message and T is a tweak.

Insecure Construction 1:

Why we have preformed encryption after evaluating hash function in IFHCTR?

First we have thought about construction in Figure 3.4 in which we are not performing encryption after evaluating hash function. But, we have found out key recovery attack on this construction.

FIGURE 3.4: Insecure construction-I

```
$\boldsymbol{IC1.Encrypt}^{T}_{K,h,\alpha}(P)$

1. $P_1||P_2||\ldots||P_m \leftarrow P$
2. $MM \leftarrow P_1 \oplus H_h(P_2||\ldots||P_m, T)$
3. $CC \leftarrow \alpha \times MM$
4. $S \leftarrow MM \oplus CC$
5. $(C_2, \ldots, C_{m-1}, C_m)$
   $\leftarrow \mathsf{Ctr}_{K,S}(P_2, \ldots, P_m)$
6. $C_1 \leftarrow CC \oplus H_h(C_2||\ldots||C_m, T)$
$\boldsymbol{return}(C_1||\ldots||C_m);$
```

```
$\boldsymbol{IC1.Decrypt}^{T}_{K,h,\alpha}(C)$

1. $C_1||C_2||\ldots||C_m \leftarrow C$
2. $CC \leftarrow C_1 \oplus H_h(C_2||\ldots||C_m, T)$
3. $MM \leftarrow \alpha^{-1} \times CC$
4. $S \leftarrow MM \oplus CC$
5. $(P_2, \ldots, P_{m-1}, P_m)$
   $\leftarrow \mathsf{Ctr}_{K,S}(C_2, \ldots, C_m)$
6. $P_1 \leftarrow MM \oplus H_h(P_2||\ldots||P_m, T)$
$\boldsymbol{return}(P_1||\ldots||P_m);$
```

For an attack consider an adversary with the following behaviour:

1. for i=1 to 3

   (a) Query $(P_1^i||0^n)$ to the encryption oracle and gets response $(C_1^i||C_2^i)$, where $P_1^i \neq P_1^j$ for each $i < j$

   As per construction, an adversary gets following equations:

2. for i=1 to 3

   (a) $C_1^i = \alpha P_1^i \oplus \alpha((\mathsf{length})h) \oplus C_2^i h^2 \oplus (\mathsf{length})h$

3. From above equations, an adversary gets two equations $C_1^1 \oplus C_1^2 = \alpha(P_1^1 \oplus P_1^2) \oplus (C_2^1 \oplus C_2^2)h^2$ and $C_1^1 \oplus C_1^3 = \alpha(P_1^1 \oplus P_1^3) \oplus (C_2^1 \oplus C_2^3)h^2$.

4. By solving these equations he retrieves two keys α and h.

To prevent this key recovery attack, we must have to do encryption after evaluating hash function.

Insecure Construction 2:

Why S is used as counter not CC in original HCTR?

We have thought using CC as initial counter value to counter mode instead of S in original HCTR construction showed in Figure 3.5. But, there is a key recovery attack, that we have found out.

Figure 3.5: Insecure construction-II

```
$\textbf{\textit{IC2.Encrypt}}^{T}_{K,h,\alpha}(P)$

1. $P_1||P_2||\ldots||P_m \leftarrow P$
2. $MM \leftarrow P_1 \oplus H_h(P_2||\ldots||P_m, T)$
3. $CC \leftarrow E_K(MM)$
4. $(C_2, \ldots, C_{m-1}, C_m)$
   $\leftarrow \mathsf{Ctr}_{K,CC}(P_2, \ldots, P_m)$
5. $C_1 \leftarrow CC \oplus H_h(C_2||\ldots||C_m, T)$
$\textbf{\textit{return}}(C_1||\ldots||C_m);$
```

```
$\textbf{\textit{IC2.Decrypt}}^{T}_{K,h,\alpha}(C)$

1. $C_1||C_2||\ldots||C_m \leftarrow C$
2. $CC \leftarrow C_1 \oplus H_h(C_2||\ldots||C_m, T)$
3. $MM \leftarrow E_K^{-1}(CC)$
4. $(P_2, \ldots, P_{m-1}, P_m)$
   $\leftarrow \mathsf{Ctr}_{K,CC}(C_2, \ldots, C_m)$
5. $P_1 \leftarrow MM \oplus H_h(P_2||\ldots||P_m, T)$
$\textbf{\textit{return}}(P_1||\ldots||P_m);$
```

For an attack consider an adversary with the following behavior:

1. Query $(C_1^1||C_2^1)$ to the decryption oracle and gets response $(P_1^1||P_2^1)$.

   Here counter value is CC^1, so $C_2^1 \oplus P_2^1 = E_K(CC^1 \oplus 1)$.
2. Query $(P_1^2||P_2^2)$=$(C_1^1 \oplus 1||C_2^1)$ to the encryption oracle and gets response $(C_1^2||C_2^2)$.

   Here counter value is $CC^2 = E_K(CC^1 \oplus 1)$. So $C_1^2 = CC^2 \oplus C_2^2 h^2 \oplus \mathsf{length}^2 h$, which forms quadratic equation and solving this equation adversary can retrieve h.

To prevent this attack it is necessary to use S as a counter value instead of CC.

Insecure Construction 3:

Why S is used as counter not CC in IFHCTR?

We have thought using CC as initial counter value to counter mode instead of S in IFHCTR construction showed in Figure 3.6. But, there is a distinguishing attack, that we have found out.

Distinguishing attack:

FIGURE 3.6: Insecure construction-III

IC3.Encrypt$^T_{K,h,\alpha}(P)$	***IC3.Decrypt***$^T_{K,h,\alpha}(C)$
1. $P_1\|\|P_2\|\|\ldots\|\|P_m \leftarrow P$	1. $C_1\|\|C_2\|\|\ldots\|\|C_m \leftarrow C$
2. $MM \leftarrow P_1 \oplus E_K(H_h(P_2\|\|\ldots\|\|P_m, T))$	2. $CC \leftarrow C_1 \oplus E_K(H_h(C_2\|\|\ldots\|\|C_m, T))$
3. $CC \leftarrow \alpha \times MM$	3. $MM \leftarrow \alpha^{-1} \times CC$
4. $(C_2, \ldots, C_{m-1}, C_m)$	4. $(P_2, \ldots, P_{m-1}, P_m)$
$\leftarrow \mathsf{Ctr}_{K,CC}(P_2, \ldots, P_m)$	$\leftarrow \mathsf{Ctr}_{K,CC}(C_2, \ldots, C_m)$
5. $C_1 \leftarrow CC \oplus E_K(H_h(C_2\|\|\ldots\|\|C_m, T))$	5. $P_1 \leftarrow MM \oplus E_K(H_h(P_2\|\|\ldots\|\|P_m, T))$
return$(C_1\|\|\ldots\|\|C_m)$;	***return***$(P_1\|\|\ldots\|\|P_m)$;

1. Query $(C_1^1||C_2^1||\cdots||C_m^1)$ to the decryption oracle and gets response $(P_1^1||P_2^1||\ldots||P_m^1)$.
2. Query $(C_1^2||C_2^2||\cdots||C_m^2)$ to the decryption oracle and gets response $(P_1^2||P_2^2||\ldots||P_m^2)$, where $C_1^2 = C_1^1 \oplus 1$ and $C_i^2 = C_i^1$ for $i = 2$ to m.

