TIME MEMORY TRADE OFF ATTACK ON SYMMETRIC CIPHERS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF APPLIED MATHEMATICS
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

A. NURDAN SARAN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF PHILOSOPHY OF DOCTORATE
IN
CRYPTOGRAPHY

FEBRUARY 2009

Approval of the thesis:

TIME MEMORY TRADE OFF ATTACK ON SYMMETRIC CIPHERS

submitted by **A. NURDAN SARAN** in partial fulfillment of the requirements for the degree of **Philosophy of Doctorate in Department of Cryptography, Middle East Technical University** by,

ABSTRACT

TIME MEMORY TRADE OFF ATTACK ON SYMMETRIC CIPHERS

Saran, A. Nurdan
Ph.D., Department of Cryptography
Supervisor : Assoc. Prof. Dr. Ali Doğanaksoy

February 2009, 94 pages

Time Memory Trade Off (TMTO) is a cryptanalytic method that aims to develop an attack which has a lower memory complexity than lookup table and a lower online time complexity than exhaustive search. TMTO methods are widely studied in the literature and used for inverting various cryptosystems. We focus on the design and the analysis of TMTO on symmetric ciphers in this thesis. Firstly, the summary of the random mapping statistics from the view point of TMTO is presented. We also recalculate some expected values with a simpler approach than the existing proofs. Then, we propose some variant constructions and also present three new distinguishers based on random mappings. Next, we provide a detailed analysis of the success rate of two main improvements of the attack; Distinguished Point Method and Rainbow Method. Finally, we discuss the adjustment of the parameters to achieve a high success rate. To support our theoretical framework, we also present empirical results of our analysis to actual ciphers.

Keywords: Time Memory Trade Off, Success Rate, Random Mappings, Symmetric Key, Cryptanalysis

ÖZ

SİMETRİK ŞİFRELER ÜZERİNE ZAMAN HAFIZA ÜLEŞİMİ ATAĞI

Saran, A. Nurdan
Doktora, Kriptografi Bölümü
Tez Yöneticisi : Doç. Dr. Ali Doğanaksoy

Şubat 2009, 94 sayfa

Zaman Hafıza Üleşimi (ZHÜ), amacı başvuru tablosundan daha az hafıza kullanan ve tam kapsamlı aramadan daha az zaman harcayan saldırılar geliştirmek olan bir kriptanaliz yöntemidir. ZHÜ literatürde geniş ölçüde çalışılmış ve bir çok kriptosistemin kırılmasında kullanılmıştır. Bu çalışmada, simetrik şifrelerin tasarım ve analizleri üzerine odaklanılmıştır. Öncelikle, rastlantısal dönüşümlerin istatistiksel özellikleri göz önünde bulundurularak bu methodun uygulanmasında gerekli olan bazı parametrelerin beklenen değerleri daha önce verilen yöntemlerden daha basit bir yaklaşımla yeniden hesaplanarak listelenmiştir. Bilinen tekniklere bazı yeni yapılandırmalar önerilmiş ve rastlantısal dönüşümler üzerinde üç yeni belirleyici tanımlanmıştır. Daha sonra, atağın iki önemli iyileştirmesi olan gökkuşağı (Rainbow) yöntemi ve ayırt edilmiş nokta (Distinguished Point) yöntemlerinin başarı oranları ayrıntılı olarak analiz edilmiştir. Bu sonuçları kullanarak yüksek başarı elde edilebilmesi için parametrelerin nasıl seçilmesi gerektiği üzerinde tartışılmıştır. Yaptığımız teorik çalışmaları desteklemek için güncel şifre algoritmaları üzerinde deneysel sonuçlar sunulmuştur.

Anahtar Kelimeler: Zaman Hafıza Üleşimi , Başarı Oranı, Simetrik Anahtar, Kriptanaliz

To my father, Hasan Buz
my husband, Murat Saran
and my little son, Onur Saran

ACKNOWLEDGMENTS

This thesis was done under the supervision of Assoc. Dr. Prof. Ali Doğanaksoy in the Institute of Applied Mathematics. I owe my deepest gratitute to Ali Doğanaksoy for his insightful support. Without his sincere and patient effort, this work would never have been completed. I would also like to express my special thanks to examining committee members.

I am also grateful to all my friends in the Institute of Applied Mathematics. I am obliged to Çağdaş Çalık, he supported me by writing the codes of this thesis. And I am also pleased to thank Dr. Meltem Sönmez Turan for her friendship and friutful discussions.

I am so lucky that I have a sacrificing family. Especially, I am under obligation to two special mothers: my mother, Enise Buz, and my husband's mother, Şener Saran, who looked after my lovely baby, Onur Saran, when I was studying. And also my sister, Ebru Buz spent many of her weekends by doing my works. Last, but of course not least, I am indebted my husband for his love and also for his valuable contributions. Although the subject is quite far from his research interest, with a great patience, he examined this thesis for grammar and offered suggestions for improvement.

TABLE OF CONTENTS

ABSTRACT iv

ÖZ v

DEDICATION vi

ACKNOWLEDGMENTS vii

TABLE OF CONTENTS viii

LIST OF TABLES xii

LIST OF FIGURES xiii

CHAPTERS

1 INTRODUCTION 1

1.1 Symmetric Key Cryptography 2

1.1.1 Block Ciphers 2

1.1.2 Stream Ciphers 3

1.1.3 Security Analysis of Symmetric Ciphers 3

1.2 One way Functions 4

1.2.1 Some Generic Attacks on One-way Functions 4

1.3 Outline and Main Contributions of This Thesis 6

2 RANDOM MAPPINGS 8

2.1 Introduction 8

2.2 Graph Representation of Functions 8

2.3 Random Mapping Statistics 11

2.4 Random Permutations 15

3 TIME MEMORY TRADE OFF 17

3.1 Introduction 17

3.2 Hellman's Construction 18
3.2.1 False Alarm 23
3.2.2 Parameters 24
3.2.3 Hellman's Curve 30
3.3 TMTO on Block Ciphers 30
3.3.1 Distinguished Point (DP) 30
3.3.1.1 Variants of the Distinguished Point Method . 36
3.3.1.2 Distinguished Point Table on Permutations . . 37
3.3.2 Rainbow Table 39
3.3.2.1 Rainbow-Like Schemes 43
3.3.2.2 False alarm detection using Checkpoints . . . 44
3.3.3 Perfect Tables 44
3.3.3.1 Perfect Hellman 45
3.3.3.2 Perfect Tables with Threshold 46
3.3.4 Time/ Memory/Key (TMK) Tradeoff 47
3.3.4.1 Modes of Operations 48
3.3.5 FPGA implementations 49
3.4 TMTO on Stream Ciphers 49
3.4.1 Babbage-Golic Attack 50
3.4.2 Biryukov-Shamir Attack- TMD Trade Off 51
3.4.3 Extensions and Refinements 52
3.4.3.1 Biryukov-Shamir-Wagner Attack 52
3.4.3.2 Chosen Initial Value 52
3.4.4 Design Principles for TMTO 53
3.4.5 Applications on Stream Ciphers 54
3.5 Statistical Tests Based on Random Mappings 56
3.6 Other Relevant Applications 57
4 COMPUTATIONS OF SUCCESS RATES 59
4.1 Introduction 59
4.2 Hellman Tables 60

4.2.1 A Survey on Known Computations 60
4.2.2 Detailed Computations 61
4.2.3 Specialized Computations 62
4.2.4 Experimental Results 64
4.2.5 Hellman's Curve 66
4.3 DP Success 66
4.3.1 A Survey on Known Computations 66
4.3.2 Detailed Computations 68
4.3.3 Specialized Computations 70
4.3.4 Experimental Results 73
4.4 Rainbow Tables 74
4.4.1 A Survey on Known Computations 74
4.4.2 Detailed Computations 74
4.4.3 Specialized Computations 75
4.4.4 Experimental Results 75
4.5 Comparing TMTO techniques on A5 75
5 CONCLUSION AND FUTURE WORKS 77
REFERENCES 79
APPENDICES
A Description of Related Ciphers 84
A.1 A5/1 84
A.2 LILI 85
A.3 Toyocrypt 86
A.4 eStream Project 86
A.4.1 LEX 87
A.4.2 MICKEY 87
A.4.3 GRAIN 88
A.5 SHA 89
B Relation with Incomplete Euler Gamma Function 90

C Statistical Randomness Test 92

C.1 Pearson’s Chi Square test 92

VITA 93

LIST OF TABLES

TABLES

Table 2.1 A Random Function 10

Table 2.2 Empirical Results for $n = 2^8, 2^{10}$ 15

Table 2.3 Empirical Results for $n = 2^{12}, 2^{13}$ 15

Table 3.1 Mask Function, $h(x) = f(x) \oplus 1$ 27

Table 3.2 Comparison of Hellman Tables 30

Table 3.3 Effects of t_{min} and t_{max} for Reduced Sha-1 $N = 2^{28}$, d=6, $m = 2^{16}$ 36

Table 3.4 Threshold Results for $m = 2^{22}$ $t = 2^{11}$ 47

Table 4.1 Comparison Table for Asymptotic Approach, Observed by Kusuda, Kusuda et al Approach, Hellman's Approach 61

Table 4.2 Comparison for Reduced Sha-1 with $N = 2^{21}$ 65

Table 4.3 For $N = 2^{40}$ Hellman Construction 65

Table 4.4 Effects of r on Success Rates 65

Table 4.5 Hellman Construction for Different λ s for $N = 2^{40}$. 66

Table 4.6 DP Results on Reduced Sha-1 73

Table 4.7 Rainbow Table Coverage for $N = 2^{30}$ 75

Table 4.8 Comparing Three Methods on A5 76

Table A.1 A5/1 Specificitaion 85

LIST OF FIGURES

FIGURES

Figure 1.1 Cryptographic Primitives 2

Figure 2.1 Functional Graph 9

Figure 2.2 A Directed Graph of a Random Function 10

Figure 2.3 Functional Graph of a Random Permutation 15

Figure 3.1 Reduction Function 19

Figure 3.2 Hellman Table 20

Figure 3.3 Cycle 23

Figure 3.4 Merge 23

Figure 3.5 Graph Representation of h(x) 28

Figure 3.6 Variant Distinguished Point on Permutations 37

Figure 3.7 Classic Table vs. Rainbow Table 39

Figure 3.8 Perfect Hellman Construction 45

Figure 4.1 Feasibility of a Table 64

Figure A.1 A5-1 Stream Cipher 84

Figure A.2 Lili -128 Stream Cipher 85

Figure A.3 Lex Stream Cipher 87

Figure A.4 Mickey Stream Cipher 88

Figure A.5 Grain Stream Cipher 88

CHAPTER 1

INTRODUCTION

The expansion of worldwide communications and an explosive growth of the digital storage make information more vulnerable to misuse, and require adequate security. *Cryptography* is the art of science concerning to provide information security in the digital world. Indeed, the protection of new communication systems has been the emphasis of cryptography throughout much of its history.

Information security has three main aspects: authenticity, confidentiality and data integrity. *Authenticity* aims to ensure the identity of two parties entering into communication. *Confidentiality* means keeping the content of information from unauthorized person. *Data integrity* aims to verify that the information have not been modified by unauthorized person.

When a sender (usually called Alice) wants to send a message to a receipt (Bob), she applies a mathematical transformation, $E()$, to the *plaintext*, P. This process of converting ordinary information (plaintext) into some unreadable form *ciphertext*, C, is called *encryption*. And it is shown as $C = E(P)$. Bob will decrypt the ciphertext by applying the inverse transformation which is called *decryption*.

The *cryptographic primitives* (tools) may be divided into two categories: keyed and unkeyed primitives as shown in Figure 1.1. Hash functions, secret sharing schemes, and compression functions may be listed as examples of unkeyed primitives. Keyed cryptosystems may be classified into two sub categories: Public Key cryptography and Symmetric Key (Secret Key /Private Key) cryptography.

In *Public Key* Cryptosystems each party has a pair of key which are called public, e, and private keys, d. The encryption key e does not need to be kept secret. The main idea which lies

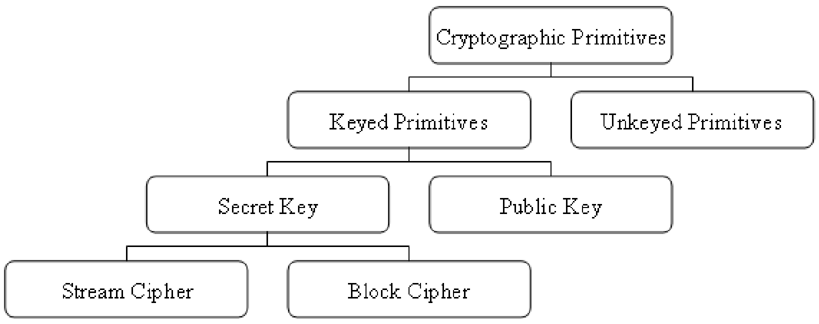

Figure 1.1: Cryptographic Primitives

behind public key cryptography is that given *e*, it is infeasible to determine the corresponding decryption key *d*. Public keys are used for digital signatures, key exchange protocols, etc.

In *Symmetric Key* Cryptosystems the two parties communicating over distrusted channel share a private key, *k*. There are two types of symmetric key cryptosystems: stream ciphers and block ciphers. Block ciphers are encryption algorithms that transform *n* bit blocks into *n* bit blocks. On the other hand, stream ciphers may be described as keyed generators of pseudo random sequences over a finite alphabet. In this study, we focus on analysis of symmetric key cryptosystems. The following section describes basic concepts related to this study.

1.1 Symmetric Key Cryptography

1.1.1 Block Ciphers

A *block cipher* is an algorithm to encrypt n-bit plaintext blocks to m-bit ciphertext blocks; in most of ciphers, $m = n$ and called the block length. It is parameterized by a random k-bit key $K \in \mathcal{K}$. For a unique decryption, once key K is fixed, the encryption function must be a bijection. The most usual block lengths for existing block ciphers are $n = 64$ and 128 bits. Modes of operations are used (See [27]) for plaintext messages exceeding one block in length. Block ciphers have been used as standards, such as DES (Data Encryption Standard, AES (Advanced Encryption Standard). Several famous block ciphers are also used in cryptography, such as IDEA, RC5, MISTY1 and KASUMI.

1.1.2 Stream Ciphers

A *stream cipher* is an algorithm to encrypt individual plaintext alphabet, usually the binary alphabet {0,1}, one at a time. Classical stream ciphers mostly produce one output bit on each clock. However, word-oriented stream ciphers encrypt the plaintext as bytes or larger units. A stream cipher may be thought as a cryptographically secure pseudo random number generator. Stream ciphers are generally faster than block ciphers and they have less hardware complexity. The main idea which lies behind the stream ciphers based on the One Time Pad (OTP) method.

One Time Pad / Vernam Cipher over the binary alphabet is defined by $c_i = m_i \oplus k_i$ for $i = 1, 2, 3, \ldots$, where $m_1, m_2, m_3, \ldots$ are the plaintext bits, $k_1, k_2, k_3, \ldots$ are the key bits, $c_1, c_2, c_3, \ldots$ are the ciphertext bits, and $\oplus$ is the XOR function (bitwise addition modulo 2). Shannon proved that if the key bits are generated independently and randomly, the Vernam cipher is unconditionally secure against a ciphertext-only attack (See 1.1.3). It should be noted that the keystream should be as long as the ciphertext. This is the drawback of OTP, since it is difficult to distribute and manage such a key. The idea behind the design of stream ciphers is to generate keystream bits pseudo randomly from a smaller secret key.

1.1.3 Security Analysis of Symmetric Ciphers

A cryptosystem should be secure even if everything about the system, except the key, is known by the attacker. This was firstly stated by Auguste Kerckhoffs in the 19th century.

Kerckhoffs' Principle : The cryptosystem should be secure even if the attacker knows all details of the encryption algorithm, except the secret key. This section presents various cryptanalytic attack scenarios.

Cryptanalytic Attack Scenarios

- Ciphertext Only Attack:
  The attacker has only access to ciphertext.
- Known Plaintext Attack:
  The attacker has access to both plaintext and ciphertext pairs.

- Chosen Plaintext Attack:

  The attacker has ability to choose the plaintexts and to obtain the corresponding plaintexts.

- Chosen Ciphertext Attack:

  The attacker has ability to choose the ciphertext and to obtain the corresponding ciphertext under an unknown key.

- Adaptive Chosen Plaintext/Ciphertext Attack:

  The attacker has access to a number of ciphertexts/plaintexts to be decrypted/encrypted, and then uses the results of these decryptions/encryptions to select subsequent ciphertexts.

The efficiency of attack depends on the amount of ciphertexts/plaintexts required.

1.2 One way Functions

One way functions are functions that are easy to compute (in polynomial time) but it is hard to invert. By inversion of a one way function we mean that no polynomial-time algorithm can compute a preimage of a given point, if there exists any.

1.2.1 Some Generic Attacks on One-way Functions

Generic Attacks consider the encryption function as a black box, and they are not interested in the construction details of encryption function. Exhaustive search, lookup table and time memory trade off are examples of generic attacks.

Exhaustive Search

Exhaustive search is a method to find a pre-image of a one way function. It is the simplest attack against a cryptosystem in which all possible keys are tested in order to find a proper key. An encryption scheme may be broken by decrypting a fixed ciphertext with trying all possible keys and discarding keys that does not yield the known plaintext.

For a stream cipher, if attacker has a keystream generated by an unknown key, he simply tries all possible keys (or internal states), and compares the result with the known keystream. If a stream cipher has n bit internal state, the exhaustive search will take less computational steps than 2^n. Therefore, for a valid attack, the attack complexity must be less than 2^n.

Exhaustive searching consumes an excessive time. When attack has to be carried multiple times, it is not an efficient algorithm.

Lookup Tables

Lookup Table replaces a runtime computation with a simple table construction. A table is constructed containing all possible outcomes (ciphertexts) and corresponding incomes (keys). This method may be another solution to invert a one-way function. But in this situation, it requires extreme amounts of memory for a function acting on large sets.

Time Memory Trade Off

Time Memory Trade Off (TMTO) is a method combining Exhaustive Search and Lookup Table methods. The aim of the method is to mount an attack which has a lower memory complexity than lookup table and a lower online time complexity than exhaustive searching. Basically, the attack may be summarized as following:

Let a oneway function be $f : X \to X$.

1. The attacker prepares a table of size M. He is allowed to prepare tables with in time, P.
2. Given a target point $c_0 \in X$, utilizing his table, the attacker searches $x \in X$ satisfying $f(x) = c_0$ within time T among the values in his table.

There are mainly two phases in TMTO; offline (precomputation) and online. In Step 1 (called as Precomputation/Offline Phase), a table is constructed as in Lookup Table Method, but all table is not stored, only a partial data is stored. In Step 2 (Online Phase), given a point, the preimage is searched over this precalculated table. The precomputation time, P, is still on the order of exhaustive searching, the trade off is between M and T where M represents memory complexity and T represents online time complexity.

1.3 Outline and Main Contributions of This Thesis

Inverting a oneway function has an important role in the security of most encryption schemes. For instance, under a fixed key, a block cipher is a oneway function which maps a (secret) key to the ciphertext.If an efficient way can be found to invert this map, this will imply totally breaking down the encryption system. TMTO is a probabilistic method to invert a oneway functions. In most cases, it is a theoretical attack since the precomputation time exceeds the exhaustive searching. However, it is an important countermeasure for designers when security of symmetric ciphers are taken into consideration.