   Suppose CC^i is the counter value of i^{th} query, then $CC^2 = CC^1 \oplus 1$.

   An adversary defines $Z_i^j = P_i^j \oplus C_i^j$.
3. for all $j = 3$ to m.
   (a) An adversary checks whether $Z_i^1 = Z_{i-1}^2$ or not.

Insecure Construction 4:

Why we have restricted message query length should be at least $2n$ bits in IFHCTR?

There is a key recovery attack has been found when we allow the length of query is less than $2n$ bits in the construction described in Figure 3.1.

1. for $i = 1, 2$
   (a) Query $(P_1^i||0)$ to the encryption oracle and gets response $(C_1^i||C_2^i)$.

   Here an adversary gets equations as per our construction are

   $C_1^1 = \alpha P_1^1 \oplus \alpha E_K((\mathsf{length})h) \oplus E_K(\mathsf{pad}(C_2^1)h^2 \oplus (\mathsf{length})h)$ and $C_1^2 = \alpha P_1^2 \oplus \alpha E_K((\mathsf{length})h) \oplus E_K(\mathsf{pad}(C_2^2)h^2 \oplus (\mathsf{length})h)$.

   Length of C_2^1 and C_2^2 is one bit. So they will be same with probability 1/2. if they are same than $C_1^1 \oplus C_1^2 = \alpha(P_1^1 \oplus P_1^2)$.
2. So, an adversary recovers $\alpha = (C_1^1 \oplus C_1^2)/(P_1^1 \oplus P_1^2)$ with 1/2 probability in two queries.

Here problem is that there is only one bit of randomness in the second evaluation hash function which causes collision with 1/2 probability. This problem will be solved by adding at least n bits of randomness, in other words, restricting query length at least $2n$.

3.4 Security of IFHCTR

Theorem 3.1. *Fix n, σ_n to be positive integers and function $E : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$. Then,*

$$\mathsf{Adv}^{\pm \tilde{prp}}_{IFHCTR(Func(n))}(\sigma) \leq \frac{6.5\sigma^2}{2^n} \tag{3.2}$$

$$\mathsf{Adv}^{\pm \tilde{prp}}_{IFHCTR(E))}(\sigma, t) \leq \frac{6.5\sigma^2}{2^n} + \mathsf{Adv}^{prf}_{E}(\sigma, t') \tag{3.3}$$

where, $t' = t + O(\sigma)$ ■

To prove Theorem 3.1, we have done following reductions. First we define:

$$\begin{aligned} \mathsf{Adv}^{\pm\text{rnd}}_{\text{IFHCTR(Func}(n))}(A) &= \Pr[f \xleftarrow{\$} \text{Func}(n) : A^{\mathbb{E}_f, \mathbb{D}_f} \Rightarrow 1] \\ &\quad - \Pr[A^{\$(\cdot,\cdot),\$(\cdot,\cdot)} \Rightarrow 1]| \end{aligned} \tag{3.4}$$

$$\begin{aligned} \mathsf{Adv}^{\pm\tilde{\text{prp}}}_{\text{IFHCTR(Func}(n))}(A) &= \Pr[f \xleftarrow{\$} \text{Func}(n) : A^{\mathbb{E}_f, \mathbb{D}_f} \Rightarrow 1] \\ &\quad - \Pr[\pi \xleftarrow{\$} Perm^{\mathcal{T}}(\mathcal{M}) : A^{\pi(\cdot,\cdot),\pi^{-1}(.,.)} \Rightarrow 1] \\ &= \Pr[f \xleftarrow{\$} \text{Func}(n) : A^{\mathbb{E}_f, \mathbb{D}_f} \Rightarrow 1] \\ &\quad - \Pr[A^{\$(\cdot,\cdot),\$(\cdot,\cdot)} \Rightarrow 1] \\ &\quad + \Pr[A^{\$(\cdot,\cdot),\$(\cdot,\cdot)} \Rightarrow 1] \\ &\quad - \Pr[\pi \xleftarrow{\$} Perm^{\mathcal{T}}(\mathcal{M}) : A^{\pi(\cdot,\cdot),\pi^{-1}(.,.)} \Rightarrow 1] \\ &\leq \mathsf{Adv}^{\pm\text{rnd}}_{\text{IFHCTR(Func}(n))}(A) + \binom{q}{2}\frac{1}{2^n} \end{aligned} \tag{3.5}$$

To bound $\mathsf{Adv}^{\pm\text{rnd}}_{\text{IFHCTR(Func}(n))}$, we use a sequence of games as used in [7, 8, 14, 15] and Difference Lemma 3.2 and some properties of hash function H.

H is a special AXU (Almost Xor Universal) hash function. It has following property:

For any $X_1, X_2 \in \{0,1\}^*$, $Y \in \{0,1\}^n$ and $X_1 \neq X_2$, $H_h(X_1) \oplus H_h(X_2)$ is a nonzero polynomial in h without constant term. So $\Pr[h \xleftarrow{\$} \{0,1\}^n : H_h(X_1) \oplus H_h(X_2) = Y] \leq \ell/2^n$, where $\ell = \max\{|X|_n, |Y|_n\} + 1$. In other words, H is a $\ell/2^n$ -AXU hash function.

Lemma 3.2. (Difference Lemma): *Let A, B, F be events over some probability space such that $A \wedge \neg F \Leftrightarrow B \wedge \neg F$, then* $|\Pr(A) - \Pr(B)| \leq \Pr(F)$

Proof.

$$\begin{aligned}
|\Pr(A) - \Pr(B)| &= |\Pr(A \wedge F) + \Pr(A \wedge \neg F) - \Pr(B \wedge F) - \Pr(B \wedge \neg F)| \\
&= |\Pr(A \wedge F) - \Pr(B \wedge F)| \\
&= |\Pr(A|F)\Pr(F) - \Pr(B|F)\Pr(F)| \\
&= \Pr(F)|\Pr(A|F) - \Pr(B|F)| \\
&\leq \Pr(F)
\end{aligned}$$

■

In subsection 3.5.1 we prove that,

$$\mathsf{Adv}^{\pm\mathrm{rnd}}_{\mathrm{IFHCTR(Func}(n))}(\sigma) \leq \frac{6\sigma^2}{2^n} \tag{3.6}$$

3.5 Game Sequence

Game IFHCTR1: in IFHCTR1, the adversary interacts with $\mathbb{E}_f$ when f is a randomly chosen function from Func(n). Instead of initially choosing f, we build f in the following manner.

Intially f is assumed to be undefined everywhere. when $f(X)$ is required, but $f(X)$ is undefined then a random value is chosen from $\{0,1\}^n$.

The domain of f is maintained in set *Domain*. The game IFHCTR1 is shown in Figure 3.7. The figure shows the sub-routine Ch-f, the initialization steps and how the game responds to a encryption and decryption query. The i^{th} query of the adversary depends on its previous queries, the responses to those queries and on randomness of the adversary. As the game IFHCTR1 completely mimics the IFHCTR scheme instantiated with a uniform random function f we have.

$$\Pr[A^{\mathbb{E}_f, \mathbb{D}_f} \Rightarrow 1] = \Pr[A^{\mathrm{IFHCTR1}} \Rightarrow 1]. \tag{3.7}$$

Game RAND1: We modify IFHCTR1 by removing the boxed entries in IFHCTR1 and call the modified game as RAND1. By removing the boxed entries it cannot be guaranteed that f is a function as though we do the consistency checks, but we do not reset the value of Y (in Ch-f), the games IFHCTR1 and RAND1 are identical except when the bad flag is set. By using Lemma 3.2, we obtain

$$|\Pr[A^{\text{IFHCTR1}} \Rightarrow 1] - \Pr[A^{\text{RAND1}} \Rightarrow 1| \leq \Pr[A^{\text{RAND1}} \text{ sets } \mathsf{bad}]. \tag{3.8}$$

In line 106 Z_i^s gets set to random values for both encryption and decryption queries. Thus the adversary gets random values in response to both his encryption and decryption queries. Similarly in line 111 w_2^s gets set to a random value for an encryption query and w_1^s gets set to a random value for a decryption query.

$$\Pr[A^{\text{RAND1}} \Rightarrow 1] = \Pr[A^{\$(\cdot,\cdot),\$(\cdot,\cdot)} \Rightarrow 1]. \tag{3.9}$$

Thus using Equations (2.5), (3.7), (3.8) and (3.9) we have

$$\begin{aligned}
\mathsf{Adv}^{\pm\text{rnd}}_{\text{IFHCTR(Func}(n))} &= |\Pr[A^{\mathbb{E}_f,\mathbb{D}_f} \Rightarrow 1] - |\Pr[A^{\$(\cdot,\cdot),\$(\cdot,\cdot)} \Rightarrow 1]| \\
&= |\Pr[A^{\text{IFHCTR1}} \Rightarrow 1] - \Pr[A^{\text{RAND1}} \Rightarrow 1]| \qquad (3.10) \\
&\leq \Pr[A^{\text{RAND1}} \text{ sets } \mathsf{bad}]. \qquad (3.11)
\end{aligned}$$

Game RAND2: Now we make some subtle changes in the game RAND1 to get a new game RAND2 which is described in the Figure 3.8. In game RAND1 the function was not maintained and a call to the function was responded by returning random strings, so in Game RAND2 we no more use the subroutines Ch-f. The Game RAND2 returns random strings to the adversary in response to his encryption or decryption queries. Later in the finalization step we adjust variables and maintain multi set $\mathcal{D}$ that was supposed to be inputs of the function f. In the second phase of the finalization step, we check for collisions in the set $\mathcal{D}$. If collision occurs 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 of RAND1 sets bad is same as the probability of RAND2 sets bad. Thus we get:

$$\Pr[A^{\text{RAND1}} \text{ sets } \mathsf{bad}] = \Pr[A^{\text{RAND2}} \text{ sets } \mathsf{bad}] \tag{3.12}$$

Thus from Equations (3.11) and (3.12) we obtain

$$\mathsf{Adv}^{\pm\text{rnd}}_{\text{IFHCTR(Func}(n))} \leq \Pr[A^{\text{RAND2}} \text{ sets } \mathsf{bad}] \tag{3.13}$$

FIGURE 3.7: Games IFHCTR1 and RAND1

```
    Subroutine Ch-$f(X)$
11.     $Y \xleftarrow{\$} \{0,1\}^n$;
12.     **if** $X \in Domain$ **then**
        bad $\leftarrow$ true;
        $\boxed{Y \leftarrow f(X)}$;
        **endif**
13.     $f(X) \leftarrow Y$; $Domain \leftarrow Domain \cup \{X\}$; **return**$(Y)$;

    Initialization:
14.     **for** all $X \in \{0,1\}^n$ $f(X) \leftarrow$ undef **endfor**
15.     bad $\leftarrow$ false
```

Respond to the s^{th} query as follows: (Assume $l^s = n(m^s - 1) + r^s$, with $0 \le r^s < n$.)

Encipher query: Enc(T^s, P^s)

```
100.    parse $P^s$ as $X_1^s||X_2^s$ such that
        $X_1^s \leftarrow P_1^s$
        $X_2^s \leftarrow P_2^s||\ldots||P_m^s$
101.    **if** $(X_2^s, T^s) = (X_2^{s'}, T^{s'})$ for any $s' < s$
          $\mathsf{w}_1^{\mathsf{s}} \leftarrow \mathsf{w}_1^{\mathsf{s}'}$
        **else if** $(X_2^s, T^s) = (Y_2^{s'}, T^{s'})$ for any $s' < s$
          $\mathsf{w}_1^{\mathsf{s}} \leftarrow \mathsf{w}_2^{\mathsf{s}'}$
        **else**
          $\mathsf{w}_1^{\mathsf{s}} \leftarrow$ Ch-$f(H_h(X_2^s, T^s))$
102.    $MM^s \leftarrow P_1^s \oplus \mathsf{w}_1^{\mathsf{s}}$;
103.    $CC^s \leftarrow \alpha \times MM^s$;

104.    $S^s \leftarrow MM^s \oplus CC^s$;
105.    **for** $i = 1$ to $m^s - 2$,
106.      $Z_i^s \leftarrow$ Ch-$f(S^s \oplus \mathsf{bin}_n(i))$;
107.      $C_{i+1}^s \leftarrow P_{i+1}^s \oplus Z_i^s$;
108.    **end for**
109.    $Z_{m^s}^s \leftarrow$ Ch-$f(S^s \oplus \mathsf{bin}_n(m^s - 1))$;
110.    $C_{m^s}^s \leftarrow P_{m^s}^s \oplus \mathsf{drop}_{n-r^s}(Z_{m^s}^s)$;

111.    $\mathsf{w}_2^{\mathsf{s}} \leftarrow$ Ch-$f(H_h(C_2^s||\ldots||C_m^s, T^s))$
112.    $C_1^s \leftarrow CC^s \oplus \mathsf{w}_2^{\mathsf{s}}$;
113.    **return** $C_1^s||C_2^s||\ldots||C_{m^s}^s$
```