In this thesis, we study the design and analysis of TMTO attacks. We also study in detail the statistical behaviors of random mappings (also permutations) which affects the selection of parameters for trade off attacks. In Chapter 2, we present the summary of the random mapping statistics from the view point of TMTO. We also recalculate some expected values with a simpler manner than the existing proofs. In addition, we present a comprehensive survey of the TMTO attacks on symmetric ciphers in Chapter 3. In the literature, TMTO also occupies a place in both hash functions (as herding attacks) and asymmetric ciphers, but these are out of the scope of thesis. Some variant constructions of TMTO are also presented in Chapter 3. We propose a variant method which is based on distinguished point method. Our construction has a high success rate when one way function is a random permutation. We provide a technique to prepare a Perfect Hellman table. Besides, we propose to use a threshold which prevents to eliminate waste of nonmerging chains by a load of memory. In Section 3.5, three new distinguishers are given which not only distinguish a keytsream of a cipher from a truly random sequence but also consider the feasibility of TMTO method. There is no precise guideline in the literature that points out how to choose parameters for Hellman and two main improvements; Distinguished Point(DP) Method and Rainbow Method. In the subsequent chapter, we present a detailed analysis of the success rate of Hellman table via new parameters and also show how to choose parameters to achieve a higher success rate. The results are also experimentally confirmed. Hellman's TMTO Curve is discussed. And we discuss the average length of a chain when we use distinguished points as endpoints. We calculate the average length of a chain, and also the number of left chains after elimination which naturally allows us to derive a formula regarding the overall success rate of the method. We study on the coverage of a Rainbow table and show that no matter how the parameters are chosen, coverage remains the same. As the success probability of a trade off algorithms play an important role in any comparison between trade offs algorithms or in their practical use, we give more simple and accurate formulas for three trade off algorithms which yields choosing robust optimal parameters. Then, we compare the three methods applying A5 stream cipher. Concluding remarks are given in Chapter 5.

CHAPTER 2

RANDOM MAPPINGS

2.1 Introduction

Random mappings are functions from a finite set of n elements onto m elements. This chapter introduces the random mapping statistics from the view point of TMTO method. They are widely used in combinatorial problems. Various characteristics of random mappings have been studied in [2, 44, 33, 45]. TMTO deals with functions from a finite set of n elements onto n elements. Let $\mathcal{X}$ denote the finite domain of size n and $\mathcal{F}_n$ denote the collection of all functions (f) from domain $\mathcal{X}$ into range $\mathcal{X}$. We will consider only random mapping model where every function from $\mathcal{F}_n$ is chosen equally likely. Statistical behaviors of both random mappings and random permutations are summarized. They have a great importance when choosing the parameters of TMTO method. In this chapter, the expected values for ρ length and cycle length and resulting tail length are calculated with a different approach. This approach yields more simpler and intelligible calculations than the presented calculations. We also give the number of points with exactly i-preimages. This chapter is organized as follows. First, we describe the terms in graph representation of functions with an example of size $n = 2^6$. Then, the probability distributions and expected values of some random mapping properties are given and the statistics of random mappings in the literature are summarized in Section 2.3. Finally, the statical behaviors of random permutations are given in Section 2.4.

2.2 Graph Representation of Functions

Let f be a function, $f : \mathcal{X} \to \mathcal{X}$, where $\mathcal{X}$ denote the finite domain of size n and let $\mathcal{F}_n$ denote the collection of all functions where every function is equally likely to be chosen. So the sample space consists of n^n random mappings, in other words, the probability that a particular function (f) from $\mathcal{F}_n$ is chosen is $\frac{1}{n^n}$.

Starting from a point $x_0 \in \mathcal{X}$ and iteratively applying f, the following sequence is obtained;

$$\{x_0, f(x_0), f^2(x_0), \ldots\}. \tag{2.1}$$

The k^{th} iteration of f on $\mathcal{X}$, where $0 \leq k \leq n$ will be $f^k(x_0) = f(f^{k-1}(x_0))$ where $f^0(x_0) = x_0$. For some $k \geq 0$ if $f^k(x_0) = y$ then we call y, a k^{th} image of x_0 in f. For $k < 0$, $f^k(x_0)$ may not exist, which we will call *terminal nodes*, (in other words, a node may have no inverse image) or may not be uniquely determined (more than one inverse which cause a *false alarm* from the view point of TMTO). A random mapping, f can be represented by a functional graph.

Definition 2.2.1 *A functional graph of a function $f : \mathcal{X} \to \mathcal{X}$ is a directed graph whose nodes are the elements of $\mathcal{X}$ and whose edges are the ordered pairs $(x, f(x))$, for all $x \in \mathcal{X}$.*

In Figure 2.1, the typical behavior of an iteration operation is given. Since the set $\mathcal{X}$ is finite, after some iterations, we will encounter a point that has occurred before. Let $f^m(x_0)$, $0 \leq m \leq n-1$ be the point that the iteration enters a loop. Then, $f^m(x_0) = f^k(f^m(x_0))$, k is the smallest positive integer which we call the *cycle length.* The path between x_0 and $f^m(x_0)$ is called the *tail length.* The sum of the tail length and cycle length is defined as the *ρ-length.*

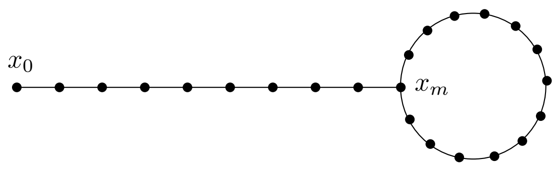

Figure 2.1: Functional Graph

Example 2.2.2 Consider a random function for $n = 2^6$ is prepared as in Table 2.1.

Table 2.1: A Random Function

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$f(x)$	15	63	59	53	53	51	5	12	32	43	60	13	48	16	61	54
x	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
$f(x)$	4	57	11	35	28	62	36	62	22	61	27	1	16	36	17	43
x	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
$f(x)$	15	59	10	61	22	2	60	9	19	59	19	41	57	14	45	38
x	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
$f(x)$	22	4	48	3	62	51	36	22	54	39	40	24	20	23	45	23

The functional graph will be as in Figure 2.2.

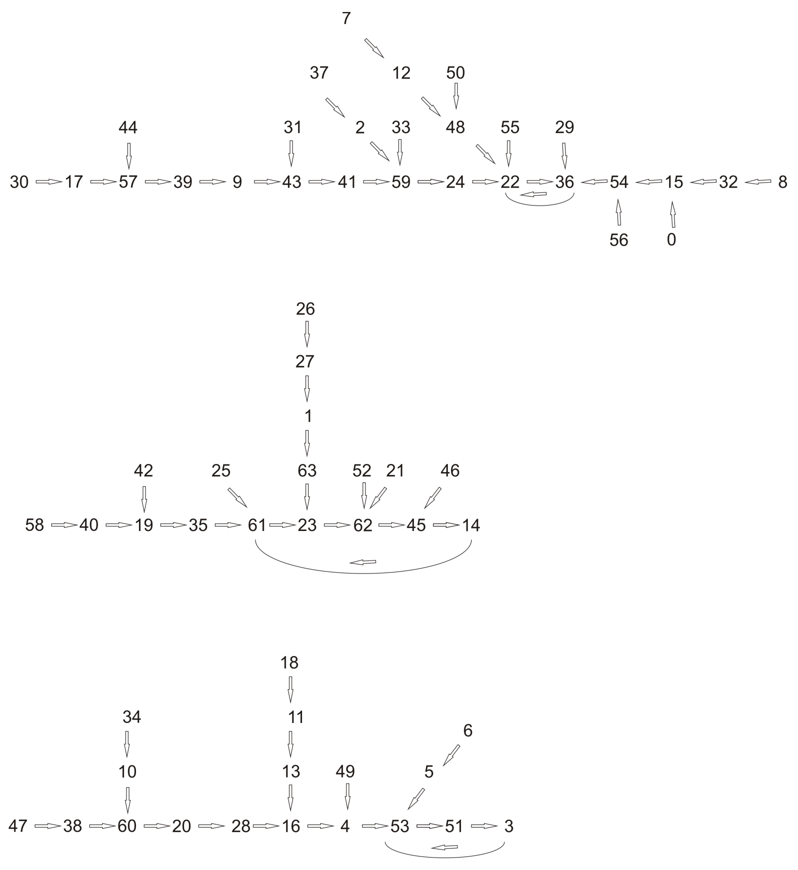

Figure 2.2: A Directed Graph of a Random Function

Random mapping properties of this function may be listed as follows:

Number of components = 3

Number of cyclic nodes = 10

Number of nodes which has k preimages:

k	0	1	2	3	4
number of nodes	24	23	12	4	1

For instance, for a point $x_0 = 43$, the length of the tail is 3, and the length of the cycle is 2 then the rho length is 5.

2.3 Random Mapping Statistics

The results of the statistical behaviors of random mappings (See [33, 30, 44, 45, 52, 2]) are summarized in the following theorems. Although the expected values of random mappings are widely studied in the literature, we calculate the expected values for ρ length and cycle length and resulting tail length with a simpler approach.

Theorem 2.3.1 *The expected ρ-length for a random mapping of n elements is given as*

$$E(\rho - length) = \sqrt{\frac{\pi n}{2}}.$$

Proof. The probability distribution of the ρ-length is:

$$Pr(\rho - length = k)\frac{(n-1)!}{(n-k)!}\frac{k}{n^k} \tag{2.2}$$

Since

$$\begin{aligned} Pr(\rho - length = 1) &= \frac{1}{n} \\ Pr(\rho - length = 2) &= \frac{n-1}{n}\frac{2}{n} \\ Pr(\rho - length = 3) &= \frac{n-1}{n}\frac{n-2}{n}\frac{3}{n} \\ &\dots \end{aligned}$$

Then the expected value of ρ-length is

$$\begin{aligned}E(\rho - length = k) &= \sum_{k=0}^{n} \frac{(n-1)!}{(n-k)!}\frac{k^2}{n^k} \text{ by writing } k \to n-k\\ &= \frac{(n-1)!}{n^n}\sum_{k=0}^{n}\frac{(n-k)^2}{k!}n^k\\ &= \frac{(n-1)!}{n^n}\left[n^2\sum_{k=0}^{n}\frac{n^k}{k!} - 2n\sum_{k=0}^{n}\frac{n^k k}{k!} + \sum_{k=0}^{n}\frac{k^2 n^k}{k!}\right]\end{aligned}$$

Let $Y_1 = \sum_{k=0}^{n+1}\frac{n^k}{k!}$, $Y_2 = \sum_{k=0}^{n+1}\frac{n^k k}{k!}$, $Y_3 = \sum_{k=0}^{n+1}\frac{n^k k^2}{k!}$ and define

$F(x) = \frac{e^{nx}}{2}$ thus $F(x) = \sum_{k=0}^{\infty}\frac{n^k x^k}{k!}$ then $Y_1 \approx F(1)$

$F'(x) = \frac{ne^{nx}}{2}$ thus $F'(x) = \sum_{k=0}^{\infty}\frac{kn^k x^{k-1}}{k!}$ then $Y_2 \approx F'(1)$

$F''(x) = \frac{n^2 e^{nx}}{2}$ thus $F''(x) = \sum_{k=0}^{\infty}\frac{k^2 n^k x^{k-2}}{k!} - \frac{kn^k}{k!}$ then $Y_3 \approx F''(1) + F'(1)$

$$\begin{aligned}E(k) &= \frac{n!}{n^n}\left[\left(n^2\right)e^n - 2n(ne^n) + n^2e^n + ne^n\right]\\ &\cong \frac{n!}{n^n}\frac{e^n}{2} = \sqrt{\frac{\pi n}{2}}.\end{aligned}$$

■

Theorem 2.3.2 *The expected cycle-length for a random mapping of n elements is given as*

$$E(cycle - length) = \sqrt{\frac{\pi n}{8}}.$$

Proof.

$$Pr(cycle - length = k) = \sum_{j=k}^{n}\frac{(n-1)!}{(n-j)!n^j} \tag{2.3}$$

$$
\begin{aligned}
E(cycle - length = k) &= \sum_{k=0}^{n}\sum_{j=k}^{n}\frac{(n-1)!}{(n-j)!n^j}k \\
&= \sum_{j=1}^{n}\frac{(n-1)!}{(n-j)!n^j} + 2\sum_{j=2}^{n}\frac{(n-1)!}{(n-j)!n^j} + \ldots + \frac{(n-1)!}{n^n} \\
&= \frac{1}{2}\sum_{j=0}^{n}\frac{(n-1)!(j+1)j}{(n-j)!n^j} \\
&= \frac{(n-1)!}{2}\sum_{j=0}^{n}\frac{(j+1)j}{(n-j)!n^j} \text{ by writing } j \to n-j \\
&= \frac{(n-1)!}{2n^n}\sum_{j=0}^{n}\frac{(n-j)(n-j+1)}{j!n^{-j}} \\
&= \frac{(n-1)!}{2n^n}\left[(n^2+n)\sum_{j=0}^{n}\frac{n^j}{j!} - (2n+1)\sum_{j=0}^{n}\frac{n^j j}{j!} + \sum_{j=0}^{n}\frac{j^2 n^j}{j!}\right] \\
&= \frac{(n)!}{2n^n}\frac{e^n}{2} = \sqrt{\frac{\pi n}{8}}.
\end{aligned}
$$

■

Since $E(\rho - length) = E(cycle - length) + E(tail - length)$, then $E(tail - length = k) = \sqrt{\frac{\Pi n}{8}}$.

Theorem 2.3.3 *The number of points with exactly i-preimages is*

$$P(i) \approx \frac{n}{e.i!}.$$

Proof. The probability that a point, x, has i preimages is

$$P(i) = \binom{n}{i}\left(\frac{1}{n}\right)^i\left(\frac{n-1}{n}\right)^{n-i}$$

The probability that n points have i preimages is

$$P(i) = n\binom{n}{i}\left(\frac{1}{n}\right)^i\left(\frac{n-1}{n}\right)^{n-i}$$

For $i = 0$,

$$P(0) = n\left(\frac{n-1}{n}\right)^n \approx \frac{n}{e}$$

For $i = 1$,

$$P(1) = n\frac{n}{n-1}\left(\frac{n}{n-1}\right)^n \approx \frac{n}{e}$$

$\vdots$

$$P(i) \approx \frac{n}{e.i!}.$$

■

The above result is also given by [54]. Number of terminal points (number of points which has no preimages) is $P(0) = \frac{n}{e}$. Therefore, number of images points are $n - P(0) = (1 - e^{-1})n$.

Theorem 2.3.4 *The expected values for a random mapping are given as[30]-[33]*

(i) *Number of components* $= \dfrac{ln(n)}{2}$

(i) *Number of terminal nodes* $= e^{-1}n \approx 0,3679n$

(ii) *Number of image nodes* $= (1 - e^{-1})n \approx 0,6321n$

(iii) *Average cycle length* $= \sqrt{\dfrac{\pi n}{8}} \approx 0,6267\sqrt{n}$

(iv) *Average tail length* $= \sqrt{\dfrac{\pi n}{8}} \approx 0,6267\sqrt{n}$

(v) *Averageplength* $= \sqrt{\dfrac{\pi n}{2}} \approx 1,2533\sqrt{n}$

(vi) *Average component size* $= \dfrac{2n}{3}$

(vii) *Maximum cycle length* $= 0.78248\sqrt{n}$

(viii) *Maximum tail length* $= 1.73746\sqrt{n}$

The last two items are given in preprint, "Mapping the discrete logarithm" , by Joshua Holden and Daniel R. Cloutier. Some empirical examples for $n = 2^8, 2^{10}, 2^{12}, 2^{13}$ are presented in Table 2.2 and 2.3.

Table 2.2: Empirical Results for $n = 2^8, 2^{10}$

	$n = 2^8$			$n = 2^{10}$		
	Calculated	Empirical		Calculated	Empirical	
Components	3	5	3	4	4	6
Cyclic Nodes	20	44	24	40	19	30
Terminal Nodes	95	93	91	379	373	369
Av. Tail Length	10	6.96	8.7	20	26.09	28.57
Av. Cycle Length	10	8.8	8	20	4.75	5
Av. Rho Length	20	15.8	15.8	40	30	33.6

Table 2.3: Empirical Results for $n = 2^{12}, 2^{13}$

	$n = 2^{12}$				$n = 2^{13}$		
	Calculated	Empirical			Calculated	Empirical	
Components	4	7	11	3	5	3	3
Cyclic Nodes	80	72	132	62	113	63	70
Terminal Nodes	1516	1532	1513	1516	3031	3021	2994
Av. Tail Length	40	34.29	18.45	35.36	57	73	55.7
Av. Cycle Length	40	10.28	12	20.66	57	21	10
Av. Rholength	80	44.7	20.45	55.36	113	94	65.7

2.4 Random Permutations

Permutations have the unary functional graphs that the sum of in-degrees must be the same as the sum of the out-degrees. Each node must have exactly one inverse. There are no terminal nodes and tail nodes as in Figure 2.3. This also shows that every node is in a cycle.

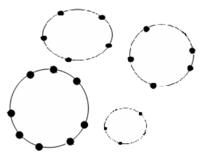

Figure 2.3: Functional Graph of a Random Permutation

Let $\alpha(n)$ be the number of cycles in a permutation. Then we have

$$Pr(\alpha(n) = j) = \frac{S_n^j}{n!}$$

where S_n^j, $1 \leq j \leq n$ is the Stirling number of first kind [45].

Let L_n be the expected length of the longest cycle in a random permutation then

$$lim_{n\to\infty}(L_n/n) = 0.62432965$$

Theorem 2.4.1 *The expected values for a random permutation are*

(i) *Number of components* $= \sum_{i=1}^{n} \frac{1}{i} = H_n$

(ii) *Number of terminal nodes* $= 0$

(iii) *Number of image nodes* $= n$

(iv) *Average cycle length* $= \frac{n+1}{2}$

(v) *Average tail length* $= 0$

(vi) *Average* ρ *length* $= \frac{n+1}{2}$

(vii) *Maximum cycle length* $= 0.62432965n$

It should be noted that the average length of the cycle is given in the above theorem is calculated by starting a random node, x_0. But we can define another average by selecting uniformly randomly from all cycles. Since #components $\approx \ln n$, the average cycle length of a mapping is

$$\text{Average cycle length} = \frac{n}{\#components} = \frac{n}{\ln n}.$$

CHAPTER 3

TIME MEMORY TRADE OFF

3.1 Introduction

This chapter first introduces the basic idea behind the Time Memory Trade Off (TMTO) method proposed by Hellman [34] with an example. In subsequent sections, the method is described in the contexts of block cipher and stream cipher, separately. There are mainly two basic improvements of method; Distinguished Points method (DP) and Rainbow method. In Section 3.3.1, we summarize the Distinguished Points and propose variant construction for this method. Starting by listing the distinguished points, the table is constructed. This method has a high coverage when it is applied on a random permutation. The number of distinguished points is equal to memory complexity. So if the attacker has access to a high memory than he can choose the number of distinguished points, D, greater. We also discuss the optimal parameters in order to have a high coverage. In Section 3.3.2, Rainbow method is summarized. In Section 3.3.3.1, we propose a technique which constructs a Perfect table which has nonmerging chains. In [3] Avoine et al stated that constructing a perfect Hellman table is not efficient since all chains have to be looked up. However it increases the precomputation complexity, with small values of t, it is possible to have a high success rates. In both rainbow and DP perfect tables, identical endpoints are discarded to reduce the merging chains. On the other hand, the merge may appear in the last columns in the chain, this will reduce coverage and waste time. As an example for a random mapping, the trade off between computation and identical points are given. Other extensions and refinements on block ciphers such as FPGA implementations and using multiple data are presented in this chapter. In Section 3.4, applications of trade off method on various stream ciphers such as A5, Lili-128 are examined and also the design principles of stream ciphers for resistance to TMTO attacks are listed. In Section 3.5 we propose three new distinguishers which may give valuable clues when applying TMTO method. The results in this chapter have been published in the papers [65]-[71].