Decipher query: Dec(T^s, C^s)

```
        parse $C^s$ as $Y_1^s||Y_2^s$ such that
        $Y_1^s \leftarrow C_1^s$
        $Y_2^s \leftarrow C_2^s||\ldots||C_m^s$
        **if** $(Y_2^s, T^s) = (Y_2^{s'}, T^{s'})$ for any $s' < s$
          $\mathsf{w}_2^{\mathsf{s}} \leftarrow \mathsf{w}_2^{\mathsf{s}'}$
        **else if** $(Y_2^s, T^s) = (X_2^{s'}, T^{s'})$ for any $s' < s$
          $\mathsf{w}_2^{\mathsf{s}} \leftarrow \mathsf{w}_1^{\mathsf{s}'}$
        **else**
          $\mathsf{w}_2^{\mathsf{s}} \leftarrow$ Ch-$f(H_h(Y_2^s, T^s))$
        $CC^s \leftarrow C_1^s \oplus \mathsf{w}_2^{\mathsf{s}}$;
        $MM^s \leftarrow \alpha^{-1} \times CC^s$

        $S^s \leftarrow MM^s \oplus CC^s$;
        **for** $i = 1$ to $m^s - 2$,
          $Z_i^s \leftarrow$ Ch-$f(S^s \oplus \mathsf{bin}_n(i))$;
          $P_{i+1}^s \leftarrow C_{i+1}^s \oplus Z_i^s$;
        **end for**
        $Z_{m^s}^s \leftarrow$ Ch-$f(S^s \oplus \mathsf{bin}_n(m^s - 1))$;
        $P_{m^s}^s \leftarrow C_{m^s}^s \oplus \mathsf{drop}_{n-r^s}(Z_{m^s}^s)$;

        $\mathsf{w}_1^{\mathsf{s}} \leftarrow$ Ch-$f(H_h(P_2^s||\ldots||P_m^s, T^s))$
        $P_1^s \leftarrow MM^s \oplus \mathsf{w}_1^{\mathsf{s}}$;
        **return** $P_1^s||\ldots||P_{m^s}^s$
```

Figure 3.8: Game RAND2

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

Encipher query  Enc($T^s; P_1^s, P_2^s, \dots, P_{m^s}^s$)
    $ty^s = \mathsf{Enc}$; $C_1^s||C_2^s||\dots||C_{m^s-1}^s||D_{m^s}^s \xleftarrow{\$} \{0,1\}^{nm^s}$;
    $C_{m^s}^s \leftarrow \mathsf{drop}_{n-r^s}(D_{m^s})$ return $C_1^s||C_2^s||\dots||C_{m^s}^s$;

Decipher query  Dec($T^s; C_1^s, C_2^s, \dots, C_{m^s}^s$)
    $ty^s = \mathsf{Dec}$; $P_1^s||P_2^s||\dots||P_{m_s-1}^s||V_{m^s}^s \xleftarrow{\$} \{0,1\}^{nm^s}$;
    $P_{m^s}^s \leftarrow \mathsf{drop}_{n-r^s}(V_{m^s})$ return $P_1^s||P_2^s||\dots||P_{m^s}^s$;
```

Finalization:

```
Case $ty^s = \mathsf{Enc}$:

  parse $P^s$ as $X_1^s||X_2^s$ such that
  $X_1^s \leftarrow P_1^s$
  $X_2^s \leftarrow P_2^s||\dots||P_m^s$
  if  $(X_2^s, T^s) = (X_2^{s'}, T^{s'})$ for any $s' < s$
    $\mathsf{w}_1^{\mathsf{s}} = \mathsf{w}_1^{\mathsf{s}'}$
  else if $(X_2^s, T^s) = (Y_2^{s'}, T^{s'})$ for any $s' < s$
    $\mathsf{w}_1^{\mathsf{s}} \leftarrow \mathsf{w}_2^{\mathsf{s}'}$
  else
    $\mathsf{w}_1^{\mathsf{s}} \xleftarrow{\$} \{0,1\}^n$
    $y_1^s \leftarrow H_h(X_2^s, T^s)$
    $D \leftarrow D \cup \{y_1^s\}$

  parse $C^s$ as $Y_1^s||Y_2^s$ such that
  $Y_1^s \leftarrow C_1^s$
  $Y_2^s \leftarrow C_2^s||\dots||C_m^s$
  $y_2^s \leftarrow H_h(Y_2^s, T^s)$
  $D \leftarrow D \cup \{y_2^s\}$

  $MM^s \leftarrow P_1^s \oplus \mathsf{w}_1^{\mathsf{s}}$;
  $CC^s \leftarrow \alpha \times MM^s$;
  $S^s \leftarrow MM^s \oplus CC^s$;
  for $i = 2$ to $m^s$,
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{S^s \oplus \mathsf{bin}_n(i-1)\}$;
  end for
```

```
Case $ty^s = \mathsf{Dec}$:

  parse $C^s$ as $Y_1^s||Y_2^s$ such that
  $Y_1^s \leftarrow C_1^s$
  $Y_2^s \leftarrow C_2^s||\dots||C_m^s$
  if  $(Y_2^s, T^s) = (Y_2^{s'}, T^{s'})$ for any $s' < s$
    $\mathsf{w}_2^{\mathsf{s}} = \mathsf{w}_2^{\mathsf{s}'}$
  else if $(Y_2^s, T^s) = (X_2^{s'}, T^{s'})$ for any $s' < s$
    $\mathsf{w}_2^{\mathsf{s}} \leftarrow \mathsf{w}_1^{\mathsf{s}'}$
  else
    $\mathsf{w}_2^{\mathsf{s}} \xleftarrow{\$} \{0,1\}^n$
    $y_2^s \leftarrow H_h(Y_2^s, T^s)$
    $D \leftarrow D \cup \{y_2^s\}$

  parse $P^s$ as $X_1^s||X_2^s$ such that
  $X_1^s \leftarrow P_1^s$
  $X_2^s \leftarrow P_2^s||\dots||P_m^s$
  $y_1^s \leftarrow H_h(X_2^s, T^s)$
  $D \leftarrow D \cup \{y_1^s\}$

  $CC^s \leftarrow C_1^s \oplus \mathsf{w}_2^{\mathsf{s}}$;
  $MM^s \leftarrow \alpha^{-1} \times CC^s$;
  $S^s \leftarrow MM^s \oplus CC^s$;
  for $i = 2$ to $m^s$,
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{S^s \oplus \mathsf{bin}_n(i-1)\}$;
  end for
```

```
Second phase
    $\mathsf{bad} \leftarrow \mathsf{false}$;
    if (some value occurs more than once in $\mathcal{D}$)
        $\mathsf{bad} \leftarrow \mathsf{true}$
```

3.5.1 Bounding collision probability in $\mathcal{D}$

For an encryption query (T^s, P^s), we take $P^s = X_1^s || X_2^s$ where, $|X_1^s| = n$ and $|X_2^s| = |P^s| - n$. Similarly, for a decryption query (T^s, C^s), we take $C^s = Y_1^s || Y_2^s$ where, $|Y_1^s| = n$ and $|Y_2^s| = |C^s| - n$.

Here our goal is to bound the probability that two elements in the set $\mathcal{D}$ take the same value. After adversary asks q queries where the s^{th} query has m^s blocks of plaintext or ciphertext and t^s blocks of tweak, then the elements in set $\mathcal{D}$ can be written as follows:

1. $y_1^s = H_h(X_2^s, T^s)$, where $X_2^s = P_2^s \parallel \cdots \parallel P_{m^s}^s$.
2. $y_2^s = H_h(Y_2^s, T^s)$, where $Y_2^s = C_2^s \parallel \cdots \parallel C_{m^s}^s$.
3. $$S_j^s = S^s \oplus \mathsf{bin}_n(j) = \begin{cases} (\alpha \oplus 1)(P_1^s + \mathsf{w}_1^s) \oplus j & \text{if } ty^s = enc \\ (\alpha^{-1} \oplus 1)(C_1^s + \mathsf{w}_2^s) \oplus j & \text{if } ty^s = dec \end{cases}$$

   For, $1 \leq s \leq q$, $1 \leq j \leq m^s - 1$.

For making collision analysis smoother, let us identify the random variables based on which the probability of collision would be computed. In game RAND2, the outputs that adversary will get are uniformly distributed, and are independent of the previous queries asked by the adversary. This game is interactive but as the output of RAND2 is not dependent on the hash key h which is also uniformly distributed. So output of game RAND2 is independent of h.

Let's consider T^s as t^s many n bit blocks. we define $\sigma = \sum_s (m^s + t^s)$.
Now, Degree of X_2^s or Y_2^s is at most $m^s + t^s$, So we define $\ell^{s,s'} = \max(m^s + t^s, m^{s'} + t^{s'})$.

Claim 1: For $s \neq s'$, $\Pr[y_1^s = y_1^{s'}] \leq (\ell^{s,s'} + 1)/2^n$.

Proof. There are four sub cases to consider.

Case 1.1 $ty^s = ty^{s'} = enc$.

As Game RAND2 proceeds, it is guaranteed that $(X_2^s, T^s) \neq (X_2^{s'}, T^{s'})$. Thus, $H_h(X_2^s, T^s) \oplus H_h(X_2^{s'}, T^{s'})$ is a nonzero polynomial with degree at most $\ell^{s,s'}$. So by fundamental theorem of algebra,

$$\Pr[y_1^s = y_1^{s'}] \leq \frac{\ell^{s,s'}}{2^n}. \tag{3.14}$$

Case 1.2 $ty^s = ty^{s'} = dec$

Note, here both X_2^s and $X_2^{s'}$ are random strings.

Here there are two sub cases to consider.

Case 1.2.1 $|X_2^s| \neq |X_2^{s'}|$

Here $H_h(X_2^s, T^s) \oplus H_h(X_2^{s'}, T^{s'})$ is nonzero polynomial. So by fundamental theorem of algebra,

$$\Pr[y_1^s = y_1^{s'}] \leq \frac{\ell^{s,s'}}{2^n}. \tag{3.15}$$

Case 1.2.2 $|X_2^s| = |X_2^{s'}|$

Here there are two sub cases to consider:

Case 1.2.2.1 $|T^s| \neq |T^{s'}|$

Here $(X_2^s, T^s) \oplus (X_2^{s'}, T^{s'})$ is nonzero polynomial. So by fundamental theorem of algebra,

$$\Pr[y_1^s = y_1^{s'}] \leq \frac{\ell^{s,s'}}{2^n}. \tag{3.16}$$

Case 1.2.2.2 $|T^s| = |T^{s'}|$

$$\begin{aligned}
\Pr[y_1^s = y_1^{s'}] &= \Pr[y_1^s = y_1^{s'} | X_2^s = X_2^{s'}]\Pr[X_2^s = X_2^{s'}] \\
&\quad + \Pr[y_1^s = y_1^{s'} | X_2^s \neq X_2^{s'}]\Pr[X_2^s \neq X_2^{s'}] \\
&\leq 1 \times \frac{1}{2^{|X_2^s|}} + \frac{\ell^{s,s'}}{2^n}\left(1 - \frac{1}{2^{|X_2^s|}}\right) \\
&\leq \frac{\ell^{s,s'}}{2^n} + \frac{1}{2^{|X_2^s|}} \\
&\leq \frac{\ell^{s,s'}}{2^n} + \frac{1}{2^n} \\
&\leq \frac{\ell^{s,s'} + 1}{2^n}.
\end{aligned} \tag{3.17}$$

So, from Equations (3.15), (3.16) and (3.17) we get,

$$\Pr[y_1^s = y_1^{s'}] \leq \frac{\ell^{s,s'} + 1}{2^n}. \tag{3.18}$$

Case 1.3 $ty^s = enc$, $ty^{s'} = dec$

In this case proof is similar to Case 1.2. So,

$$\Pr[y_1^s = y_1^{s'}] \leq \frac{\ell^{s,s'} + 1}{2^n}. \tag{3.19}$$

Case 1.4 $ty^s = dec$, $ty^{s'} = enc$

In this case proof is similar to Case 1.3. So,

$$\Pr[y_1^s = y_1^{s'}] \leq \frac{\ell^{s,s'} + 1}{2^n}. \tag{3.20}$$

So, from Equations (3.14), (3.18), (3.19) and (3.20) we get,

$$\Pr[y_1^s = y_1^{s'}] \leq \frac{\ell^{s,s'} + 1}{2^n}. \tag{3.21}$$

■

Claim 2: For $s \neq s'$, $\Pr[y_2^s = y_2^{s'}] \leq (\ell^{s,s'} + 1)/2^n$.

Proof. Here, proof is similar to the proof of Claim 1. So,

$$\Pr[y_2^s = y_2^{s'}] \leq \frac{\ell^{s,s'} + 1}{2^n}. \tag{3.22}$$

■

Claim 3: $\Pr[y_1^s = y_2^{s'}] \leq (\ell^{s,s'} + 1)/2^n$.

Proof. Here, there are four sub cases to consider.

Case 3.1 $ty^s = ty^{s'} = enc$

In this case proof is similar to Case 1.2. So,

$$\Pr[y_1^s = y_2^{s'}] \leq \frac{\ell^{s,s'} + 1}{2^n}. \tag{3.23}$$

Case 3.2 $ty^s = ty^{s'} = dec$,

In this case proof is similar to Case 1.2. So,