3.2 Hellman's Construction

In 1980, Hellman[34] proposed Time Memory Trade Off (TMTO) method which suggests a trade off between time and memory by storing some pre-computed data in the memory. This method has a lower time complexity (in online phase) than exhaustive search and a lower memory complexity than lookup table. If the attacker has access to a large memory, the computation time will be less. Thus, it is a probabilistic method; the success rate depends on the time and memory allocated for cryptanalysis. This attack can be used as a known plaintext attack. Exhaustive search consumes a lot of computing power when the same attack has to be carried out multiple times. TMTO stores some data in memory, thus it can be carried out when the attacker can guess some bytes of data such as password hashes. Moreover, it can also be used as a ciphertext-only attack by waiting for repeated ciphertext blocks and assuming the corresponding chosen plaintext.

Hellman applied TMTO on the block cipher, Data Encryption Standard (DES). For any N key cryptosystem this method may recover key in $N^{2/3}$ operations with $N^{2/3}$ words of memory after a precomputation which requires N operations [34]. He also stated that this method can be applied to invert a one-way function. TMTO method can be summarized as follows:

Let $E : K \times P \to C$ be an encryption function where $C \in \{0,1\}^n$, $K \in \{0,1\}^k$ and $P \in \{0,1\}^n$ denote the ciphertext, the secret key and the plaintext, respectively. The encryption of one block is written as:

$$c = E(x, p) \text{ or } c = E_x(p)$$

Given $c_0 = E(x, p_0)$ where p_0 is a fixed plaintext and c_0 is the corresponding ciphertext, the main aim is to recover the key $x \in K$. We may assume that $k = n$. In some ciphers, the domain and range size of E may not be the same length. In this situation, employing a reduction or an expansion function R, it is possible to define $E_{p_0} \circ R : K \to C$ such that $(E_{p_0} \circ R) = R(E(x_0, p))$. Therefore, to generate a key from a ciphertext, we use $x_{i+1} = R(c_i)$ where $c_i = E(x_i, p_0)$ as in Figure 3.1. For instance, DES operates on 64 bit plaintext to produce a 64-bit ciphertext under

56 bit key. In his paper, Hellman suggested to drop the last 8 bits to reduce 64 bits ciphertext to 56 bits.

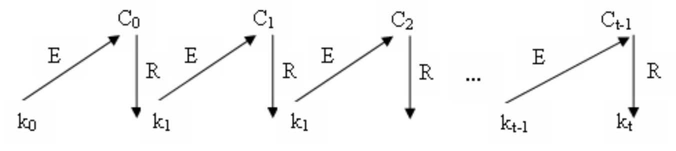

Figure 3.1: Reduction Function

A TMTO application is composed of two phases; an offline (precomputation) phase and an online phase. In the offline phase, the attacker constructs tables that contain possible keys; this process is approximately equivalent to the exhaustive search but only a part of the tables are stored. During online phase, the attacker expects to recover the particular key when the plaintext and ciphertext are known; his aim is to reduce the time of search.

Offline Phase (Precomputation):

In the Hellman table, starting from a random x_0, iteratively evaluating $E \circ R = f$, a chain of length t is generated as follows:

$$x_0 \rightarrow x_1 \rightarrow x_2 \rightarrow \ldots \rightarrow x_t.$$

To construct a Hellman table of size $m \times t$, choosing m random start points, as shown in Figure 3.2 chains of length t are generated.

Only the first and last terms of each chain are stored to save memory. The first element of a chain is called *Starting Point* (SP), and the last element is called *Ending Point* (EP). By knowing the starting point, the successive elements can be recalculated in the chain. Also the table is sorted with respect to the end points for speeding up the search in online phase. Marginal profit of each new row is quite small when the number of rows exceeds some bound (given in the Section 4.2.3). Therefore, rather than increasing the number of rows, it is much more reasonable to define new tables each of which depends on new function formed by combining the original function with some permutation (See Mask Function 3.2). By this way, r-Hellman tables are formed. It should be noted that the chains computed using different functions can intersect without merging.

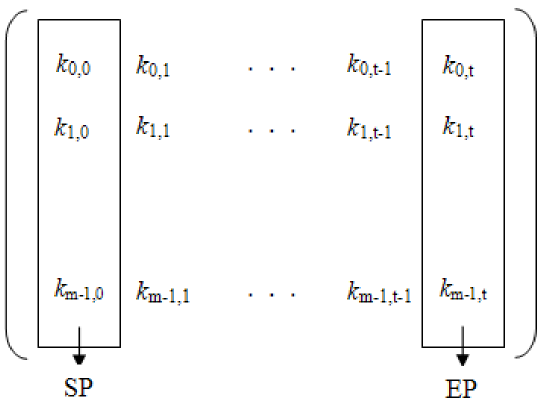
$k_{0,0}$ $k_{0,1}$ . . . $k_{0,t-1}$ $k_{0,t}$
$k_{1,0}$ $k_{1,1}$ . . . $k_{1,t-1}$ $k_{1,t}$
$k_{m-1,0}$ $k_{m-1,1}$ . . . $k_{m-1,t-1}$ $k_{m-1,t}$
SP
EP

Figure 3.2: Hellman Table

Algorithm 3.2.1: PRECOMPUTATION(*void*)

```
procedure HellmanTable($f$)
  for $j \leftarrow 0$ to $m - 1$
    do
      select random $SP_j$;
      $tmp \leftarrow f(SP_j)$;
      for $s \leftarrow 0$ to $t - 1$
        do
          $tmp \leftarrow f(tmp)$;
      $EP_j \leftarrow tmp$;
    store $(SP_j, EP_j)$ pairs in table $i$, sorted with respect to end points;


main
  for $i \leftarrow 1$ to $r$
    do
      choose random $R_i \ni E \circ R_i = f_i$ ;
      HellmanTable($f_i$)
```

Online Phase:

For a preimage of a given ciphertext c_0, our aim is trying to find out if the key is used to generate c_0 is among the one used in any of the generated tables. A chain for c_0 is generated by iteratively applying f over c_0 and after each encryption, the obtained value is compared to the endpoints of all tables. If a match is found, then from the corresponding starting point, the whole chain is regenerated and the found key is expected to be the searched key. It should be noted that sometimes desired key is a part of a chain that is merged with another chain of the table, resulting false alarms (details can be found in Section 3.2.1). Therefore, the success rate of the attack is closely related to number of distinct keys that are covered in the offline phase.

The search in the i^{th} table is realized as follows. Firstly, it is checked whether c_0 is equal to any points of the last column (any endpoints). If $Y_0 = R_i(c_0)$ is equal to any end point EP_j for some j, $1 \leq j \leq m$ then the $k_{j,t-1}$ is the desired key. Starting from SP_j, after $t-1$ iterations $k_{j,t-1}$ can be calculated. If $R_i(c_0)$ is not equal to any endpoints, we repeatedly apply f_i at most t times and check whether the output is equal to any endpoints. If a match is found (at s_{th} iteration), it may be a desired key or EP_j has more than one inverse. Starting from SP_j till $k_{j,s-1}$ we check whether $R_i(c_0) = E(k_{j,s-1}, p_0)$ or not.

Algorithm 3.2.2: ONLINE(c_0)

```
for $i \leftarrow 1$ to $r$
  do
    if any of the $EP_j$ is equal to $R_i(c_0)$
      then
        $tmp \leftarrow Sp_j$;
        for $s \leftarrow 1$ to $t - 1$
          do
            $tmp \leftarrow f_i(tmp)$;
        return ($tmp$)
      else
        for $s \leftarrow 1$ to $t - 1$
          do
            if any of the $EP_j$ is equal to $f_i(c_0)$
              then
                $tmp \leftarrow Sp_j$;
                for $k \leftarrow 1$ to $s$
                  do
                    $tmp \leftarrow f_i(tmp)$; if $E(tmp, p_0) == c_0$
                  then return ($tmp$)
return ()$keyisnotfound$
```

Mask Function

Hellman suggested to use different tables with different mask functions for decreasing the repetitions of tables, since, different mask functions will have different cycle structures. The definition of mask function f_i, is $f_i(x) = \Phi_i(f(x))$ where Φ_i s are independent random permutations. f_i 's are obtained from f by a minor modification such as permuting the output bits of f as in Hellman's original work on DES. It should be noted that by permuting output bits, the cycle structure of the random mapping will not change. In the literature there are some suggestions for the mask functions:

- Hellman suggested to permute bits, $f_i(x) = f_i(x \oplus i)$, but this construction does not change the properties of functional graph of $f(x)$ (only the name of nodes change).
- $\Phi_i(x) = x \oplus i$ is given in [69],[61]

- An interesting idea by Mukhopadhyay et al. is using LFSR sequences instead of counters [58]. The authors proposed to generate a sequence s_i of n bit vectors, by randomly choosing a maximal length LFSR, then $\Phi_i(x) = x \oplus s_i$.

3.2.1 False Alarm

In online phase finding a matching endpoint does not necessarily imply that the key is in that chain. Since f_i is not injective, an EP may have more than one inverse such that the key may be part of another chain which has the same endpoint. This is called as a *false alarm*. It is caused by merges. There are two types of merges:

- There may be a cycle. Let $i \neq j$ and $x_i = x_j$ as in Figure 3.3

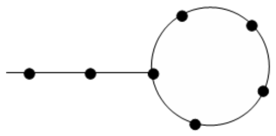

Figure 3.3: Cycle

- Two chains can merge as in Figure 3.4.

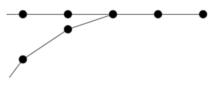

Figure 3.4: Merge

In such situations, after a collision occurs at least two chains will merge and different starting points will end with a same endpoint. If a collision occurs in the same Hellman Table, the corresponding starting point will not generate a chain that reaches the desired key. This is called a false alarm. False alarms increase the online time complexity of cryptanalysis, but they are omitted when calculating the time complexity in general usage. Since different tables use different reduction functions, collisions do not necessarily lead to a merge. Merging chains reduce the coverage, consequently reduce the rate of success.

Hellman proved that the expected number of false alarms per table is

$$E(falsealarm) \leq \frac{mt(t+1)}{2N}$$

Hellman also claimed that if $mt^2 = N$ and >> 1, then the expected increase of time complexity due to false alarms will be at most 50 percent, since false alarms occurs at most t operations of f and there are t tables, the expected time complexity is bounded by $\approx mt^4/(2N) = (NT)/(2N) = T/2$ [8].

3.2.2 Parameters

When applying the method by means of Hellman tables the parameters in question are the length of each chain(t), the number of chains (m), number of tables (r).

Memory complexity is the amount of memory that is necessary to store pre-computed data.

$M = 2 * m * r * m_0$ where M is the amount of memory; m_0 is the amount of memory that is necessary to store a start and end point (in general, it is omitted).

Time complexity is the amount of time required to perform the attack successfully. It is often compared to exhaustive key search. In time-memory trade offs, time complexity is divided into two parts; precomputation and online time complexities.

$T = t * r * t_0$ where T is the worst case time required in online phase; t_0 is the amount of time that is necessary to evaluate the function (in general, it is omitted). False alarms are not considered.

$P = m * t * r$ where P is the time complexity required in precomputation phase. Generally, P is not included in the attack complexity.

Data complexity (D) is the amount of data that is required to mount an attack (like ciphertexts, known plaintexts, chosen plaintexts, ...). For block ciphers, it is accepted to be 1, but in multiple data (See Section 3.3.4), if some plaintexts are encrypted with the same key, it is in question to use data which is called TMDT (Time Memory Data Trade Off).

Complexity of attack is generally called for the maximum of above complexities. It is mentioned in [34] as ”The N operations required to compute the table are not counted because they constitute a pre-computation which can be performed at the cryptanalyst's leisure."

Effects of parameters

- *t*: If chosen to be too large chains will probably enter in loops which will undesirably increase both precomputation time and online time. If it is too small, it will increase the memory requirement.
- *m*: If chosen to be too large, the number of identical points will increase. If it is too small, it is necessary to construct more tables.
- *r*: If it is too large, it will increase both memory and online time complexity.

Coverage

Coverage is the total number of distinct keys covered by the tables. Each table can contain information at most mt distinct points. Coverage of a function $f : \{0, 1, ...N-1\} \rightarrow \{0, 1, ..., N-1\}$ is the number of distinct points and is given in [34] as in the following:

$$Y(m,t) = \sum_{i=1}^{m} \sum_{j=0}^{t-1} [(N - it)/N]^{j+1}$$

Let us give some examples for $N = 2^6$, and construct Hellman tables, and discuss how to choose its parameters. The random function used in the following sections is given Example 2.2.2 for $N = 2^6$.

Example 3.2.1 Let us take $m = 2^3$, $t = 2^3$, $r = 1$

$3 \rightarrow 53 \rightarrow 51 \rightarrow 3 \rightarrow 53 \rightarrow 51 \rightarrow 3 \rightarrow 53$

$50 \rightarrow 48 \rightarrow 22 \rightarrow 36 \rightarrow 22 \rightarrow 36 \rightarrow 22 \rightarrow 36$

$60 \rightarrow 20 \rightarrow 28 \rightarrow 16 \rightarrow 4 \rightarrow 53 \rightarrow 51 \rightarrow 3$

$59 \rightarrow 24 \rightarrow 22 \rightarrow 36 \rightarrow 22 \rightarrow 36 \rightarrow 22 \rightarrow 36$

$49 \rightarrow 4 \rightarrow 53 \rightarrow 51 \rightarrow 3 \rightarrow 53 \rightarrow 51 \rightarrow 3$

$40 \rightarrow 19 \rightarrow 35 \rightarrow 61 \rightarrow 23 \rightarrow 62 \rightarrow 45 \rightarrow 14$

$12 \rightarrow 48 \rightarrow 22 \rightarrow 36 \rightarrow 22 \rightarrow 36 \rightarrow 22 \rightarrow 36$

$9 \rightarrow 43 \rightarrow 41 \rightarrow 59 \rightarrow 24 \rightarrow 22 \rightarrow 36 \rightarrow 22$

There are 21 distinct points in the above table (the first column is omitted since it will not give any clue for the key). Thus the coverage of table is 21. In general, the success rate of a table is given as the ratio of coverage to whole space. However, we are given a ciphertext in time memory trade offs, we mean that we are sure that the given point has at least an inverse image. Since approximately $N(1-e^{-1})$ (See Theorem 2.3.4) points have inverse images, in our example there are 24 terminal nodes which is also $2^{64}(1-e^{-1}) \approx 40 = 64 - 24$. Thus, the success rate of the above table is 21/40.

Assume in online phase we are given c_0. Since we have only the the tuple $\{SP, EP\}$, we are searching if c_0 is in the last column.

- Let $c_0 = 4$ is given, it is not in the last column. Then $f(c_0) = 53$ is in the first row, $SP = 3$, since $f^6(3) \neq 4$, it is a false alarm. $f^2(c_0) = 51$ is not in table. $f^3(c_0) = 3$ is in table, $SP = 60$, since $f^4(SP) = c_0$ then the key is $f^3(SP) = 16$.
- Let $c_0 = 62$ is given, it is not in the last column. Then $f(c_0) = 45$ is not in table. $f^2(c_0) = 14$ is in table, $SP = 40$, since $f^5(SP) = c_0$ then the key is $f^4(SP) = 23$.
- Let $c_0 = 27$ is given, it is not in the last column. Then $f(c_0) = 1$ is not in table. $f^2(c_0) = 63$ is not in table. $f^3(c_0) = 23$ is not in table. $f^4(c_0) = 62$ is not in table. $f^5(c_0) = 45$ is not in table. $f^6(c_0) = 14$ is in table, $SP = 40$, since $f(SP) \neq c_0$ then it is a false alarm.
- Let $c_0 = 53$ is given, since it is an EP. It is trivial that the key is $f^7(SP) = 3$. However it may not be desired key since both $f(4) = 53$ and $f(3) = 53$. But clearly, this is not expected as a false alarm.

Note: $f^s(c_0)$ means the s^{th} iteration of f that is $f^s(c_0) = f(f^{s-1}(c_0))$ where $f^0(c_0) = c_0$.

Trying all ciphertexts, 21 ciphertexts are found, that is equal to our expectations.

For $m = 2^2$, $t = 2^4$, $r = 1$, the success rate is 13/40. For $m = 2^4$, $t = 2^2$, $r = 1$, the success rate is 25/40. These suggest that choosing larger t causes tables to be consist of loops, choosing large m is better but it takes a lot of memory as in lookup tables.

Since adding many rows increases the merges, Hellman suggests to use r tables. Let us look at the example with two tables, the mask function of second table is $h(x) = f(x) \oplus 1$ as in Table 3.1.

The functional graph of $h(x)$ will be as in Figure 3.5.
Number of terminal nodes of $h(x) = 24$

Example 3.2.2 Let us take $m = 2^3$, $t = 2^2$, $r = 2$

Table 3.1: Mask Function, $h(x) = f(x) \oplus 1$

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$h(x)$	14	62	58	52	52	50	4	13	33	42	61	12	49	17	60	55
x	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
$h(x)$	5	56	10	34	29	63	37	63	23	60	26	0	17	37	16	42
x	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
$h(x)$	14	58	11	60	23	3	61	8	18	58	18	40	56	15	44	39
x	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
$h(x)$	23	5	49	2	63	50	37	23	55	38	41	25	21	22	44	22

Table1	***Table2***
$6 \to 5 \to 51 \to 3$	$10 \to 60 \to 22 \to 37$
$56 \to 54 \to 36 \to 22$	$42 \to 18 \to 10 \to 61$
$17 \to 57 \to 39 \to 9$	$34 \to 11 \to 12 \to 49$
$22 \to 36 \to 22 \to 36$	$56 \to 55 \to 23 \to 63$
$47 \to 38 \to 60 \to 20$	$30 \to 16 \to 5 \to 50$
$49 \to 4 \to 53 \to 51$	$60 \to 21 \to 63 \to 22$
$62 \to 45 \to 14 \to 61$	$29 \to 37 \to 3 \to 52$
$46 \to 45 \to 14 \to 61$	$3 \to 52 \to 63 \to 22$

The coverages of above tables are both 17, but the joint coverage is expected to be less than the sum of these values that is 26 (since the intersection of two table is 8).

In the online phase, tables can be looked up in sequentially or column by column. Online phase will be as in the following:

- Let $c_0 = 4$ is given, $c_1 = 4 \oplus 1$ is not in the last column. Then $h(c_1) = 50$ is in the table, $SP = 30$, since $f^2(SP) = c_1$, then the key is $f(SP) = 16$.
- Let $c_0 = 62$ is given, $c_1 = 62 \oplus 1$ is in the last column. Then $SP = 56$, since $h^3(SP) = c_1$ then the key is $h^2(SP) = 23$.
- Let $c_0 = 27$ is given, $c_1 = 27 \oplus 1$ is not in the last column. Then $h(c_1) = 26$ is not in table. Since $h(c_1) = c_1$,it is trivial that key will not be found.
- Let $c_0 = 22$ is given, $c_1 = 22 \oplus 1$ is not in the last column. Then $h(c_1) = 63$ is in the table, $SP = 56$, since $f^2(SP) = c_1$, then the key is $f(SP) = 55$. Moreover c_0 is in

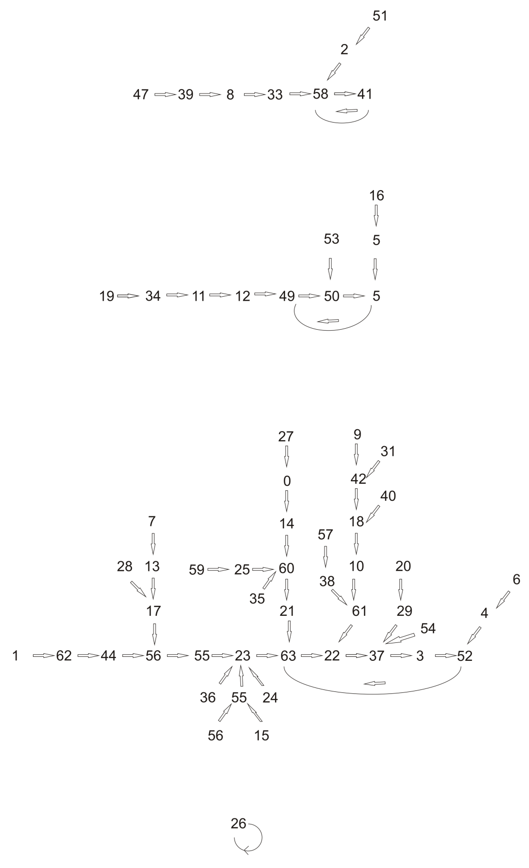

Figure 3.5: Graph Representation of h(x)

Table1. $SP = 56$, then the key is $f^3(SP) = 36$. Two keys are found, since 22 has more than one inverse.