$$\Pr[y_1^s = y_2^{s'}] \leq \frac{\ell^{s,s'} + 1}{2^n}. \tag{3.24}$$

Case 3.3 $ty^s = enc$, $ty^{s'} = dec$

In this case proof is similar to Case 1.1. So,

$$\Pr[y_1^s = y_2^{s'}] \leq \frac{\ell^{s,s'}}{2^n}. \tag{3.25}$$

Case 3.4 $ty^s = dec$, $ty^{s'} = enc$

In this case proof is similar to Case 1.2. So,

$$\Pr[y_1^s = y_2^{s'}] \leq \frac{\ell^{s,s'} + 1}{2^n}. \tag{3.26}$$

From Equations (3.23), (3.24),(3.25) and (3.26) we get,

$$\Pr[y_1^s = y_2^{s'}] \leq \frac{\ell^{s,s'} + 1}{2^n}. \tag{3.27}$$

■

Claim 4: $\Pr[y_1^s = S_j^{s'}] \leq (m^s + t^s)/2^n$.

Proof. For finding collision probability between y_1^s and $S_j^{s'}$. Here, $y_1^s = H_h(X_2^s, T^s)$ and $S_j^{s'} = (\alpha \oplus 1)(P_1^{s'} \oplus \mathsf{w}_1^{s'}) \oplus j$ if $ty^{s'} = enc$.

$$\begin{aligned}
\Pr[y_1^s = S_j^{s'}] &= \Pr[H_h(X_2^s, T^s) = (\alpha \oplus 1)(P_1^{s'} \oplus \mathsf{w}_1^{s'}) \oplus j] \\
&= \sum_{x \in \{0,1\}^n} \Pr[\alpha = x] \times \Pr[H_h(X_2^s, T^s) = (\alpha \oplus 1)(P_1^{s'} \oplus \mathsf{w}_1^{s'}) \oplus j | \alpha = x] \\
&\leq \frac{1}{2^n} \times \frac{2^n(m^s + t^s)}{2^n} \\
&\leq \frac{m^s + t^s}{2^n}.
\end{aligned} \tag{3.28}$$

Similarly for $ty^{s'} = dec$, $\Pr[y_1^s = S_j^{s'}] \leq (m^s + t)/2^n$. ■

Claim 5: $\Pr[y_2^s = S_j^{s'}] \leq (m^s + t^s)/2^n$.

Proof. Here, proof is similar to the proof of Claim 4. So,

$$\Pr[y_2^s = S_j^{s'}] \quad \leq \quad \frac{m^s + t^s}{2^n}. \tag{3.29}$$

■

Claim 6: For $(s, j) \neq (s', j')$, $\Pr[S_j^s = S_{j'}^{s'}] \leq 3/2^n$.

Proof. Here, there are three sub cases to consider.

Case 6.1 $ty^s = ty^{s'} = enc$

There are two sub cases to consider.

Case 6.1.1 $(X_2^s, T^s) = (X_2^{s'}, T^{s'})$

if $(X_2^s, T^s) = (X_2^{s'}, T^{s'})$ then, $\mathsf{w}_1^s = \mathsf{w}_1^{s'}$ but, $P_1^s \neq P_1^{s'}$

$$\begin{aligned}
\Pr[S_j^s = S_{j'}^{s'}] &= \Pr[(\alpha \oplus 1)(P_1^s \oplus \mathsf{w}_1^s) \oplus j = (\alpha \oplus 1)(P_1^{s'} \oplus \mathsf{w}_1^{s'}) \oplus j'] \\
&= \Pr[(\alpha \oplus 1)(P_1^s \oplus P_1^{s'}) \oplus (j \oplus j') = 0] && (3.30) \\
&= \frac{1}{2^n}. && (3.31)
\end{aligned}$$

Since Equation (3.30) is nonzero polynomial which has solution for one choice of α. So $\Pr[S_j^s = S_{j'}^{s'}] = 1/2^n$

Case 6.1.2 $(X_2^s, T^s) \neq (X_2^{s'}, T^{s'})$

If $(X_2^s, T^s) \neq (X_2^{s'}, T^{s'})$ then, w_1^s and $\mathsf{w}_1^{s'}$ are two independent n bit strings. So,

$$\begin{aligned}
\Pr[S_j^s = S_{j'}^{s'}] &= \Pr[(\alpha \oplus 1)(P_1^s \oplus \mathsf{w}_1^s) \oplus j = (\alpha \oplus 1)(P_1^{s'} \oplus \mathsf{w}_1^{s'}) \oplus j'] \\
&= \Pr[(\alpha \oplus 1)(P_1^s \oplus P_1^{s'} \oplus \mathsf{w}_1^s \oplus \mathsf{w}_1^{s'}) \oplus (j \oplus j') = 0] && (3.32)
\end{aligned}$$

If Equation (3.32) is a zero polynomial, then this equation is already satisfied. This case occurs with probability at most $1/2^n$. If Equation (3.32) is a nonzero polynomial then this equation has solution for at most one choice of α. Let's take $P(\alpha)$ as a polynomial in Equation in (3.32) and E be an event that $P(\alpha)$ is a nonzero polynomial. Thus we can write,

$$\begin{aligned}
\Pr[P(\alpha) = 0] &= \Pr[P(\alpha) = 0 | E]\Pr[E] + \Pr[P(\alpha) = 0 | \overline{E}]\Pr[\overline{E}] \\
&\leq \frac{1}{2^n}\left(1 - \frac{1}{2^n}\right) + 1\left(\frac{1}{2^n}\right) \\
&\leq \frac{1}{2^n} + \frac{1}{2^n} \\
&\leq \frac{2}{2^n} && (3.33)
\end{aligned}$$

From Equations (3.31) and (3.33),

$$\Pr[S_j^s = S_{j'}^{s'}] \leq \frac{2}{2^n}. \quad (3.34)$$

Case 6.2 $ty^s = ty^{s'} = dec$

Here, proof is similar to Case 6.1. So,

$$\Pr[S_j^s = S_{j'}^{s'}] = \frac{1}{2^n}. \quad (3.35)$$

■

Case 6.3 $ty^s = enc$, $ty^{s'} = dec$

Proof. There are two sub cases to consider.

Case 6.3.1: $(X_2^s, T^s) = (Y_2^s, T^s)$

If $(P_2^s, T^s) = (C_2^s, T^s)$ then, $\mathsf{w}_1^s = \mathsf{w}_2^{s'}$.

$$\begin{aligned}
\Pr[S_j^s = S_{j'}^{s'}] &= \Pr[(\alpha \oplus 1)(P_1^s \oplus \mathsf{w}_1^s) \oplus j = (\alpha^{-1} \oplus 1)(C_1^{s'} \oplus \mathsf{w}_2^{s'}) \oplus j'] \\
&= \Pr[(\alpha^2 \oplus \alpha)(P_1^s \oplus \mathsf{w}_1^s) \oplus \alpha j = (1 \oplus \alpha)(C_1^{s'} \oplus \mathsf{w}_2^{s'}) \oplus \alpha j'] \\
&= \Pr[\alpha^2(P_1^s \oplus \mathsf{w}_1^s) \oplus \alpha(P_1^s \oplus C_1^{s'} \oplus j \oplus j') \oplus C_1^{s'} \oplus \mathsf{w}_2^{s'} = 0] \quad (3.36)
\end{aligned}$$

If Equation (3.36) is a zero polynomial, then this equation is already satisfied. This case occurs with probability at most $1/2^n$. If Equation (3.36) is a nonzero polynomial then this equation has solution for at most two choice of α. Let's take $P(\alpha)$ as a polynomial in Equation in (3.36) and E be an event that $P(\alpha)$ is a nonzero polynomial. Thus we can write,

$$\begin{aligned}
\Pr[P(\alpha) = 0] &= \Pr[P(\alpha) = 0|E]\Pr[E] + \Pr[P(\alpha) = 0|\overline{E}]\Pr[\overline{E}] \\
&\le \frac{2}{2^n}\left(1 - \frac{1}{2^n}\right) + 1\left(\frac{1}{2^n}\right) \\
&\le \frac{2}{2^n} + \frac{1}{2^n} \\
&\le \frac{3}{2^n} \quad (3.37)
\end{aligned}$$

So,

$$\Pr[S_j^s = S_{j'}^{s'}] \le \frac{3}{2^n} \quad (3.38)$$

Case 6.3.2: $(X_2^s, T^s) = (Y_2^s, T^s)$

If $ty^s = enc, ty^{s'} = dec$ then, w_1^s and $\mathsf{w}_2^{s'}$ are two independent n bit strings.

$$\begin{aligned}
\Pr[S_j^s = S_{j'}^{s'}] &= \Pr[(\alpha \oplus 1)(P_1^s \oplus \mathsf{w}_1^s) \oplus j = (\alpha^{-1} \oplus 1)(C_1^{s'} \oplus \mathsf{w}_2^{s'}) \oplus j'] \\
&= \Pr[(\alpha^2 \oplus \alpha)(P_1^s \oplus \mathsf{w}_1^s) \oplus \alpha j = (1 \oplus \alpha)(C_1^{s'} \oplus \mathsf{w}_2^{s'}) \oplus \alpha j'] \\
&= \Pr[\alpha^2(P_1^s \oplus \mathsf{w}_1^s) \oplus \alpha(P_1^s \oplus C_1^{s'} \oplus \mathsf{w}_1^s \oplus \mathsf{w}_2^{s'} \oplus j \oplus j') \\
&\quad \oplus C_1^{s'} \oplus \mathsf{w}_2^{s'} = 0] \quad (3.39)
\end{aligned}$$

If Equation (3.39) is a zero polynomial, then this equation is already satisfied. This case occurs with probability at most $1/2^n$. If Equation (3.39) is a nonzero polynomial then this equation has solution for at most two choice of α. Let's take $P(\alpha)$ as a polynomial in Equation in (3.39) and E be an event that $P(\alpha)$ is a nonzero polynomial. Thus we can write,

$$\begin{aligned}\Pr[P(\alpha)=0] &= \Pr[P(\alpha)=0|E]\Pr[E]+\Pr[P(\alpha)=0|\overline{E}]\Pr[\overline{E}]\\ &\le \frac{2}{2^n}\left(1-\frac{1}{2^n}\right)+1\left(\frac{1}{2^n}\right)\\ &\le \frac{2}{2^n}+\frac{1}{2^n}\\ &\le \frac{3}{2^n} \end{aligned} \tag{3.40}$$

So,

$$\Pr[S_j^s = S_{j'}^{s'}] \le \frac{3}{2^n} \tag{3.41}$$

From Equations (3.38) and (3.41),

$$\Pr[S_j^s = S_{j'}^{s'}] \le \frac{3}{2^n}. \tag{3.42}$$

From Equations (3.34),(3.35) and (3.42),

$$\Pr[S_j^s = S_{j'}^{s'}] \le \frac{3}{2^n}. \tag{3.43}$$

■

Suppose COL is an event that in RAND2 bad flag sets true. τ denotes $\Sigma_{1\le s\le q} t^s$. Therefore,

$$\begin{aligned}\Pr[\mathsf{COL}] \le &\sum_{1\le s<s'\le q}\frac{\ell^{s,s'}+1}{2^n}+\sum_{1\le s<s'\le q}\frac{\ell^{s,s'}+1}{2^n}+\sum_{1\le s,s'\le q}\frac{\ell^{s,s'}+1}{2^n}\\ &+\sum_{1\le s\le q}(\sigma-\tau-q)\frac{m^s+t^s}{2^n}+\sum_{1\le s\le q}(\sigma-\tau-q)\frac{m^s+t^s}{2^n}\\ &+\binom{\sigma-\tau-q}{2}\frac{3}{2^n}. \end{aligned} \tag{3.44}$$

Let's bound the value of $\ell^{s,s'}$. As per sum bound we can write as,

$$\begin{aligned}
\ell^{s,s'} &= \max(m^s + t^s, m^{s'} + t^{s'}) \\
&\leq m^s + t^s + m^{s'} + t^{s'}.
\end{aligned}$$

Let's bound the value of $\Sigma_{1\leq s<s'\leq q}(\ell^{s,s'} + 1)$.

$$\begin{aligned}
\sum_{1\leq s<s'\leq q}(\ell^{s,s'} + 1) &\leq \sum_{1\leq s<s'\leq q}(m^s + m^{s'} + t^s + t^{s'} + 1) \\
&\leq (q-1)\sum_{1\leq s\leq q}(m^s + t^s) + \frac{q(q-1)}{2} \\
&\leq (q-1)\sigma + \frac{q^2}{2} \\
&\leq \frac{\sigma^2}{2} + \frac{\sigma^2}{8} \\
&\leq \frac{5\sigma^2}{8}. \qquad (3.45)
\end{aligned}$$

Similarly to bound the value of $\Sigma_{1\leq s,s'\leq q}(\ell^{s,s'} + 1)$,

$$\begin{aligned}
\sum_{1\leq s,s'\leq q}(\ell^{s,s'} + 1) &\leq \sum_{1\leq s,s'\leq q}(m^s + m^{s'} + t^s + t^{s'} + 1) \\
&\leq 2q\sum_{1\leq s\leq q}(m^s + t^s) + q^2 \\
&\leq 2q\sigma + q^2 \\
&\leq \sigma^2 + \frac{\sigma^2}{4} \\
&\leq \frac{5\sigma^2}{4}. \qquad (3.46)
\end{aligned}$$

From Equations (3.44), (3.45) and (3.46) we get,

$$\begin{aligned}
\Pr[\mathsf{COL}] &\leq 2\left(\frac{5\sigma^2}{8}\right)\left(\frac{1}{2^n}\right) + \left(\frac{5\sigma^2}{4}\right)\left(\frac{1}{2^n}\right) + 2\left(\frac{\sigma^2}{2^n}\right) + \left(\frac{\sigma^2}{2}\right)\left(\frac{3}{2^n}\right) \\
&\leq \frac{6\sigma^2}{2^n}.
\end{aligned}$$

So,

$$\mathsf{Adv}^{\pm\mathrm{rnd}}_{\mathrm{IFHCTR[Func(n)]}} \leq \frac{6\sigma^2}{2^n}. \qquad (3.47)$$

Chapter 4

Implementation of IFHCTR

Software implementation of IFHCTR are built from the implementation of AES and the implementation of polynomial hash function. The AES based parts consist of the Ctr mode while the hash function are built from Horner method. Here encryption function of AES is used to instantiate Ctr mode. Three keys which are used in this construction are chosen independently and stored. Inverse of α which is α^{-1} is calculated using math sage and also stored.