For whole space, in the online phase, 17 ciphertexts are found from table 1, and 17 ciphertexts are found from table 2. The intersection is 8. Thus, the success rate of this example is 26/40, that is approximately equal to $1 - (1 - \frac{17}{40})^2$ (See Eq. 4.2).

Example 3.2.3 Let us take $m = 2^2$, $t = 2^3$, $r = 2$

Table1	***Table2***
$50 \to 48 \to 22 \to 36 \to 22 \to 36 \to 22 \to 36$	$10 \to 61 \to 22 \to 37 \to 3 \to 52 \to 63 \to 22$
$60 \to 20 \to 28 \to 16 \to 4 \to 53 \to 51 \to 3$	$42 \to 18 \to 10 \to 61 \to 22 \to 37 \to 3 \to 52$
$59 \to 24 \to 22 \to 36 \to 22 \to 36 \to 22 \to 36$	$34 \to 11 \to 12 \to 49 \to 5 \to 50 \to 49 \to 5$
$49 \to 4 \to 53 \to 51 \to 3 \to 53 \to 51 \to 3$	$56 \to 55 \to 23 \to 63 \to 22 \to 37 \to 3 \to 52$

For whole space, in the online phase 11 ciphertexts are found from table 1, and 15 ciphertexts are found from table 2. The intersection is 6. Thus the success rate of this example is 20/40. This shows us that taking t larger than the expected tail length will yield tables that consists of loops, and the success rate will decrease.

Example 3.2.4 Let us look at the example with four tables, the mask function for the tables are $h_1(x) = f(x)$, $h_2(x) = f(x) \oplus 1$, $h_3(x) = f(x) \oplus 2$, $h_4(x) = f(x) \oplus 3$, respectively. $m = 2^2$, $t = 2^2$,$r = 2^2$

Table1

$42 \to 19 \to 35 \to 61$

$41 \to 59 \to 24 \to 22$

$58 \to 40 \to 19 \to 35$

$10 \to 60 \to 20 \to 28$

Table2

$49 \to 5 \to 50 \to 49$

$45 \to 15 \to 55 \to 23$

$14 \to 60 \to 21 \to 63$

$29 \to 37 \to 3 \to 52$

Table3

$48 \to 20 \to 30 \to 19$

$21 \to 60 \to 22 \to 38$

$11 \to 15 \to 52 \to 60$

$39 \to 11 \to 15 \to 52$

Table4

$41 \to 56 \to 53 \to 48$

$60 \to 23 \to 61 \to 20$

$28 \to 19 \to 32 \to 12$

$38 \to 63 \to 20 \to 31$

For whole space, in the online phase, 10, 12, 9, 11 ciphertexts are found from tables, respectively. The intersection is 10. Thus, the success rate of this example is 25/40.

To summarize this section, Table 3.2 gives the comparison of success rates of Hellman tables for different parameters.

Table 3.2: Comparison of Hellman Tables

r	m	t	Success
1	2^3	2^3	0.525
	2^2	2^4	0.325
	2^4	2^2	0.625
2	2^3	2^2	0.65
	2^2	2^3	0.5
2^2	2^2	2^2	0.625

3.2.3 Hellman's Curve

The number of tables constructed in Hellman attack is $r = t$, then $M = m * t$, $T = t * t$. The time required in offline phase is $P = mt^2 = N$. Then, for *a random function*, trade off curve is obtained as :

$$TM^2 = N^2 \text{ where } 1 \leq T \leq N$$

Fiat and Naor [29] give a rigorous TMTO curve for inverting *any one way function* as

$$TM^3 = N^3$$

This is weaker than Hellman's Curve, but it can be used to invert any function rather than a truly random function.

3.3 TMTO on Block Ciphers

There are two main improvements of TMTO method; Distinguished Method and Rainbow Method. According to [8], these are the best algorithms when the structure of f is not to be used.

3.3.1 Distinguished Point (DP)

In 1982, Rivest suggested to use distinguished points as endpoints to reduce the number of memory accesses. When the table is too large, memory accesses have a great expense. Rather than generating a fixed chain length, the chain is iterated until an endpoint that satisfies a certain property is found. It is extensively studied in [18]-[69]-[62]-[53].

Definition 3.3.1 *[18] Let $K \in \{0,1\}^k$ and $d \in 1, \dots k-1$. Then K is a distinguished point (DP) of order d if there is an easily checked property which holds for 2^{k-d} different elements of K.*

Distinguished points (DP) are the range points (keystreams /ciphertexts) that satisfy a given criterion, for example, the last d bits are all zero. They reduce the memory accesses in the online phase, since a search will be done only if the ciphertext (or the iterations of ciphertext) is a distinguished point. It should be noted that the chain may enter an infinite loop if it contains no distinguished points. Only endpoints which encountered a DP in less than t iterations will be stored, the others will be discarded. Chains are generated until a DP is encountered and the triple $\{SP, EP, length\}$ is stored where $length$ is the length of the chain. If a DP is not reached until t_{max} iterations, that chain is discarded. Moreover, if the chain length is less than some t, say t_{min}, that chain is also discarded. If same DP occurs in different chains, the triple with maximum chain length is stored (since merging chains will have the same endpoint). Also, the tables are sorted with respect to the endpoints.

In online phase, when given a first ciphertext (c_0), keys will be generated until a distinguished point is found, and only then looked up in the memory. This greatly reduces the number of memory lookups. Starting by checking if c_0 is a DP then it is equal to any endpoints, EP_j, the key is found by iterating SP_j till $length_j - 1$.

If c_0 is not equal to any points of the last column, lookup the maximum $length_j$, calculate $f_j(c_0)$ and check if it is a DP then check for a match in the $(length_j - 1)^{st}$ column else take $j-1$. Check until whether c_0 is in the second column of the matrix.

Algorithm 3.3.1: PRECOMPUTATIONDP(*void*)

```
procedure TABLEDP(f)
  for $j \leftarrow 0$ to $m - 1$
    do
      select random $SP_j$;
      $tmp \leftarrow f(SP_j)$;
      for $s \leftarrow 0$ to $t - 1$
        do
          if ($tempisaDP$)
            then
              $EP_j \leftarrow tmp$;
              $length_j \leftarrow s$;
              $break$;
  store $(SP_j, EP_j, length_j)$ tuples in table $i$,sorted with respect to end points.
  If two chains have the same end point store the maximal length.;


main
  for $i \leftarrow 1$ to $r$
    do
      Choose DP-property of order $d$.choose random $R_i \ni E \circ R_i = f_i$ ;
      TABLEDP($f_i$)
```

Algorithm 3.3.2: ONLINEDP(c_0)

```
for $i \leftarrow 1$ to $r$
  do
  if $R_i(c_0)$ is a DP and any of the $EP_j$ is equal to $R_i(c_0)$
    then
      $tmp \leftarrow Sp_j$;
      for $s \leftarrow 1$ to $t - 1$
        do
          $tmp \leftarrow f_i(tmp)$;
      return ($tmp$)
    else
      length is the maximum $length_j$ of matrix;
      for $s \leftarrow 1$ to $length$
        do
        if $f_i^s(c_0)$ is a DP and any of the $EP_j$ is equal to $f_i^s(c_0)$
          then
            $tmp \leftarrow Sp_j$;
            for $k \leftarrow 1$ to $s - 1$
              do
                $tmp \leftarrow f_i(tmp)$; if $E(tmp, p_0) == c_0$
                  then return ($tmp$)
return ($keyisnotfound$)
```

Example 3.3.2 The random function used in this example is given Example 2.2.2 for $N = 2^6$. Let us take $d = 2$ (least significant 2 bits are zero). Thus, we have $D = 2^{n-d} = 16$ DP's. For $m = 2^5$, $t_{max} = 2^4$, $t_{min} = 2$. Starting from m points, we eliminate the chains which has length less than t_{min} and also identical endpoints.

$17 \rightarrow 57 \rightarrow 39 \rightarrow 9 \rightarrow 43 \rightarrow 41 \rightarrow 59 \rightarrow 24$

$18 \rightarrow 11 \rightarrow 13 \rightarrow 16$

$34 \rightarrow 10 \rightarrow 60$

Number of remaining rows is 3. Many endpoints are discarded since search space is small, there are many identical endpoints. For the above random graph, we may write at most 4 chains, since there are 16 DP in the graph in which 4 of them are terminal nodes. So, they do not have pre-images. 32, 12, 48, 40, 20, 28, 4 will cause chain lengths to be less than t_{min}. Thus, the last chain may be $55 \rightarrow 22 \rightarrow 36$. In online phase, we are waiting for a c_0 which is a DP. Since there are 2^{n-d} distinguished points, the number of table lookups is reduced by a factor of 2^d.

Online phase will be as in the following:

- Let $c_0 = 40$ is given, c_0 is not in the last column. Then $f(c_0) = 19$ is not a DP, $f^2(c_0) = 35$ is not a DP, $f^3(c_0) = 61$ is not a DP, $f^4(c_0) = 23$ is not a DP, $f^5(c_0) = 62$ is not a DP, $f^2(c_0) = 45$ is not a DP, it is not in table.
- Let $c_0 = 39$ is given, c_0 is not a DP. $f(c_0) = 9$ is not a DP, $f^2(c_0) = 43$ is not a DP, $f^3(c_0) = 41$ is not a DP, $f^4(c_0) = 59$ is not a DP, $f^5(c_0) = 24$ is a DP and in table. Then $SP = 17$, since $f^2(SP) = c_0$ then the key is $SP = 57$.

For whole space, in the online phase 12/40 ciphertexts are found from table. This example suggests that precomputation increases rapidly when the key space is small.

Advantages of distinguished points are given in [61] as the followings:

- Reduced table lookups
  In the online phase, if only a DP is encountered then the tables are searched. Table lookups is reduced by a factor of 2^d.
- Loop freeness
  If no DP is encountered till t_{max} iterations, then the chain is suspected to contain a cycle and it is discarded. Thus, the tables are free of loops.
- Merge freeness
  Since the chains with identical endpoint are discarded in Perfect DP tables (see Section3.3.3.1, merges will be discarded without additional cost (since tables are sorted, in any case).

Let α be the expected number of chains in a table (at most m) and β be the expected length of a chain (at most $t + 1$). Since the chain length is variable and a chain is discarded when it does not reach a DP, α and β play an important role in the success probability. An important question is how to choose t_{max} and t_{min}, depending on the average chain length, β, and also number of chains in a table, α, to have a high table coverage, hence a high probability of success. In his thesis [57], Mukhopadhyay stated that **in the literature there is no precise guideline about how to choose parameters**.

Effects of parameters:

- Number of distinguished points (depends on d)

  If d chosen to be too small then the memory access will increase in online phase. If it is too large, then the number of distinguished points will decrease, the number of identical endpoints will increase. So, in precomputation, more chains will be iterated, but lots of them will be discarded. Moreover, probability to reach a distinguished point will be small, than the average chain length will be high. Borst et al.[19] suggested to choose $d \approx \frac{1}{3}k$, $m \approx 2^{\frac{k}{3}}$ and $t \approx 2^{\frac{k}{3}+3}$ for a success rate of 0.55. In [69] for 40 bit DES, d is between 11 and 15. In [18] for 16 bit RC2 , d was chosen to be 5 and 6 for 20 bit RC2 d was chosen to be 6 and 7, for 24 bit RC2 d was chosen to be 7, 8 and 9. There is no reason why they preferred these values.

- t_{min}

  If chosen to be too large, it eliminates lots of DP's which will increase precomputation. If it is too small, it will increase the memory requirement.(See Table 3.3.1)

- t_{max}

  It should be chosen such that the probability to reach a DP in less than t_{max} operations is large. It would be better to choose t_{max} a bit larger than average chain length. If it is too large, it will consume a lot of time and space unnecessarily. As seen in table 3.3.1, for instance for $t_{min} = 2^3$, choosing t_{max} larger than 2^9 consumes unnecessary effort.

- m

  If chosen to be too large the number of identical endpoints will increase. So, as in Hellman method, it is sensible to construct r tables with different mask functions.

It should be noted that when discarding identical endpoints since longer chains are more often discarded then short chains which results in reduced average chain length.

Table 3.3: Effects of t_{min} and t_{max} for Reduced Sha-1 $N = 2^{28}$, d=6, $m = 2^{16}$

t_{min}	t_{max}	r_1	β_0	r_2	β_1	Coverage
2^5	2^7	31240	69	29160	69	1786350
	2^8	38858	89	33140	88	2444536
	2^9	40051	96	33444	93	2554662
	2^{10}	40051	96	33444	93	2554662
2^4	2^7	42869	56	38165	56	1988773
	2^8	50487	74	40257	72	2593928
	2^9	51680	79	40225	75	2690772
	2^{10}	51696	80	40219	75	2692027
	2^{11}	51696	80	40219	75	2692027
2^3	2^7	49828	50	43234	50	2037106
	2^8	57446	66	44130	62	2606526
	2^9	58639	71	43878	65	2696209
	2^{10}	58655	72	43878	65	2697300
	2^{11}	58655	72	43878	65	2697300

3.3.1.1 Variants of the Distinguished Point Method

The main idea of the method is eliminating the need to store the starting points so as to lessen storage requirements, but it will decrease memory only a factor of 2. Starting from special points iteration is done until a pattern is found [35]. For $m = t = N^{1/3}$ and $d = \frac{1}{3}log_2N$. When creating the $j - th$ Hellman chain of the $i - th$ table, starting points is taken to be:

$$SP_j^i = (0||i||j)$$

where each of the components are d bits. The chain is constructed until a point which has the most significant d bits is found to be j and store at $HT_i[j]$, the $j - th$ row of $i - th$ table. There is no sorting since starting points are not stored, but if a special pattern is not found then $HT_i[j]$ will be empty. Moreover applying t_{min} will discard some of the chains and this will yield empty Hellman table entries. It is stated that it is not mandatory to store chain lengths, because it will increase the probability of false alarm, and the online complexity will increase.

They offer a solution for empty chains to use starting points of the form

$$SP_j^i = (\tau||i||j)$$

where τ is incremented every time creation of the $j - th$ row fails, but in this case, it increases the storage requirement.

The second idea is applying DP to rainbow, since using DPs will vary the chain lengths, they suggest to use different tables for every chain length.

3.3.1.2 Distinguished Point Table on Permutations

We construct tables consisting of chains starting from distinguished points and ending in a distinguished point. Our aim is to increase the rate of success. A table of size D (the number of distinguished points) is constructed. To do this, we first list distinguished points as starting points. Then, for each starting point we iteratively apply f until a distinguished point is reached.

Since a random permutation has no terminal nodes; every node is in a cycle (See in Section 2.4). By constructing a table in this manner, it is possible to mark almost all points in every cycle by a good selection of D as in Figure 3.6.

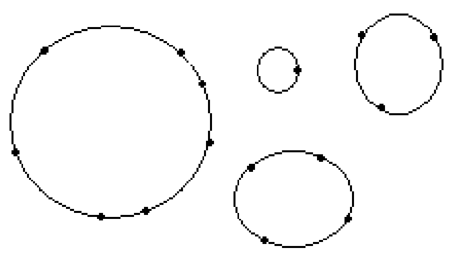

Figure 3.6: Variant Distinguished Point on Permutations

It is possible to have a high success rate $\approx$ %99 for a random permutation. To save memory it is possible to discard some points which are not in the region $[t_{min}, t_{max}]$ as in DP technique but it is trival that the success rate will get smaller. The memory complexity of the attack is equal to the number of distinguished points.

The question is how to choose distinguished property d resulting in the number of distinguished points D to have at least one distinguished point in each cycle:

Let N be the number of all points, k is the length of smallest cycle in a random permutation and D is the number of special points. The probability that there is no special point in the set

A which has k elements is:

$$P'(k,D,N) = \frac{\begin{pmatrix} N-k \\ D \end{pmatrix}}{\begin{pmatrix} N \\ D \end{pmatrix}} = \frac{(N-k)!D!(N-D)!}{D!(N-k-D)!N!}$$

$$= \frac{\left(1-\frac{k}{N}\right)^{N-k}\left(1-\frac{D}{N}\right)^{N-D}}{\left(1-\frac{k+D}{N}\right)^{N-k-D}} \approx \frac{e^{-k+\frac{k^2}{N}-D+\frac{D^2}{N}}}{e^{-k-D+\frac{(k+D)^2}{N}}} = e^{-\frac{2DK}{N}}$$

Then the probability that there is at least one point in the set A is:

$$P(k,D,N) = 1 - e^{-\frac{2DK}{N}}.$$

To choose the optimal parameters, let our desired success rate be P. Our aim is to find appropriate D such that the probability of success of the trade-off is at least P:

$$P(k,D,N) = 1 - e^{\frac{-2Dk}{N}} > P$$

Then

$$D > \frac{N}{2k}\ln\left(\frac{1}{1-P}\right).$$

For instance, let $D = N^{2/3}$ then if the smallest cycle length k is greater than $2.3N^{1/3}$ there will be at least one special point in each cycle with probability, $P = 0.99$. As given in Section 2.4,

$$\text{Average cycle length of a random permutation} = \frac{n}{\#components} = \frac{n}{lnn}.$$

For a random mapping, we can construct the same table. If there is no distinguished point in the chain, then the chain will enter in an infinite loop. In order to prevent this undesired situation, we iterate until a predefined time, t, if the chain does not encounter a distinguished point until t then that chain will be discarded. It should be noted that t will be smaller than the average cycle length (See in 2.3.2) of a random mapping.

Note: Assume that the state update function of a stream cipher is linear or it is easy to calculate S_t, the t^{th} state when S_0 is known. Then, in offline phase, the variant distinguished point method can easily be applied and also TMTO table can be constructed by starting from random S_0 and corresponding n bit keystream then S_{2^m} and corresponding keystream then S_{2*2^m} and till $S_{2^n-2^m}$. However, for today's stream ciphers, the state transition functions are highly nonlinear so choosing states reasonable according to the particular cipher may be meaningful as in [15].

3.3.2 Rainbow Table

Rather than constructing r tables with different reduction functions, Oechslin[61] proposed new tables, called rainbow tables in which a successive reduction function is used for each column in the chain. There are t reduction functions. The main advantage of this method is to reduce the online time complexity by a factor of 2. The total number of computations is given as $\frac{t(t-1)}{2}$ whereas in classical method it is rt.

$$k_{1,1} \xrightarrow{f_1} k_{1,2} \xrightarrow{f_2} k_{1,3} \cdots \xrightarrow{f_t} k_{1,t}$$

Instead of constructing r tables with $m \times t$ elements, choosing $m \times r$ random starting points, $m \times t \times r$ tables are constructed Figure 3.7.

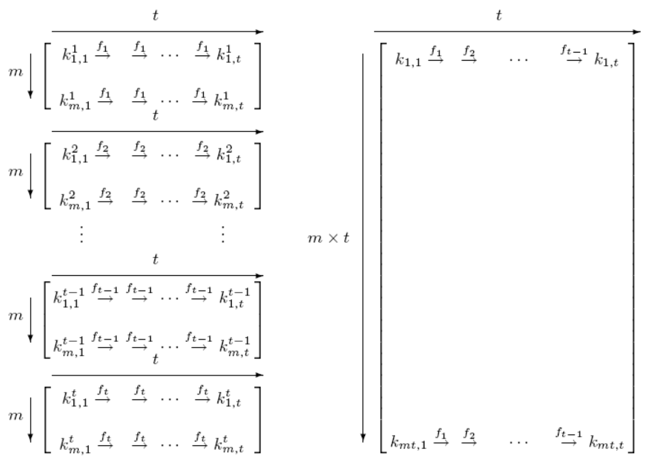

Figure 3.7: Classic Table vs. Rainbow Table

Algorithm 3.3.3: RAINBOWPRECOMPUTATION(*void*)

```
for $j \leftarrow 1$ to $mt$
  do
    select random $SP_j$,choose random $R_0 \ni E \circ R_0 = f_0$;
    $tmp \leftarrow f_0(SP_j)$;
    for $i \leftarrow 1$ to $t - 1$
      do
        choose random $R_i \ni E \circ R_i = f_i$ ;
        $tmp \leftarrow f_i(tmp)$;
    $EP_j \leftarrow tmp$;
store $(SP_j, EP_j)$ pairs in table sorted with respect to end points;
```

In online phase, when given a first ciphertext (c_0), by checking if $R_t(c_0)$ is equal to any points of the last column EP_j for some j, $1 \leq j \leq mt$ then the $k_{j,t-1}$ is the desired key. Starting from SP_j, after $t - 1$ iterations $k_{j,t-1}$ can be recalculated. If $R_t(c_0)$ is not equal to any endpoints, by repeatedly applying $R_{t-i}, f_{t-(i-1)}, f_{t-(i-2)}, \ldots, f_{t-1} \ni 0 \leq i \leq t - 1$ to c_0 and it is checked that whether the output is equal to any endpoints. If a match is found (at s_{th} iteration), starting from SP_j , $k_{j,s-1}$ is recalculated and checked whether $c_0 = E(k_{j,s-1}, p_0)$ or not. The total number of calculations is $\frac{t(t-1)}{2}$.

Algorithm 3.3.4: RainbowOnline(c_0)

```
if any of the $EP_j$ is equal to $R_t(c_0)$
  then
    $tmp \leftarrow Sp_j$;
    for $s \leftarrow 1$ to $t-1$
      do
        $tmp \leftarrow f_s(tmp)$;
    return ($tmp$)
  else
    for $s \leftarrow 1$ to $t-1$
      do
        $val \leftarrow R_{t-s}(c_0)$;
        for $k \leftarrow s$ to 1
          do
            $val \leftarrow f_{t-k-1}(val)$;
        if any of the $EP_j$ is equal to val
          then
            $tmp \leftarrow Sp_j$;
            for $k \leftarrow 1$ to $s-1$
              do
                $tmp \leftarrow f_k(tmp)$;
            if $E(tmp, p_0) == c_0$
              then return ($tmp$)
return ($keyisnotfound$);
```

Example 3.3.3 Let us take $m = 2^3$, $t = 2^3$,$r = 1$. In each column, the mask function is taken as $f(x) \oplus i \ni 0 \le i \le 6$.

$29 \to 36 \to 23 \to 60 \to 23 \to 58 \to 45 \to 8$

$51 \to 3 \to 52 \to 60 \to 23 \to 58 \to 45 \to 8$

$13 \to 16 \to 5 \to 49 \to 7 \to 8 \to 37 \to 4$

$36 \to 22 \to 37 \to 0 \to 12 \to 52 \to 59 \to 30$

$5 \to 51 \to 2 \to 57 \to 36 \to 18 \to 14 \to 59$

$57 \to 39 \to 8 \to 34 \to 9 \to 47 \to 35 \to 59$

$6 \to 5 \to 50 \to 50 \to 51 \to 7 \to 9 \to 45$

$17 \to 57 \to 38 \to 62 \to 46 \to 41 \to 62 \to 43$

There are 37 distinct points for the above construction (the last column is omitted since it will not give any clue for the key). Thus the coverage of table is 37.

Assume in online phase we are given c_0. Since we have only the tuple $\{SP, EP\}$ we are searching if $R_i(c_0)$ is in the last column.

- Let $c_0 = 24$ is given, $R_6(c_0) = 24 \oplus 6 = 30$ is in table, $SP = 36$, since $h^6(SP) = R_6(c_0)$ then the key is 59.
- Let $c_0 = 7$ is given, $R_6(c_0) = 7 \oplus 6 = 1$ is not in table , $f(c_0 \oplus 5) \oplus 6 = 61$ is not in table, $f(f(c_0 \oplus 4) \oplus 5) \oplus 6 = 16$ is not in table, $f(f(f(c_0 \oplus 3) \oplus 4) \oplus 5) \oplus 6 = 57$ is not in table, $f(f(f(f(c_0 \oplus 2) \oplus 3) \oplus 4) \oplus 5) \oplus 6 = 59$ is in table, $SP = 5, 57$, since $h^3(SP)! = c_0 \oplus 2$ then false alarm.$f(f(f(f(f(c_0 \oplus 1) \oplus 2) \oplus 3) \oplus 4) \oplus 5) \oplus 6 = 53$ is not in table.
- Let $c_0 = 15$ is given, $R_6(c_0) = 15 \oplus 6 = 9$ is not in table, $f_6(R_5(c_0)) = 58$ is not in table, $f_6(f_5(R_4(c_0))) = 38$ is not in table, $f_6(f_5(f_4(R_3(c_0)))) = 30$ is in table, $SP = 36$, key is 0.

For whole space, in the online phase 26/40 keys are found from table.

The advantages of rainbow tables are given in [61] as :

- If two chains collide, they merge only if the collision appears at the same column. If the collision does not appear at the same column, since they will continue with different reduction functions, they will not merge. Since merges will result with identical endpoints, identical endpoints are discarded in Perfect Rainbow Tables (See Section 3.3.3.1). It should be noted that if identical endpoints are discarded in Rainbow tables when compared with distinguished points it will increase the precomputation since there are t tables in DP tables, but in rainbow tables, the identical points will be huge with in one table. This will result in excessive precomputation.
- Rainbow tables have no loops since each column has different reduction functions.
- Rainbow chains have a constant length. It reduces false alarms.

Oechlin has implemented an attack on MS-Windows password hashes. Using 1.4 GB of data (two CD-ROMs) he cracked 99.9% of all alphanumerical password hashes (2^{37}) in 13.6 seconds whereas it takes 101 seconds with using distinguished points [61]. The coverage is given as 78.8%. It is possible to construct a table with given coverage by Perfect Rainbow Table (See Section 3.3.3.1) with $m_{max} = 2^25 = 38223872$. It should be noted that the precomputation exceeds exhaustive searching.

The main disadvantage of the attack is pointed out by Barkan et al. [8] that the cost of storage in the rainbow method is substantially higher than for DP method.

Trade off Curve

The number of tables constructed is $r = 1$. The memory required is $M = mt$, the time required for online phase is $T = \frac{t^2}{2}$ then trade off curve is obtained as:

$$TM^2 = N^2 \text{ where } 1 \leq T \leq N$$

3.3.2.1 Rainbow-Like Schemes

Barkan et al gave several Rainbow-like schemes which uses multiple data points in [8] and also gave an upper bound on the coverage of TMTO methods.

Thin-Rainbow: Reduces the number of colors (reduction functions) as:

$$f_1 f_2 \cdots f_S f_1 f_2 \cdots f_S \cdots f_1 f_2 \cdots f_S$$

where S is the number of colors (repeated t times). But it loses the main advantage (reduce the oline time complexity by a factor of 2) of Rainbow Tables.

Thick-Rainbow: Reduces the number of reduction functions as:

$$\underbrace{f_1 f_1 \cdots f_1}_{\text{t times}} \underbrace{f_2 f_2 \cdots f_2}_{\text{t times}} \cdots \underbrace{f_S f_S \cdots f_S}_{\text{t times}}$$

Fuzzy-Rainbow: In this case the notion of distinguished points is used. The reduction function of f is switched when a DP is determined. It is claimed that DP occurs with probability t^{-1}.

General Model of TMTO for the inversion of a random function In [8], Barkan et al. gave a general model which contains all techniques of TMTO. The success rate of TMTO techniques depends on the chosen random function's (mask function) graph, which is called "stateful random graphs". In other words, the graph representation of functions depends on a "hidden state" such as the table number in Hellman scheme. They gave an upper bound on the number of images for which f can be inverted using a trade off scheme with S hidden states and then gave a lower bound on the number of hidden states. In addition, they gave a lower bound on its worst case time complexity $T = \frac{N^2}{M^2 lnN}$.

3.3.2.2 False alarm detection using Checkpoints

In [3]-[4] to gain cryptanalysis time that is wasted by false alarms ($\approx 50\%$ of online time), Avoine et al. defined a method which is called checkpoints by using additional memory. The main aim is to detect a value matching with an endpoint that will lead to a false alarm, without regenerating the chain from its starting point.

While preparing tables they define a set of positions x_i as checkpoints. For a given function G (e.g. G is a parity test) for each x_i of each chain, $G(k_{j,x_i})$ is stored. In the online phase, while searching ciphertext and iterations of ciphertexts in EPs, the values for G is also calculated. If they are not same at least for one checkpoint, it is clear that this is a false alarm without regenerating the whole chain.

3.3.3 Perfect Tables

Merges in chains decrease the coverages of tables. Generating a table without merges is another trade because it will increase the precomputation time. Perfect tables are tables in which merges are rarely occurred. In both Rainbow [61]-[3] and DP tables [19], merging chains will have identical endpoints. Firstly more chains are generated, then identical endpoints are discarded. Throwing away such chains will increase the precomputation time. In [3] Avoine et al stated that constructing a perfect Hellman table is not efficient since all chains have to be looked up. We give in Section 3.3.3.1 a new construction technique that constructs Perfect Hellman Tables.

It is clear that when two chains collide in the same column as in the first two rows of Ex. 3.3.3, the endpoints will be identical. By discarding one of these rows, merge free tables can be constructed. However, this means that starting from m distinct points, it is expected to remain $m_{\max} = \dfrac{m}{1+\frac{tm}{2N}}$ points [3].

In Ex.3.3.3 starting from $m = N$. The number of remaining (SP,EP)' s is 15. For whole space, in the online phase, 35/40 keys are found from table. This example shows that precomputation increases rapidly when the key space is small.

The number of perfect table is given

$$l = \left\lceil -\frac{\ln(1-p)}{2} \right\rceil$$

where p is the success rate then

$$t \geqslant \frac{-Nln(1-p)}{m_{\max} l}$$

3.3.3.1 Perfect Hellman

Assuming the encryption function is modeled as a random mapping the set into itself, if we consider the graph representation of a random mapping, we choose our start points as terminal nodes (we mean the points which have no preimage) as shown in Figure 3.8.

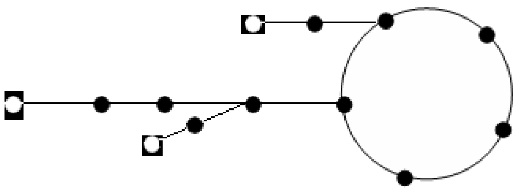

Figure 3.8: Perfect Hellman Construction

It is clear that to find the terminal nodes, it is necessary to increase precomputation from N to $2N$. Since there are $N' = N * e^{-1}$ terminal nodes, starting from N' by iteratively applying f we construct a Hellman table. In [3], starting from m_0 random points, maximum number of perfect chains of length t is given as

$$m(t, m_0) = \frac{m_0}{1+\frac{tm_0}{2N}}$$

and maximum number of chains of length t by starting with m_0 equal to N is given as

$$m_{max} = \frac{2N}{t+2}.$$

Starting from N' points, maximum number of chains of length t of Perfect Hellman table will be $m_{max} \approx \dfrac{2N}{t+6}$.

This shows that the complexity of Perfect Hellman table is approximately equal to the complexity of Perfect Rainbow table.

3.3.3.2 Perfect Tables with Threshold

If a perfect table is constructed by eliminating the chains which has identical endpoints, we believe that if the merge occur in the columns ($< t-\epsilon$), a lot of non merging points are wasted. Therefore, we propose to use threshold after some column, in other words, starting from N points if identical endpoint is encountered in the same column, it will be discarded only if it occurs before some threshold. So, it is important to find out where to put this threshold. For instance, in Table 3.4, the effects of threshold (T) on coverage and computation are given.

For a Perfect Table without threshold, the expected computation time can be given as

$$P = m_0 + m_1 + \cdots + m_{max}.$$

and success rate is

$$P_{success} = 1 - e^{-\frac{m_0+m_1+\cdots+m_{max}}{N}}.$$

On the otherhand, for Perfect table with threshold, the expected computation time can be given as

$$P = m_0 + m_1 + \cdots + m_{t_0} + m_{t_0} + \cdots + m_{t_0}.$$

and success rate is

$$P_{success} = 1 - e^{-\frac{m_0+m_1+\cdots+m_{t_0}+(t-t_0)m_{t_0}}{N}}.$$

And the ratio of the success rates of two method is

$$\lambda = \frac{e^{-\frac{m_0+m_1+\cdots+m_{t_0}+(t-t_0)m_{t_0}}{N}}}{e^{-\frac{m_0+m_1+\cdots+m_{t_0}+\cdots+m_{\max}}{N}}} = e^{-\frac{(t-t_0)m_{t_0}+m_{t_1}+\cdots+m_{\max}}{N}}$$

It is clear that there is a trade off between coverage and precomputation that can be seen in Table 3.4.

Table 3.4: Threshold Results for $m = 2^{22}$ $t = 2^{11}$

T	*MFinal*	*Computation*	*Coverage*	*T*	*MFinal*	*Computation*	*Coverage*
0	4194304	0.0000P	1.0000%	1056	7892	14.5909P	0.9630%
32	239323	120.9370P	1.0000%	1088	7655	14.5357P	0.9607%
64	124773	66.2363P	0.9998%	1120	7465	14.4930P	0.9587%
96	84386	47.2705P	0.9997%	1152	7254	14.4471P	0.9563%
128	63894	37.8074P	0.9994%	1184	7045	14.4033P	0.9537%
160	51313	32.0940P	0.9991%	1216	6857	14.3653P	0.9511%
192	42785	28.2865P	0.9987%	1248	6677	14.3303P	0.9484%
224	36743	25.6357P	0.9983%	1280	6504	14.2980P	0.9457%
256	32106	23.6367P	0.9978%	1312	6342	14.2689P	0.9429%
288	28547	22.1296P	0.9972%	1344	6195	14.2437P	0.9401%
320	25795	20.9851P	0.9965%	1376	6060	14.2216P	0.9375%
352	23481	20.0405P	0.9958%	1408	5916	14.1991P	0.9344%
384	21590	19.2830P	0.9950%	1440	5745	14.1736P	0.9305%
416	19961	18.6429P	0.9942%	1472	5604	14.1537P	0.9270%
448	18578	18.1102P	0.9933%	1504	5469	14.1357P	0.9234%
480	17295	17.6259P	0.9923%	1536	5362	14.1223P	0.9204%
512	16226	17.2304P	0.9912%	1568	5249	14.1090P	0.9170%
544	15251	16.8770P	0.9901%	1600	5149	14.0979P	0.9138%
576	14406	16.5773P	0.9888%	1632	5047	14.0874P	0.9103%
608	13670	16.3218P	0.9876%	1664	4933	14.0766P	0.9062%
640	12960	16.0808P	0.9862%	1696	4835	14.0680P	0.9025%
672	12354	15.8797P	0.9849%	1728	4753	14.0615P	0.8992%
704	11788	15.6963P	0.9834%	1760	4669	14.0554P	0.8956%
736	11256	15.5279P	0.9818%	1792	4590	14.0503P	0.8920%
768	10802	15.3876P	0.9802%	1824	4505	14.0454P	0.8880%
800	10398	15.2659P	0.9787%	1856	4436	14.0420P	0.8845%
832	10010	15.1519P	0.9770%	1888	4368	14.0392P	0.8810%
864	9662	15.0524P	0.9753%	1920	4293	14.0366P	0.8768%
896	9294	14.9499P	0.9733%	1952	4219	14.0346P	0.8725%
928	8997	14.8695P	0.9715%	1984	4148	14.0333P	0.8682%
960	8712	14.7945P	0.9696%	2016	4084	14.0326P	0.8641%
992	8413	14.7181P	0.9674%	2048	4029	14.0323P	0.8605%
1024	8150	14.6529P	0.9653%				

3.3.4 Time/ Memory/Key (TMK) Tradeoff

The Time/Memory/Data in Section 3.4.2 trade-off in the case of stream ciphers can be applied to the block-cipher Time-Memory-Key case. It is clear that if multiple data is available, this will improve the effectiveness of a TMTO attack since it will decrease the precomputation time. In [37], Hong and Sarkar proposed some ways to use multiple data when applying TMTO on block ciphers.

- fixed plaintext is enciphered multiple times with distinct keys

- the plaintext is enciphered multiple times using the same key under different IVs.
- the plaintext is enciphered several times in a single long session.
- the k-block plaintext is consisting of k identical blocks.

In [13], Hellman's algorithm was generalized for the case of multiple data (a fixed plaintext enciphered under different keys), which resulted in a formula $N^2 = TM^2D^2$, $1 \leq D^2 \leq T$, $P = N/D$. It is clear that the actual table preparation time will be less than exhaustive search in this case. In the online phase, D points are given, that is enciphered under different keys, the goal of the attack is to find the preimage of any one of these points. Since D points are given, a set of tables are prepared covering N/D of the domain points by choosing $r = t/D$, resulting $P = mtr = mt\frac{T}{D} = N/D$.

3.3.4.1 Modes of Operations

Block ciphers encrypt n-bit plaintext blocks to n-bit ciphertext blocks; n is called the block length. Since the plaintext may be of any length, there are several modes of operations (See [52]-[27]-[17]) such as Electronic Code Book (ECB), Cipher Block Chaining (CBC), Output Feedback Mode (OFB), Counter Mode (CTR). They require an initialization vector (IV). In some cases, last block is padded to bring the plaintext length to a multiple of the block size. There are some techniques in the literature that are called message padding schemes. Adding null bytes may be the easiest way. In [38]-[37], Hong and Sarkar showed that modes of operations are vulnerable to TMTO attacks in contradiction with the general belief. Modes of operations which are susceptible to TMTO attacks can be summarized as in the following [38]:

- ECB is vulnerable to TMTO under chosen plaintext scenario if the plaintext is chosen to be as long as the key.
- CTR is vulnerable to TMTO chosen plaintext scenario if counter update is predictable.
- OFB is vulnerable to TMTO known plaintext scenario if key length is any longer than block or IV length.

- CBC, CFB are vulnerable to TMTO chosen plaintext scenario if key size is any longer than IV size (using multiple data 3.3.4). In [9], Biham showed that CBC is vulnerable only if IV is considered to be a part of the secret.

- OCB, OMAC use MAC addition to ciphertext. They are vulnerable to TMTO chosen plaintext scenario if key is longer than nonce. The one way function is defined as (key, nonce) pair to (ciphertext||MAC).

CMC, EME are also considered in [38].

3.3.5 FPGA implementations

Amirazizi and Hellman [1] proposed time/memory/processor trade-off. More processors executed in parallel by sharing a large memory. The hardware design for a key search machine is firstly proposed by Quisquater et al. [63] for a 40-bit DES. FPGA (Field Programmable Gate Array) is a programmable device that can be used to perform some cryptographic encryption at very high speed. The precomputation is performed with FPGA design using distinguished points technique in [63]-[69]. The hardware design on the rainbow was proposed by Mentens et al.[53], they presented an FPGA architecture for cracking UNIX passwords. Rainbow table was precomputed with one salt and estimated that precomputation of all salts will be nearly one year for UNIX passwords.

3.4 TMTO on Stream Ciphers

Stream ciphers behave like random one way functions whenever we try to obtain key/state using the keystream. TMTO on stream cipher is different from block cipher. If an attacker can find any of the actual states, then he can find the rest of the keystream. Thus, he does not have to find the initial state (or key). In many cases, with reverse engineering he may find the initial state. In addition, when a $D + log(N) - 1$ bits of keystream is given (there is no difference between known plaintext and chosen plaintext attack when a stream cipher is taken into account), he will have D overlapping prefixes. Since Hellman's table for block ciphers is associated with a particular plaintext block, the multiple prefix is useless (multiple data as in Time/Memory/Key Trade Off is not considered). However, for the stream ciphers precomputed table can be used on multiple prefixes.