A very efficient software implementation of a TES called FAST is done in [16]. The implementation in [16] contains all the basic modules required for implementing IFHCTR. We have used them in the implementation. The implementation in [16] uses the AES-NI instructions for computing AES and polynomial multiplication. This table shows comparison of IFHCTR with HCTR and FAST constructions in terms of cycles per byte.

FAST[Horner]	1.46
HCTR	1.40
IFHCTR	**1.42**

TABLE 4.1: Comparison of the cycles per byte measure of HCTR with original HCTR and FAST

Conclusion

In this dissertation, we proposed length preserving tweakable enciphering scheme called as inverse free HCTR, which is the modification of HCTR scheme in which the inverse function of an underlying block cipher is not needed. We proved IFHCTR is a strong tweakable pseudorandom permutation (sprp), when the underlying blockcipher is a pseudorandom function (prf). We also implemented IFHCTR in modern Intel Processors equipped with the AES-NI instruction set and compared it's performance against HCTR and FAST schemes.

Bibliography

[1] Shai Halevi and Phillip Rogaway. A parallelizable enciphering mode. In Tatsuaki Okamoto, editor, *Topics in Cryptology – CT-RSA 2004*, pages 292–304, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg. ISBN 978-3-540-24660-2.

[2] Shai Halevi. Eme*: Extending eme to handle arbitrary-length messages with associated data. In Anne Canteaut and Kapaleeswaran Viswanathan, editors, *Progress in Cryptology - INDOCRYPT 2004*, pages 315–327, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. ISBN 978-3-540-30556-9.

[3] Shai Halevi and Phillip Rogaway. A tweakable enciphering mode. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003*, pages 482–499, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg. ISBN 978-3-540-45146-4.

[4] Ritam Bhaumik and Mridul Nandi. An inverse-free single-keyed tweakable enciphering scheme. In *Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part II*, pages 159–180, 2015. doi: 10.1007/978-3-662-48800-3_7. URL https://doi.org/10.1007/978-3-662-48800-3_7.

[5] Viet Tung Hoang, Ted Krovetz, and Phillip Rogaway. Robust authenticated-encryption: Aez and the problem that it solves. Cryptology ePrint Archive, Report 2014/793, 2014. https://eprint.iacr.org/2014/793.

[6] Debrup Chakraborty and Palash Sarkar. A new mode of encryption providing a tweakable strong pseudo-random permutation. *IACR Cryptology ePrint Archive*, 2006:275, 2006.

[7] Shai Halevi. Invertible universal hashing and the tet encryption mode. In Alfred Menezes, editor, *Advances in Cryptology - CRYPTO 2007*, pages 412–429, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. ISBN 978-3-540-74143-5.

[8] Palash Sarkar. Improving upon the tet mode of operation. In Kil-Hyun Nam and Gwangsoo Rhee, editors, *Information Security and Cryptology - ICISC 2007*, pages

180–192, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. ISBN 978-3-540-76788-6.

[9] Peng Wang, Dengguo Feng, and Wenling Wu. HCTR: A variable-input-length enciphering mode. In *Information Security and Cryptology, First SKLOIS Conference, CISC 2005, Beijing, China, December 15-17, 2005, Proceedings*, pages 175–188, 2005. doi: 10.1007/11599548_15. URL https://doi.org/10.1007/11599548_15.

[10] David A. McGrew and Scott 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*, pages 311–327, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. ISBN 978-3-540-77360-3.

[11] Debrup Chakraborty, Sebati Ghosh, Cuauhtemoc Mancillas Lopez, and Palash Sarkar. Fast: Disk encryption and beyond. Cryptology ePrint Archive, Report 2017/849, 2017. https://eprint.iacr.org/2017/849.

[12] D. Chakraborty, C. Mancillas-Lpez, and P. Sarkar. Stes: A stream cipher based low cost scheme for securing stored data. *IEEE Transactions on Computers*, 64(9): 2691–2707, Sept 2015. ISSN 0018-9340. doi: 10.1109/TC.2014.2366739.

[13] Palash Sarkar. Tweakable enciphering schemes using only the encryption function of a block cipher. *Information Processing Letters*, 111(19):945 – 955, 2011. ISSN 0020-0190. doi: https://doi.org/10.1016/j.ipl.2011.06.014. URL http://www.sciencedirect.com/science/article/pii/S0020019011001852.

[14] Debrup Chakraborty and Mridul Nandi. An improved security bound for HCTR. In *Fast Software Encryption, 15th International Workshop, FSE 2008, Lausanne, Switzerland, February 10-13, 2008, Revised Selected Papers*, pages 289–302, 2008. doi: 10.1007/978-3-540-71039-4_18. URL https://doi.org/10.1007/978-3-540-71039-4_18.

[15] D. Chakraborty and P. Sarkar. Hch: A new tweakable enciphering scheme using the hash-counter-hash approach. *IEEE Transactions on Information Theory*, 54 (4):1683–1699, April 2008. ISSN 0018-9448. doi: 10.1109/TIT.2008.917623.

[16] URL https://github.com/sebatighosh/FAST.