Two different scenarios may be stated when stream ciphers are considered as oneway functions. The one way function may be a function from (internal state→ keystream prefix (state size)) or from (key+iv →keystream prefix (key+iv size)). In the next two sections we will summarize the Babbage-Golic (BG) attack and Biryukov-Shamir (BS) attack which carry out the first scenario.

3.4.1 Babbage-Golic Attack

TMTO on stream ciphers was firstly proposed by Babbage [5] in 1995 and Golic [31] in 1997 through independent works. Golic applied TMTO on the alleged A5 stream cipher [31].

Let internal state size is n. Then the number of possible states is $N = 2^n$.

- S_i is the i^{th} state where $S_i = (s_{i,1}, s_{i,2}, \dots, s_{i,n})$
- $StateUpdate$ is the State Update function
- z_i is the i^{th} bit of keystream
- Out is the output function of a stream cipher.

Generation of i^{th} bit of keystream z_i is as in the following:

$$z_i = Out(S_{i-1})$$
$$S_i = StateUpdate(S_{i-1})$$

where the initial state S_0 is derived from a secret key.

In offline phase, choosing 2^m random states $S_1, S_2, \dots, S_{2^m}$ the corresponding keystream prefixes $K_1, K_2, \dots, K_{2^m}$ where $K_i = z_{i,1}, z_{i,2}, \dots, z_{i,n}$ are stored in a table as in the below. Table is also sorted with respect to the keystream sequence. Therefore, the precomputation time is $P = M$

$$s_{1,1}, s_{1,2}, \dots, s_{1,n} \rightarrow z_{1,1}, z_{1,2}, \dots, z_{1,n}$$

$$s_{2,1}, s_{2,2}, \ldots, s_{2,n} \rightarrow z_{2,1}, z_{2,2}, \ldots, z_{2,n}$$

$$\ldots$$

$$s_{m,1}, s_{m,2}, \ldots, s_{m,n} \rightarrow z_{m,1}, z_{m,2}, \ldots, z_{m,n}$$

In Online Phase, given $D + n - 1$ bit keystream means $\approx D$ subsequences (of length n) of possible keystream, $K_1, K_2, \ldots, K_D$ is derived. Each one is searched in the precomputed table, if a match is found then with high probability the corresponding state is the desired internal state then, remaining keystream bits after the prefix is computed. The online time is $T = D$.

In[5], Babbage suggested as a design principle for stream cipher that a state size of $2i$ bits is desirable for a secret key length of i bits.

BG's Curve

$TM = N$ for $1 \leq T \leq D$ and $P = M$.

3.4.2 Biryukov-Shamir Attack- TMD Trade Off

Biryukov and Shamir combined BG scheme with Hellman attack. As in Hellman construction by r different functions, r different tables is constructed iteratively applying encryption function. However, different from Hellman approach, since D keystream prefixes are given, the total number of points covered by a all tables is reduced from N to N/D. There are two possible ways of doing this, they proposed to reduce the number of tables instead of making small tables. In Biryukov-Shamir(BS) construction, there are $r = t/D$ tables each having m rows, therefore $t > D$. Consequently, $M = mt/D$, $P = N/D$, $T = \frac{t}{D}tD = t^2$ with the condition $T > D^2$.

BS's Curve

$TM^2D^2 = N^2$ for $D^2 \leq T \leq N$ where $P = N/D$.

$P = T = 2^{2n/3}, M = D = 2^{n/3}$ is given as an appropriate selection for BS attack for N up to 100 [11].

3.4.3 Extensions and Refinements

3.4.3.1 Biryukov-Shamir-Wagner Attack

Distinguished points applied on block ciphers which reduces the number of table lookups was carried out on stream ciphers [14, 15]. As in DP on block cipher, the aim is to reduce the number of expensive disk operations by a factor of t. Apart from block ciphers, Biryukov-Shamir-Wagner (BSW) sampling on stream ciphers is dependent on cipher structure. Using BSW sampling, some recent ciphers are analyzed such as A5/1 [15], Mickey and Grain. These studies will be summarized in the next section.

Special states are the states which generates an output prefix with a fixed pattern such as r zero bits. In some stream ciphers, it is possible to enumerate all the special states without expensive trial and error, which is called BSW sampling. If the cipher has sampling resistance $R = 2^{-r}$, each special state can be associated with a short name of $n - r$ bits and a short output of $n - r$ bits. Then a new random mapping is defined over a reduced space of 2^{n-r} points.

When applying BSW sampling on BG attack, $P = M$ increases to $P = M * 2^r$, because to find special states we have to try a larger number of random states. Only a special prefix in the given data have to be looked, online time $T = D$ decreases to $T = D * frac12^r$. Total memory reduced from mt to mt/D but D data are tried. Trade off curve is given as $TP = MD = N$ for $1 \leq T \leq D$.

When applying BSW sampling on BS attack, $P = N/D$ remains unchanged, since to find special states we do not have to try a larger number of random states, we wait for the special endpoints to occur randomly during our chain construction. Trade off curve is given as

$TM^2D^2 = N^2$ for any $D^2 \leq T \leq N$.

3.4.3.2 Chosen Initial Value

Dunkelman and Keller presented a new approach using a publicly known initial value (IV). By choosing in advance some IVs TMTO is applied to streams produced using these IVs. The one way function to invert in this scenario is a function from (key $\rightarrow$keystream prefix(key size). For each chosen IV, the attacker prepares (Hellman/Rainbow) Tables. In the online phase, the attacker waits until IV_i is used for some i and then applies the TMTO attack with the tables prepared for that IV_i. If both the secret key and the IV are of length n, it is possible to mount an attack with data, time, and memory complexities of $2^{4n/5}$ [26]. Hence, when IV is chosen in a deterministic manner or incremented in a deterministic manner, this attack have the success rate of 0.40.

3.4.4 Design Principles for TMTO

To resist against TMTO, Golic [31] and Babbage[5] suggested to use state size which should at least be the twice of the keysize. Hong and Sarkar [38] suggested that IV length should not be less than the key length and it should not be used in predictable way, and state size should be at least the size of key plus IV. These results are discussed in [72]-[22]. The authors suggested that eStream [28] designers should choose the IV length that is equal to the key length.

In both [38] and [37], Hong and Sarkar stated some remarks on design principles of TMTO as:

- "Putting a restriction on how many frames are encrypted before the master key is renewed does not stop this attack completely. The attacker still gets to know one of the many master keys."
- "Making the state initialization process more complex has completely no effect on this TMTO attack. Neither does the size of the internal state of the stream cipher affect this TMTO in any manner."
- "The known part of keystream need not be at the very beginning. As long as they are fixed positions in the keystream, they do not even need to be continuous. The oneway function can be defined to match the known part."
- "If IV is XORed into the key before being placed into the internal state, we could set the domain of the oneway function to be at that position. In general, the domain of f should be at the point of least entropy occurring during the initialization process."
- "Using IVs in a predictable manner effectively reduces the IV space, making TMTO more efficient."

3.4.5 Applications on Stream Ciphers

A5/1

A5/1 is a stream cipher (See details in Appendix A.1) used to provide privacy in the GSM cellular telephone standard. It was kept secret until 1994. A5/1 and A5/2 were reverse-engineered by Briceno, Goldberg and Wagner [47]. [10] Golic [31] presented an attack on A5/1 based on solving a system of 2^{41} linear equations which has time complexity of about 2^{40}. He also applied TMTO on A5/1. In offline phase, a table is prepared as described in Section 3.4.1. The attack is successful if $TM \geq N$ where $D \approx T/102$ data is required with same secret key and different public key. Biryukov, Shamir and Wagner [15] showed that it is easy to directly enumerate the 2^{48} out of the 2^{64} states whose outputs start with $r = 16$ zeros. The BSW sampling of resistance of A5/1 is $R = 2^{-16}$. Since by loading 11 specified bits to the right of the clock control taps in both Register2 (R2) and Register3 (R3) and loading the remaining bits arbitrarily, we can uniquely determine the clock control of the three registers for the next few cycles, and thus determine the identity of the bits that enter their most significant bits and affect the output. We can keep this process alive for about 16 clock cycles since each register moves only 3/4 of the time. In [15], the key is computed in about one second during the first two minutes of the conversation on a single PC using the BSW sampling with 2^{42} preprocessing and memory complexity of four 73 GB hard-drives. Biham and Dunkelman proposed a different attack [10] with total work complexity of $2^{39.91}$ of A5/1 clockings, given $2^{20.8}$ known plaintext.

LILI-128

LILI-128 is a synchronous stream cipher with a 128 bit key (See details in Appendix A.2) proposed to NESSIE project[59] by Dawson, Clark, Golic, Millan, Penna and Simpson. Babbage proposed a generic TMD trade off in [6]. In [64], a tradeoff attack using BSW sampling (R = 44) against LILI-128 is proposed using 2^{64} bits of keystream, a lookup table of 2^{45} 89-bit words and 2^{48} DES operations. The attack may be summarized as follows: Saarinen proved that $LFSR_d$ is clocked exactly $\Delta_d = 5 * 2^{38} - 1$ times for each $\Delta_c = 2^{39} - 1$ times $LFSR_c$ is clocked.

- Load 89 bits of $LFSR_d$ randomly
- It is possible to extract 45 bits from f_d sampled Δ_d steps apart. Thus, 45 bit vector is

sufficient to index 89 bit register.

MICKEY

Mickey is a stream cipher proposed to eSTREAM [28] by Babbage and Dodd (See details in Appendix A.4.2) and designed for restricted hardware environments. In [36], a trade off attack using BSW ($R = 2^{27}$) sampling against Mickey version 1 is proposed by Hong and Kim. The attack may be summarized as follows:

- Load 80 bits of S and 53 bits of R (starting from r_1) randomly
- To allow the keystream to start with 27 zeros, r_0 and $r_{54}, \ldots, r_{79}$ can be filled with intermediate variables by writing some linear functions since it is possible to calculate control bits.

This allows a TMD trade off attack with sampling resistance 2^{27}. For instance, for data complexity D = 2^{60} for $P = N/2^{27} = 2^{100}$, $T = M = 2^{66}$. Although precomputation time is greater than exhaustive search, the authors preferred to increase the state size to 200 bits which is 2.5 times the key size in Mickey 2.0 [49] to resistant to these types of attacks.

LEX

Lex (Leak Extraction) is a stream cipher proposed to eSTREAM [28] by Biryukov (See details in Appendix A.4.1). It uses AES in OFB mode. A TMD trade off attack is applied on Lex by Dunkelman and Keller in [25]. This attack requires $D = 2^{36.3}$ keystream produced under the same key with different IVs. According to this attack, Lex was discarded from the final portfolio of eSTREAM. It is possible to mount an attack with BSW sampling ($R = 2^{32}$), $D = 2^{88}$, $T = M = 2^{112}$ [25].

GRAIN

Grain is a stream cipher proposed to eSTREAM [28] by Hell, Johansson and Meier (See details in Appendix A.4.3) and designed for restricted hardware environments. Grain has a low resistance to BSW sampling ($R = 2^{16}$), to recover the internal state of Grain v1 [50] using $T = M = 2^{71}$), and $D = 2^{53.5}$ bits of known keystream. The cost of the precomputation is $P = 2^{106.5}$.

”Every bit that enters the registers remains unused for at least 16 clock cycles, since the highest register indices occurring in the state update functions are n_{i+63} and l_{i+64} respectively. This enables implementors to compute up to 16 clocks of the cipher in parallel, greatly speeding up the cipher with only a moderate increase in gate count" [16]. Given the value of 133 particular state bits of Grain and the first 18 keystream bits produced from that state, another 18 internal state bits can be deduced efficiently.

This does not lead to a practical attack due to the cost of the precomputation which is far above exhaustive searching. The authors preferred to increase the size to 128 bits and IV size to 96 bits in Grain-128 [49] in order to be resistant to these types of attacks.

3.5 Statistical Tests Based on Random Mappings

There are various statistical test suites such as NIST [60], DIEHARD [48] that contain a collection of tests which evaluate whether the sequence is generated by a random source or not. In this section we will deal with tests based on random mappings over stream ciphers and our main aim is to try to find suitable test statistics which give some clues for applying TMTO. It is possible to generate a keystream and apply randomness tests on stream ciphers as in [51]. We considered the chosen IV approach in which keystream sequences are generated using a random key and different IVs. Since key and IV sizes are large to apply a statistical test on a stream cipher defined as a random mapping, we define random mappings as functions from a subset of key and IV to keystream prefix (which have the same size as key + IV).

We define three tests in our paper [71] called as Coverage Test, ρ Test and Distinguished Point Coverage Test.

In Coverage Test, as in Babbage-Golic attack (See Section 3.4.1) a table is constructed by defining a one way function from the key+iv (define size of key+iv as n) to the corresponding keystream prefix in which $n - l$ bits are fixed with random value. For all possible 2^l IVs, l bits keysteram prefixes are generated and the coverage of keystream prefixes is calculated. The coverages are calculated repeatedly with different random IVs. To test if these coverages are different from expected distribution, Pearson's Chi Square χ_2 test is applied (Refer to [71]-[70] for details). Resulting a high coverage shows that the mapping is close to a permutation. The ciphers close the permutations are more vulnerable to trade off attacks.

In ρ Test, as in Biryukov-Shamir attack (See Section 3.4.2) a table is constructed by defining one way function as described in Coverage test by iteratively applying until a repetition is encountered. This is repeated with different random $n-l$ bits IVs then the values are compared with expected distribution of ρ length in Section 2.2 using χ^2 test. If the observed value is smaller than the expected value this means that it is appropriate to use smaller values of t. As a result, choosing a large value of the number of tables, r, will increase the both time and memory complexities.

DP Coverage Test is similar to Coverage Test which differs in that we calculate the coverage of l bits of keystream after the first DP is encountered. If different IVs have similar keystream prefixes, this will result in low coverage.

Using these tests, we give a relation between randomness properties of cipher and the success of TMTO method. This is the first study in the literature which uses the results of randomness test as clues for attacking a cipher using trade off methods. It is known that the TMTO method is successful if a table has a large coverage. By indexing the first repetition using probability distribution of ρ length in Section 2.2, we try to investigate if the test statistics gives us a result that TMTO is applicable.

3.6 Other Relevant Applications

Meet in the Middle Attack: TMTO methods are also used in a part of a Meet in the Middle Attack (MTM) in several papers. MTM attack is a known plaintext attack which can be summarized as follows: Assume the plaintext is encrypted two times with two separate keys (k_1,k_2)

$$C = E_{k_1}(E_{k_2}(P))$$

the attacker encrypts a plaintext for all possible keys and store the values in memory and decrypts the ciphertext with each key and finding any match between these the stored set are likely to reveal the desired keys.

Firstly, Amirazizi and Hellman suggested to use trade off attack for multiple encryptions [1]. In [24], Demirci and Selçuk developed attacks on 7 and 8 rounds of AES-256 and 7 rounds of AES-192. Since their attack had a huge memory and precomputation complexities, they proposed to use trade off attack to balance complexities between time and memory (also between precomputation and data). Choy et al. proposed several attacks [23] on multiple encryptions such as TMTO-MTM applying the time-memory trade-off attack to the meet-in-the-middle attack on a single ciphertext, Rainbow-MTM applying the Rainbow method, BS-MTM applying on multiple data.

Maiorana-McFarland functions: In [55], Mihaljevic and Imai presented a guess and determine attack in Toyocrypt (See in Appendix A.3). For each guess on 96 bits of the 128-bit LFSR, they formed 32 linear equations. This linear system can be solved to determine the remaining 32 bits in the LFSR. Thus, the effective key length is 96 bits. Using the BS attack, they reduced the attack complexity to 2^{32} with 2^{80} pre-computation and 2^{64} memory. In [39] Khoo et al. generalized the attack and proposed a TMD attack on stream ciphers filter function generators and filter combiners based on Maiorana-McFarland functions. Maiorana-McFarland is a class of Boolean functions with good cryptographic properties [39]. This class can be viewed as constructions based on concatenating linear functions. The Maiorana-McFarland function is defined [39] by the following equation:

$$f(x_0, ..., x_{n-1}) = g(x_0, ..., x_{k-1}) + (x_k, ..., x_{n-1})\phi(x_0, ..., x_{k-1})$$

where $f : GF(2)^n \rightarrow GF(2)$, $g : GF(2)^k \rightarrow GF(2)$ and $\phi : GF(2)^k \rightarrow GF(2)^{n-k}$. ϕ is usually an injection or 2-to-1 map which would require $k < n/2$.

T-functions: T-Functions are proposed as building blocks in cryptographic applications by Klimov and Shamir [41]-[42][43]. The i^{th} bit of the output of T function can depend only on input bits $1 \dots i$. For instance, addition 2^n. In [56], Mitra and Sarkar described a trade off attack when squaring and multiplying is used as T-functions.

CHAPTER 4

COMPUTATIONS OF SUCCESS RATES

4.1 Introduction

In time memory trade offs, the trade off is not in only time and memory but also in success and computation. Kusuda and Matsumoto gave an upper bound for the success rates in terms of m and t in [46]. Kim and Matsumoto showed that the table parameters can be adjusted to achieve higher success probability [40]. In his M.Sc. thesis, Çalık gave an explicit asymptotic formula for the number of distinct points covered by a single Hellman Table. In this chapter, firstly we compare this asymptotic approach with both Kusuda et al. approach and Hellman approach. There is no precise guideline in the literature that points out how to choose parameters for Hellman TMTO. We present a detailed analysis of the success rate of Hellman table via new parameters and also show how to choose parameters to achieve a higher success rate. The results are also experimentally confirmed. Hellman's TMTO Curve is discussed. This chapter is based on our paper [66].

In the next section, we discuss the average length of a chain when we use distinguished points as endpoints. When DP technique is taken into account there are some parameters which effect the success rate of the attack such as the number of distinguished points (D), the expected chains region ($[t_{min}, t_{max}]$), and also number of starting points (m) and number of tables (r). Borst gives a formula for success probability in his unpublished thesis [7]. Standaert et al. claim that they corrected the probability of success for distinguished points [69], however, they have no explicit formula about how to choose parameters to achieve a high success probability. In this section we calculate the average length of a chain, and also the number of left chains after elimination which naturally allows us to derive a formula regarding the overall success rate of the method. We offer some parameters to achieve a high success rate. The empirical results are also supported by theoretical analysis for Rivest's distinguished points.

Then, we study on the coverage of a Rainbow table and show that no matter how the parameters are chosen, coverage remains same. In the last section we compare three technique (Hellman, DP, Rainbow) of TMTO on a stream cipher, A5.

4.2 Hellman Tables

4.2.1 A Survey on Known Computations

Hellman's Approach

The *success probability* is the frequency of success when attack is repeated certain times independently. In time memory trade offs, the probability of success depends on how well the computed chains cover the key space. If E is modeled as a random function mapping the set into itself, and if the key k is chosen uniformly from this set, then the probability of success $S(m,t)$ of a table of size (m,t) [34] is bounded by

$$S(m,t) \geq \sum_{i=1}^{m} \sum_{j=0}^{t-1} [(N-it)/N]^{j+1} . \tag{4.1}$$

Then, the probability of success $S(m,t,r)$ of r tables each of size (m,t) is bounded by

$$S(m,t,r) \geq 1 - (1 - \frac{1}{\mathrm{N}} \sum_{i=1}^{m} \sum_{j=0}^{t-1} (1 - \frac{it}{N})^{j+1})^{r}. \tag{4.2}$$

It is clear that for a large key space, it is infeasible to calculate this success rate using the above formula since it consumes a lot of time.

Kusuda-Matsumuto's Approach

Accepting the assumptions in Hellman's success probability is satisfied and $m \gg 1$, $t \gg 1$, and $mt \ll 2^N$ then the success probability [46, 40] with a single table is bounded by

$$S(m,t) \geq g(u)\frac{mt}{N} \text{ where } u = \frac{mt^2}{N},\ g(u) = \frac{1}{u}\int_0^u \frac{1-e^{-x}}{x}dx.$$

Thus for r tables, $S(m,t,r) = 1-(1-S(m,t))^r \approx 1-\exp(-\frac{rmt}{N}g(u))$.

4.2.2 Detailed Computations

Asymptotic approach

Accepting the assumptions in Hellman's success probability is satisfied and $m < \sqrt{N}$, let $Y(m,t)$ be the coverage of a single table,

$$Y(m,t) = \sqrt{2Nm}.tanh\left(\sqrt{\frac{mt^2}{2N}}\right). \tag{4.3}$$

Refer to Çalık's thesis [21] for details.

The ratio of (overall) coverage to N (of r tables) is

$$S(m,t,r) = 1-(1-\sqrt{\frac{2m}{N}}.tanh(\sqrt{\frac{mt^2}{2N}}))^r.$$

Table 4.1 gives the comparison of three approaches (Asymptotic, Kusuda et al. and Hellman) with the observed values of Kusuda. For $m = t = 2^{30}$, it consumes a lot of time to calculate Hellman probability thus we did not give the results of Hellman's approach for these values.

Table 4.1: Comparison Table for Asymptotic Approach, Observed by Kusuda, Kusuda et al Approach, Hellman's Approach

u	m	t	k	Asymptotic	Tested	Kusuda	Hellman
2^{-2}	2^{30}	2^{29}	2^{31}	0.6172	0.6133	0.6094	-
2^{-1}	2^{29}	2^{30}	2^{31}	0.6032	0.6015	0.5893	-
2^{0}	2^{30}	2^{30}	2^{30}	0.5773	0.5726	0.5506	-
	2^{13}	2^{12}	2^{12}	0.5773	0.5726	0.5507	0.5492
	2^{7}	2^{7}	2^{7}	0.5770	0.5738	0.5547	0.5489
2^{1}	2^{31}	2^{30}	2^{29}	0.5331	0.5276	0.4831	-
	2^{14}	2^{12}	2^{11}	0.5331	0.5277	0.4832	0.4830
2^{2}	2^{30}	2^{31}	2^{29}	0.4664	0.4621	0.3874	-

4.2.3 Specialized Computations

The effectiveness of method depends highly on the choice of parameters. When choosing optimal parameters there are mainly three items which should be taken into account: the availability of memory, the available computing power and also the required success rate.

For a single table, it is possible to find x such that $E(x) = y$ if and only if x appears somewhere in the table. Thus, the number of preimages that can be found in a table is equal to the number of distinct entries in the table *(the coverage of the table)* which will be denoted by $Y(m, t)$.

In general usage, success rate of a single table is defined to be $S(m,t) = \frac{Y(m,t)}{N}$. However, for a random mapping $f : X \to X$ with $|X| = N$, expectedly Ne^{-1} points will have no inverse images (See Section 2.3). Consequently, the success rate will be bounded by $(1 - e^{-1}) \approx 63\%$ if it is related with the probability of finding inverse image of an arbitrary $x \in X$. If we are focused on points $x \in f(x)$, the success rate can take any value in $(0, 1)$. Since it is a outcome of a oneway function we know that the given point has at least one inverse so the expected success rate of the table will be $S(m,t) = \frac{Y(m,t)}{N(1-e^{-1})}$.

Thus the success rate of a table is defined as

$$S(m,t) = \frac{Y(m,t)}{|Im(E)|} \text{ where } |Im(E)| = N(1 - e^{-1}) = N'.$$

An important question is how to choose m and t to have a high table coverage, hence a high success rate.

Hellman suggested that m and t should be chosen to satisfy $mt^2 = N$. We will settle the mathematical background to explain the relation between mt^2 and N(however [14] explains with matrix stopping rule).

From Eq.4.3 we obtain the marginal change (ΔY) in the value of $Y(m, t)$ as

$$\begin{aligned}
\Delta Y &\approx \frac{\partial y}{\partial m}\Delta m + \frac{\partial y}{\partial t}\Delta t \frac{1}{2} \\
&\approx \left(\frac{\sqrt{2N}}{2\sqrt{m}} \tanh\left(\sqrt{\frac{mt^2}{2N}} \right) + \sqrt{2mN}\frac{t}{\sqrt{2N}}\frac{1}{2\sqrt{m}} \sec h^2\left(\sqrt{\frac{mt^2}{2N}} \right) \right) \Delta m \\
&+ \left(\sqrt{2mN}\sqrt{\frac{m}{2N}} \sec h^2\left(\sqrt{\frac{mt^2}{2N}} \right) \right) \Delta t \\
&\approx \left(\frac{\sqrt{N}}{\sqrt{2m}} \tanh\left(\sqrt{\frac{mt^2}{2N}} \right) + \frac{t}{2} \sec h^2\left(\sqrt{\frac{mt^2}{2N}} \right) \right) \Delta m + \left(m \sec h^2\left(\sqrt{\frac{mt^2}{2N}} \right) \right) \Delta t
\end{aligned}$$

Then for a fixed value of m ($\Delta m = 0$) and for $\Delta t = 1$, in other words adding one column to a table will effect the coverage by

$$\Delta Y = m \sec h^2\left(\sqrt{\frac{mt^2}{2N}} \right).$$

For example, let $N = 2^{48}$, $m = 2^{16}$, $t = 2^{19}$ then $\Delta Y = 2^{1,67} < 4$. This means that by a load of $m = 2^{16}$ encryptions, we obtain less than 4 'distinct' points in a table. So the value of m and t should be chosen properly. Now we will try to investigate the 'rules' how to choose the feasible values for m and t.

A natural way of measuring the feasibility of a single table is the ratio of the number of distinct points (outcomes) to the number of work done (encryptions) so we define

$$R(m,t) = \frac{Y(m,t)}{mt}$$

as the *feasibility* of the table. Then

$$R(m,t) = \sqrt{\frac{2N}{mt^2}} tanh\left(\sqrt{\frac{mt^2}{2N}} \right). \tag{4.4}$$

It should be noted that R is indeed a function of 'mt^2'. This means that for any fixed $C \in R$ any choice of m and t with $mt^2 = C$ will yield the same feasibility.

Define $\alpha = \frac{mtr}{N}$ and $\lambda = \frac{mt^2}{N}$. In general, $\alpha = 1$ since precomputation time(P) is equal to mtr which is not expected to exceed N. Then we have

$$R(m,t) = \sqrt{\frac{2}{\lambda}} tanh\left(\sqrt{\frac{\lambda}{2}} \right).$$

Expected number of distinct elements in the entire set of r tables is

$$S(m,t,r) = 1 - \left(1 - \frac{Y(m,t)}{N'}\right)^r \approx 1 - e^{-\frac{Y(m,t)r}{N'}}.$$

Since $r = \frac{\alpha N}{mt}$,

$$S(m,t,r) = 1 - e^{\frac{-Y(m,t)}{mt}\frac{N}{N'}} = 1 - e^{\frac{-R(m,t)\alpha}{\varphi}}$$

where $\varphi = 1 - e^{-1}$.

Figure 4.1 gives the $R(m,t)$ for different values of λ. For $\lambda = 1$, $R(m,t) \approx 0,86$.

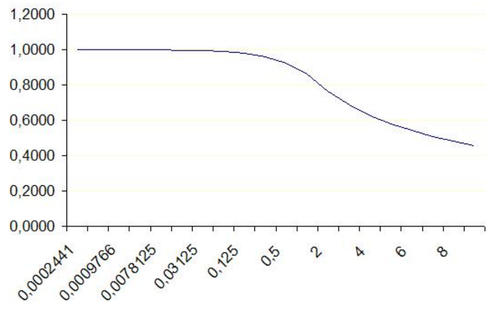

Figure 4.1: Feasibility of a Table

4.2.4 Experimental Results

We give our experimental results on Sha-1 (Refer to Appendix A.5 for details).

1. To compare our analysis with empirical results, we give empirical table coverage, $H(m,t)$, overall coverage $H(m,t,r)$ for reduced Sha-1 with $N = 2^{21}$ in Table 4.2. $\alpha = 1$ in this table, so precomputation time is not exceeding N.

2. Hellman suggested that m and t should be chosen to satisfy $mt^2 = N$ to obtain optimal coverage rate (the row colored in gray in Table 4.3). The optimal coverage of a single table is given as about $0.80mt/N$. Now we demonstrate that with different M/T (memory/ time complexities), it is still possible to have the same success rate. The table is constructed for $N = 2^{40}$, $\lambda = 1$, $\alpha = 1$, resulting in $R(m,t) = 0,86$.

Table 4.2: Comparison for Reduced Sha-1 with $N = 2^{21}$

λ	4	2	1	1/2	1/4
m	128	64	128	64	128
t	256	256	128	128	64
r	64	128	128	256	256
R(m,t)	0,63	0,76	0,86	0,92	0,96
S(m,t)	0,02	0,01	0,01	0,01	0,01
S(m,t,r)	0,63	0,70	0,75	0,77	0,78
Y(m,t)	20459	12364	13979	7494	7733
H(m,t)	20584	12478	14108	7572	7867
Y(m,t,r)	782679	864433	918141	947460	960488
H(m,t,r)	838736	930556	988617	1019724	1036794

Table 4.3: For $N = 2^{40}$ Hellman Construction

m	t	r	S(m,t)	S(m,t,r)	M	Time
2^{20}	2^{10}	2^{10}	0,0013	0,744128319	30	20
2^{18}	2^{11}	2^{11}	0,0007	0,744012227	29	22
2^{16}	2^{12}	2^{12}	0,0003	0,743954199	28	24
2^{14}	2^{13}	2^{13}	0,0002	0,743925191	27	26
$2^{13,4}$	$2^{13,3}$	$2^{13,3}$	0,0001	0,743919745	26,7	26,6
2^{12}	2^{14}	2^{14}	8,31E-05	0,743910687	26	28
2^{10}	2^{15}	2^{15}	4,15E-05	0,743903436	25	30
2^{8}	2^{16}	2^{16}	2,07E-05	0,74389981	24	32
2^{6}	2^{17}	2^{17}	1,03E-05	0,743897998	23	34
2^{4}	2^{18}	2^{18}	5,19E-06	0,743897091	22	36

For instance, if an attacker has less memory access and more computational power, he may try the last row of the above table. Hellman's choice is the case where the memory and time complexities are closest.

3. Since $R(m,t)$ is the single table coverage, if $R(m,t) = 1$ then $\lambda \cong 2^{-12}$, this means that $mt^2 = 2^{-12}N$ (See Table 4.4). Then, $r = \frac{\alpha t}{\lambda}$.

Table 4.4: Effects of r on Success Rates

α	r	$S(m,t,r)$
1	$2^{12}t$	0.7955
2	$2^{13}t$	0.9582
3	$3 * 2^{12}t$	0.9915

It is clear that if m and t is chosen to be too small, it will increase the single table coverage. However, it will increase the number of tables, r, which will cause excessive M and T.

4.2.5 Hellman's Curve

Hellman suggests to construct $N^{1/3}$ tables, each with $m = N^{1/3}$ words, and the number of operations per table is also as $t = N^{1/3}$ [34].

By choosing $\lambda \leq 1$, it is possible to achieve a higher success without increasing memory but additional cost of time complexity. In Table 4.5, $t = 2^{13,3}$, $\alpha = 1$.

Table 4.5: Hellman Construction for Different λ s for $N = 2^{40}$.

m	r	λ	R(m,t)	S(m,t)	S(m,t,r)	M	T
11	10327588	2^{-10}	0,999	1,53E-07	0,794	26,7	36,6
169	645474	2^{-6}	0,997	2,44E-06	0,793	26,7	32,6
676	161369	2^{-4}	0,989	9,70E-06	0,791	26,7	30,6
2702	40342	2^{-2}	0,960	3,76E-05	0,781	26,7	28,6
5405	20171	2^{-1}	0,924	7,24E-05	0,768	26,7	27,6
10809	10086	2^{0}	0,861	0,0001	0,743	26,7	26,6
216189	5043	2^{1}	0,762	0,0002	0,700	26,7	25,6
43238	2521	2^{2}	0,628	0,0003	0,630	26,7	24,6

4.3 DP Success

In [19, 18], Borst et al give some suggestions how to choose parameters (r, m, t) with empirical results. Although Standaert et al give a detailed analysis in [69], the optimal choice of parameters is not given.

4.3.1 A Survey on Known Computations

Borst et al. Approach

Borst gave a formula for success probability in his thesis. The probability to reach a distinguished point in $\leq k$ iterations [18] is given as

$$P(k) \approx 1 - e^{-\frac{k}{2^d}}$$

A reasonable choice for t_{max} is given as 2^{d+3} since $P(2^{d+3}) \approx 1$. Yet we believe that the maximum chain length may not only depend on distinguished property but also depend on the key size.

Before throwing the identical endpoints, average chain length (β_0) is given by:

$$\beta_0 = \frac{1}{2^{-d} + \frac{t}{2^{n+2}}} - \left(\frac{1}{2^{-d} + \frac{t}{2^{n+2}}} + t\right)\left(1 - \left(2^{-d} + \frac{t}{2^{n+2}}\right)^{t}\right)$$

After throwing identical endpoints, only the experimental values for average chain length (β_1) are given. And a relationship between α and β is given as :

$$\beta = \frac{\beta_0 m - \beta_1(m - \alpha)}{\alpha}$$

Borst et al. [19] suggested to choose $d \approx \frac{1}{3}k$, $m \approx 2^{\frac{k}{3}}$ and $t \approx 2^{\frac{k}{3}+3}$ for a success rate of 0.55.

Standaert et al. Approach Assuming the chains have no cycle, the probability to reach a distinguished point in less than $\leq k$ iterations is given as [69]

$$P_s(k) = 1 - \prod_{i=0}^{k-1}\left(1 - \frac{2^{n-d}}{2^n - i}\right).$$

The average chain length β in a region between $[t_{min}, t_{max}]$is given as

$$\beta \simeq \frac{(1-h)^{t_{\min}-2}\left(t_{\min} + \frac{1-h}{h}\right) - (1-h)^{t_{\max}-1}\left(t_{\max} + \frac{1}{h}\right)}{\gamma}$$

where $h = \frac{2^{n-d}}{2^n - \frac{t_{\max}+t_{\min}}{4}}$ and $\gamma = P_s(t_{\max}) - P_s(t_{\min} - 1)$.

The success rate of a single table is given as

$$S(m,t) = \frac{g(\gamma m)}{2^n}$$

where g is the coverage function denoting the number of keys stored after sort and rejection of mergers (x be the number of chains computed in region) defined by

$$g(x) \simeq \left(g(0) - \frac{2^n}{\beta}\right)\left(1 - \frac{\beta^2}{2^n}\right)^j + \frac{2^n}{\beta}$$

4.3.2 Detailed Computations

The probability that a chain which starts with a distinguished point reaches to a distinguished point in exactly k iterations is :

$$P_{DP}(k) = \frac{(N-D)!D}{(N-D-k+1)!N^k}$$

where $k \leq N - D - 1$.

The probability that a chain which starts with a non-distinguished point reaches to a distinguished point in exactly k iterations is :

$$P_{NDP}(k) = \frac{(N-D-1)!D}{(N-D-k)!N^k}$$

where $k \leq N - D$.

Then the probability that a chain reaches to a distinguished point in exactly k iterations is :

$$P(k) = \frac{D}{N}P_{DP}(k) + \frac{N-D}{N}P_{NDP}(k).$$

Using $P(k)$, with some asymptotic approaches, our aim is to give some better approximations for the average chain length (β) and maximum chain length t_{max}.

In this part, we will try to investigate a formula for the average length of a chain (β),

$$\beta = \frac{\sum\limits_{i=0}^{M} kP(k)}{\sum\limits_{i=0}^{M} P(k)} \text{ where } M = N - D.$$

$$\sum_{k=0}^{M} P(k) = \frac{M!D}{N^{M+1}} \sum_{k=0}^{M} \frac{N^k}{k!}. \qquad (4.5)$$

$$\begin{aligned}
\sum_{k=0}^{M} kP(k) &= \frac{M!D}{N^{M+1}} \sum_{k=0}^{M} \frac{N^k(M-k+1)}{k!} \\
&= \frac{M!D}{N^{M+1}} \left[(M+1) \sum_{k=0}^{M} \frac{N^k}{k!} - N \sum_{k=0}^{M-1} \frac{N^k}{k!} \right] \\
&= \frac{M!D}{N^{M+1}} \left[(M+1) \sum_{k=0}^{M} \frac{N^k}{k!} - N \sum_{k=0}^{M} \frac{N^k}{k!} + \frac{N^M}{M!} \right] \\
&= \frac{M!D}{N^{M+1}} \left[(M+1-N) \sum_{k=0}^{M} \frac{N^k}{k!} + D \right] \\
&= \frac{M!D}{N^{M+1}} \left[(1-D) \sum_{k=0}^{M} \frac{N^k}{k!} + D \right] \qquad (4.6)
\end{aligned}$$

It is known that $\sum_{i=0}^{\infty} \frac{N^i}{i!} = e^N$ and $\sum_{i=0}^{N} \frac{N^i}{i!} \approx \frac{e^N}{2}$ then $\sum_{i=0}^{M} \frac{N^i}{i!} \approx \lambda e^N$ where $\lambda \leq \frac{1}{2}$.

Using Eq.4.5 and Eq.4.6, the average chain length will be

$$\beta = 1 - D + \frac{N^{M+1}}{M! \sum_{k=0}^{M} \frac{N^k}{k!}}$$

Using Stirling Formula

$$\beta = 1 - D + \frac{N^{M+1}}{\frac{M^M}{e^M} \sqrt{2\pi M} \lambda e^N}$$

Since $M = N - D$,

$$\beta = 1 - D + \frac{N^{M+1} e^{M-N}}{N^M \left(1 - \frac{D}{N}\right)^M \lambda \sqrt{2\pi M}}$$

$$\beta = 1 - D + \frac{N e^{-D}}{e^{-\frac{DM}{N}} \lambda \sqrt{2\pi M}} = 1 - D + \frac{N}{e^{\frac{D^2}{N}} \lambda \sqrt{2\pi M}} \qquad (4.7)$$

When N is large, it consumes a lot of time to calculate λ. To calculate λ we give a relation between partial exponential summation with incomplete gamma function (See Appendix B for details) as in the following. A relation among N and M may be given as:

$$\frac{\sum_{i=0}^{N} \frac{N^i}{i!}}{e^N} \approx \frac{\int_N^{\infty} t^M e^{-t} dt}{(M)!}$$

To choose the optimal parameters, let our desired success rate be P. Our aim is to find appropriate D such that the probability of success of the trade-off is at least P:

Let k be the smallest cycle length [1], the probability that there is no distinguished point in the smallest cycle is

$$\left(1 - \frac{D}{N}\right)^k$$

Then the probability that at least a DP is encountered in a cycle is

$$1 - \left(1 - \frac{D}{N}\right)^k > P$$

This gives us

$$D > N\left(1 - (1 - P)^{\frac{1}{k}}\right)$$

4.3.3 Specialized Computations

Since calculating the λ in Eq. 4.7 consumes a lot of time, we give another approach in this part to calculate the average length of a chain. Using a similar approach with Standaert et al. [69], we recalculated the success rate of a distinguished point in the following way. Although for a random mapping the combinatoric model is selection without replacement, to bring some ease in computations, we use the selection with replacement model. In fact, due to suitable size of the parameters, this shortcut results in reasonable approximations.

Theorem 4.3.1 *Let $f : \{0,1\}^n \to \{0,1\}^n$ be a one-way function, and DP- property of order d. Then the key space is $N = 2^n$ and the number of distinguished points is $D = 2^{n-d}$. Assuming there is no cycle before k^{th} iteration. Then the probability to reach a distinguished point in exactly k iterations is*

$$P(k) = \frac{D(N-D)!(N-k)!}{N!(N-D-(k-1))!}$$

[1] There exists some efficient algorithms to find cycles in random mappings (See [44]-[20]-[67])

Proof. The probability to reach a DP in 1^{th} iteration

$$\begin{aligned}
P(1) &= \frac{D}{N} \\
P(2) &= \left(1-\frac{D}{N}\right)\frac{D}{N-1} \\
P(3) &= \left(1-\frac{D}{N}\right)\left(1-\frac{D}{N-1}\right)\frac{D}{N-2} \\
&\cdots \\
P(k) &= \left(1-\frac{D}{N}\right)\left(1-\frac{D}{N-1}\right)\cdots\left(1-\frac{D}{N-(k-2)}\right)\frac{D}{N-(k-1)}
\end{aligned}$$

which can be also derived from [69] by $P(k) - P(k-1)$.

$$P(k) = \frac{D}{N-(k-1)}\prod_{i=0}^{k-2}\left(1-\frac{D}{N-i}\right) \quad = \quad \frac{2^{n-d}}{2^n-(k-1)}\prod_{i=0}^{k-2}\left(1-\frac{2^{n-d}}{2^n-i}\right)$$

■

In this part we will try to investigate the average length of a chain, $\beta = \frac{\sum_{i=0}^{M} kP(k)}{\sum_{i=0}^{M} P(k)}$ where $M = N - D$.

The approximation as in [69] gives us

$$\begin{aligned}
P(k) = \frac{D}{N-(k-1)}\prod_{i=0}^{k-2}\left(1-\frac{D}{N-i}\right) \quad &= \quad \frac{2^{n-d}}{2^n-(k-1)}\prod_{i=0}^{k-2}\left(1-\frac{2^{n-d}}{2^n-i}\right) \\
&= \quad 2^{-d}\left(1-2^{-d}\right)^{k-2}
\end{aligned}$$

since $k << 2^n$.

$$\sum_{\mathrm{k\,=\,t_{min}}}^{t_{\max}} P(k) = \left(1-2^{-d}\right)^{t_{\min}-2} - \left(1-2^{-d}\right)^{t_{\max}-1} \qquad (4.8)$$

and also

$$\sum_{\mathrm{k\,=\,t_{min}}}^{t_{\max}} kP(k) = 2^d\left[\left(1-2^{-d}\right)^{t_{\min}-2}(1-2^{-d}+2^{-d}t_{min}) - (1-2^{-d})^{t_{\max}-1}\left(2^{-d}t_{max}\right)\right] \qquad (4.9)$$

Finally equation (4.9) and (4.8) gives us the average chain length.

$$\beta = \frac{2^d\left[\left(1-2^{-d}\right)^{t_{\min}-2}(1-2^{-d}+2^{-d}t_{min})-(1-2^{-d})^{t_{\max}-1}\left(2^{-d}t_{max}\right)\right]}{(1-2^{-d})^{t_{\min}-2}-(1-2^{-d})^{t_{\max}-1}}$$

In this part we will consider how many chains will be left in a table. When a table is constructed we start with m starting points but since we discard chains with the identical endpoints and chains of length that are not in the region $[t_{min}, t_{max}]$, the remaining number of chains will be smaller than m.

The probability to reach a DP in less than t_{min} is

$$P(\leqslant t_{\min}) = 1-\left(1-\frac{2^{n-d}}{2^n-\frac{t_{\min}-1}{2}}\right)^{t_{\min}}.$$

$t_{min} << 2^n$, thus to reach a DP in less than t_{min} iterations in a chain is

$$P(\leqslant t_{\min}) = 1-\left(1-\frac{1}{2^d}\right)^{t_{\min}}.$$

For m chains

$$m\left(1-\left(1-\frac{1}{2^d}\right)^{t_{\min}}\right) = m\left(1-\left[\left(1-\frac{1}{2^d}\right)^{2^d}\right]^{\frac{t_{\min}}{2^d}}\right) \cong m\left(1-\left[e^{-1}\right]^{\frac{t_{\min}}{d}}\right) = m\left(1-\frac{1}{e^{\frac{t_{\min}}{2^d}}}\right).$$

The remaining rows after t_{min} elimination is $T \cong me^{-\frac{t_{min}}{2^d}}$.

There are D distinguished points, so if we pick (with replacement) T_n integers randomly, out of D integers, then the expected number of *distinct* integers in our collection will be

$$T_{n+1} = D\left(1-e^{-\frac{T_n}{D}}\right)$$

where $T_0 = T$.

By discarding identical endpoints, we remove merging chains. By approximating the reccurence relation above as in [3], the expected number of remaining chains will be

$$\alpha \cong \frac{T}{1+\frac{\beta * T}{2D}}$$

Coverage of a table is $\alpha\beta$. Success of r tables will be $1 - (1 - \frac{\alpha\beta}{N})^r = 1 - e^{-\frac{\alpha\beta r}{N}}$

4.3.4 Experimental Results

In order to confirm this analysis, some experiments on reduced Sha-1 ($N = 2^{28}$) are given in Table 4.6 on which we evaluated the influence of the DP-property and key length, k, on average chain length.

Table 4.6: DP Results on Reduced Sha-1

dp	m	$t_{min} - t_{max}$	α_1	$\alpha_{1,2}$	β_0	$\beta_{0,2}$	α_2	$\alpha_{2,2}$	β_1	Coverage
5	2^9	$2^2 - 2^8$	458	452	36	34	457	451	36	16542
		$2^3 - 2^7$	401	399	36	35	400	398	35	14346
	2^{14}	$2^2 - 2^8$	14904	14488	35	34	14459	14048	35	511926
		$2^3 - 2^7$	12814	12760	37	35	12563	12426	37	464222
	2^{18}	$2^2 - 2^8$	238361	231341	35	34	176151	157626	32	5421730
		$2^3 - 2^7$	204937	204158	37	35	162002	142834	36	5233945
6	2^9	$2^3 - 2^9$	464	452	73	70	462	450	73	33750
		$2^4 - 2^8$	396	399	74	73	395	397	74	29416
	2^{12}	$2^3 - 2^9$	3663	3614	71	70	3563	3509	70	251421
		$2^4 - 2^8$	3135	3189	74	72	3077	3105	74	228367
	2^{16}	$2^3 - 2^9$	58639	57835	71	70	43878	39042	65	2696209
		$2^4 - 2^8$	50487	51040	74	72	40257	35433	72	2593928
7	2^9	$2^4 - 2^{10}$	452	452	137	142	445	445	134	59875
		$2^5 - 2^9$	384	398	146	147	381	393	145	55287
	2^{11}	$2^4 - 2^{10}$	1808	1807	138	142	1174	1703	136	230178
		$2^5 - 2^9$	1572	1594	146	147	1507	1510	145	215549
	2^{14}	$2^4 - 2^{10}$	14560	14458	139	142	11049	9716	128	1334881
		$2^5 - 2^9$	12624	12759	146	147	10111	8824	142	1287837
8	2^7	$2^5 - 2^{11}$	114	113	295	285	113	111	297	33207
		$2^6 - 2^{10}$	98	100	326	295	97	98	328	31442
	2^9	$2^5 - 2^{11}$	454	451	286	285	434	426	284	122228
		$2^6 - 2^{10}$	389	398	306	295	375	388	304	112595
	2^{11}	$2^5 - 2^{11}$	1806	1807	282	285	1540	1451	268	394705
		$2^6 - 2^{10}$	1153	1594	294	295	1375	1303	288	370219
9	2^7	$2^6 - 2^{12}$	116	112	630	572	105	106	613	63998
		$2^7 - 2^{11}$	95	99	621	592	90	94	625	53400
	2^9	$2^6 - 2^{12}$	450	451	599	572	375	362	563	207045
		$2^7 - 2^{11}$	382	398	608	592	334	325	605	193000
	2^{11}	$2^6 - 2^{12}$	1819	1807	596	572	1177	910	518	543058
		$2^7 - 2^{11}$	1545	1594	606	592	1099	839	577	529412
10	2^7	$2^7 - 2^{13}$	111	113	1124	1147	84	90	1106	88635
		$2^8 - 2^{12}$	102	100	1205	1186	80	81	1211	88648
	2^9	$2^7 - 2^{13}$	454	452	1123	1147	277	227	993	253215
		$2^8 - 2^{12}$	403	399	1230	1186	259	210	1131	254898
	2^{11}	$2^7 - 2^{13}$	1819	1807	1151	1147	782	365	901	553829
		$2^8 - 2^{12}$	1619	1595	1247	1186	762	346	1087	567180

α_1 is the number of chains that reaches a DP and α_2 is the number of remaining chains after discarding the identical endpoints ($\alpha_{i,2}$ are the calculated number of chains with the above formula), β_0 is the average chain length, β_1 is the average chain length after elimination of identical endpoints.

4.4 Rainbow Tables

4.4.1 A Survey on Known Computations

The success rate of a single Rainbow table is given [61] by

$$P(s) = 1 - \prod_{i=1}^{t} (1 - \frac{m_i}{N})$$

where $m_1 = m$ and $m_{n+1} = N(1 - e^{-\frac{m_n}{N}})$

4.4.2 Detailed Computations

Let b_i is the number of distinct elements in column j. Then starting with $b_0 = m$ points $b_1, b_2, \ldots, b_i$ is expected to be

$$\begin{aligned} b_1 &= N(1 - exp(-\frac{b_0}{N})) \\ b_2 &= N(1 - exp(-\frac{b_1}{N})) \\ &\ldots \\ b_i &= N(1 - exp(-\frac{b_{i-1}}{N})). \end{aligned}$$

Let $\frac{b_k}{N} = 1 - c_k$ then $c_k = e^{-1}e^{c_{k-1}}$ where $c_1 = e^{-m/N}$, $2 \le k \le t$

$$\sum_{k=2}^{t} b_k = N(\sum_{k=2}^{t}(1 - e^{-1}e^{c_{k-1}}) = N(\frac{t(t+1)}{2} - \sum_{k=2}^{t} e^{-1}e^{c_{k-1}})$$

Then the coverage will be $Y(m,t) = N(1 - e^{m + \sum_{k=1}^{t} b_k})$

4.4.3 Specialized Computations

We consider a rainbow table consisting of m_0 rows and t_0 columns where $m_0t_0 = N$. As given in [3], maximum number of remaining chains of length t by starting with m_0 is $m = \frac{2Nm_0}{2N+m_0t}$. The sum of numbers of distinct elements in all rows is

$$Y(m_0, t_0) = \int_0^{t_0} \frac{2Nm_0}{2N + m_0t} dt = 2N \ln(1 + \frac{m_0t_0}{2N}).$$

Recalling that $t_0m_0 = N$,

$$Y \approx 2N \ln\left(\frac{3}{2}\right).$$

This argument shows that no matter how the number of rows and columns are chosen, Y remains the same. The expected number of distinct points in the table is then,

$$Y \cong N'\left(1 - e^{\left(-\frac{Y}{N'}\right)}\right) = N'(1 - (2/3)^{2/\varphi}) = 0,722276N'$$

4.4.4 Experimental Results

In order to confirm this analysis, an experiment on reduced A5/1 ($N = 2^{30}$) is given in Table 4.7 on which we evaluated the influence of the parameters m and t.

Table 4.7: Rainbow Table Coverage for $N = 2^{30}$

m	t	Coverage
2^{20}	2^{10}	595711153
2^{18}	2^{12}	595765307

4.5 Comparing TMTO techniques on A5

To compare these main three techniques, we apply TMTO on A5 for $N = 2^{30}$ (we mean that the first 34 bits of key are set to zero) and we summarize the results in Table 4.8. In DP method, least significant d bits are set to zero and t represents the region $[t_{min}, t_{max}]$. DP method with small d seems to be the best one. Coverages of DP Method is larger than other methods and also false alarm calculations are smaller than other methods, nevertheless, their precomputations are higher.

Table 4.8: Comparing Three Methods on A5

		Hellman	Rainbow		DP		
Parameters	d	-	-	-	8	10	12
	m	2^{10}	2^{18}	2^{20}	2^{11}	2^{11}	2^{9}
	t	2^{10}	2^{12}	2^{10}	$2^{6} - 2^{10}$	$2^{8} - 2^{12}$	$2^{10} - 2^{14}$
	r	2^{10}	-	-	2^{11}	2^{9}	2^{9}
Coverage	single table	902425	-	-	574117	1453724	2743951
	all tables	619827147	595765307	595711153	714643842	537137273	784018295
Precomputation	-	32429927	23578031	29455300	79634860	80616344	173238300
Online	average number of false alarms	520	1213	302	214	245	356
	average number of hash calculations per false alarm	1788813	1707755	105816	40022	152837	743984

CHAPTER 5

CONCLUSION AND FUTURE WORKS

In this thesis, we have studied the Time Memory Trade Off cryptanalysis method. Trade off attacks are widely studied in the literature and used for inverting various cryptosystems. Focusing on the symmetric ciphers, we have studied the trade off attacks by thoroughly analyzing the statistical properties of underlying random mappings. This analysis is expected to motivate the designers in building especially stream ciphers, to assess the resistance against cryptanalytic trade off attacks. There is no consensus in the research community as to whether trade offs constitute successful attacks. As some researchers defend to consider online complexity ignoring the time consumed for precomputation, others say that there is no practical significance if an attack requires overall complexity greater than exhaustive search.

This study can be roughly classified into three parts. The first part includes statistical properties of random mappings and permutations. The properties stated in this part are the key stones in the application of time memory trade off attacks. The second part of the thesis describes the trade off methods in details and gives the extensions and applications of methods on various ciphers. We have also proposed variant constructions techniques in this part. In addition, a unified approach to the analysis of TMTO method and statistical tests based on random mappings have been provided. As the success rates of trade off attacks are of fundamental importance in any comparisons of methods, the third part provides detailed analysis of success rates for three main trade off techniques: Hellman, Rainbow, and DP. In his study, Hellman gave a lower bound for the coverage of a single table. We have compared Hellman and Kusuda et al. approach with the success obtained from an asymptotic approach. We have presented a detailed analysis of the success rate of Hellman table via new parameters and also showed how to choose parameters to achieve a higher success rate. By using these parameters, we can easily adjust time and memory complexities to have a better coverage, for a fixed total complexity. We have examined the selection of parameters to achieve a higher success rates for DP methods. The selection of parameters does not effect the success rate of Rainbow method nevertheless it should not be forgotten that choosing t larger than average cycle length will cause loops and decrease the coverage. In order to illustrate, we have presented applications of our analysis on stream cipher A5/1 and compare the results of different techniques. This comparisons have indicated that our calculations are consistent with the empirical results. The results in this thesis have also been published in the papers [66]-[65]-[71].

Since the researchers in the community are increasingly interested in studying TMTO attacks, we offer some suggestions for future studies as follows:

- Although false alarms are not taken into consideration when online time complexities are calculated, our empirical results suggest that false alarms have considerable differences in the time complexities of all the methods examined in this study. Investigating the false alarms of these methods will be a valuable future work.
- Comparing the methods regarding to the hardware implementations may be another future work.

REFERENCES

[1] H. R. Amirazizi and M. E. Hellman. Time-memory-processor trade-offs. *IEEE Transactions on Information Theory*, 34(3):505–512, 1988.

[2] J. Arney and E. A. Bender. Random mappings with constraints on coalescence and number of origins. *Pacific J. Math.*, 103(2):269–294, 1982.

[3] G. Avoine, P. Junod, and P. Oechslin. Time-memory trade-offs: False alarm detection using checkpoints. In S. Maitra, C. E. V. Madhavan, and R. Venkatesan, editors, *INDOCRYPT*, volume 3797 of *Lecture Notes in Computer Science*, pages 183–196. Springer, 2005.

[4] G. Avoine, P. Junod, and P. Oechslin. Characterization and improvement of time-memory trade-off based on perfect tables. *ACM Trans. Inf. Syst. Secur.*, 11(4):1–22, 2008.

[5] S. Babbage. Improved exhaustive search attacks on stream ciphers. In *ECOS 95 (European Convention on Security and Detection)*, 1995.

[6] S. Babbage. Cryptanalysis of lili-128, 2001.

[7] S. Babbage and M. Dodd. The mickey stream ciphers. pages 191–209, 2008.

[8] E. Barkan, E. Biham, and A. Shamir. Rigorous bounds on cryptanalytic time/memory tradeoffs. In C. Dwork, editor, *CRYPTO*, volume 4117 of *Lecture Notes in Computer Science*, pages 1–21. Springer, 2006.

[9] E. Biham. How to decrypt or even substitute des-encrypted messages in 2^{28} steps. *Information Proceesing Letters*, 84, 2002.

[10] E. Biham and O. Dunkelman. Cryptanalysis of the a5/1 gsm stream cipher. In B. K. Roy and E. Okamoto, editors, *INDOCRYPT*, volume 1977 of *Lecture Notes in Computer Science*, pages 43–51. Springer, 2000.

[11] A. Biryukov. Some thoughts on time-memory-data tradeoffs. Cryptology ePrint Archive, Report 2005/207, 2005.

[12] A. Biryukov. Design of a new stream cipher–lex. pages 48–56, 2008.

[13] A. Biryukov, S. Mukhopadhyay, and P. Sarkar. Improved time-memory trade-offs with multiple data. In B. Preneel and S. E. Tavares, editors, *Selected Areas in Cryptography*, volume 3897 of *Lecture Notes in Computer Science*, pages 110–127. Springer, 2005.

[14] A. Biryukov and A. Shamir. Cryptanalytic time/memory/data tradeoffs for stream ciphers. In T. Okamoto, editor, *ASIACRYPT*, volume 1976 of *Lecture Notes in Computer Science*, pages 1–13. Springer, 2000.

[15] A. Biryukov, A. Shamir, and D. Wagner. Real time cryptanalysis of a5/1 on a pc. In B. Schneier, editor, *FSE*, volume 1978 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2000.

[16] T. E. Bjørstad. Cryptanalysis of grain using time / memory /data tradeoffs, 2008.

[17] J. Black. *Encyclopedia of Cryptography and Security*. Springer-Verlag, 2005.

[18] J. Borst. *Block Ciphers: Design, Analysis and Side-Channel Analysis*. PhD thesis, Katholieke Universiteit Leuven, 2001.

[19] J. Borst, B. Preneel, and J. Vandewalle. On time-memory tradeoff between exhaustive key search and table precomputation. In P. H. N. de With and M. van der Schaar-Mitrea, editors, *Proc. of 19th Symp. on Information Theory in the Benelux*, pages 111–118, Veldhoven (NL), 28-29 1998. Werkgemeenschap Informatie- en Communicatietheorie, Enschede (NL).

[20] R. P. Brent. An improved monte carlo factorization algorithm. *BIT Numerical Mathematics*, 20(2):176–184, 1980.

[21] C. Calik. How to invert one-way functions: Time-memory trade-off method. Master's thesis, METU, 2007.

[22] C. D. Canniere, J. Lano, , and B. Preneel. Comments on the rediscovery of time memory data tradeoffs. ECRYPT Stream Cipher Project, Report 2005/040, 2005.

[23] J. Choy, K. Khoo, and C.-W. Loe. Applying time-memory-data trade-off to meet-in-the-middle attack. In L. Chen, M. D. Ryan, and G. Wang, editors, *ICICS*, volume 5308 of *Lecture Notes in Computer Science*, pages 157–173. Springer, 2008.

[24] H. Demirci and A. A. Selçuk. A meet-in-the-middle attack on 8-round aes. In *FSE '08: Fast Software Encryption*, pages 116–126, Berlin, Heidelberg, 2008. Springer-Verlag.

[25] O. Dunkelman and N. Keller. A new attack on the lex stream cipher. In J. Pieprzyk, editor, *ASIACRYPT*, volume 5350 of *Lecture Notes in Computer Science*, pages 539–556. Springer, 2008.

[26] O. Dunkelman and N. Keller. Treatment of the initial value in time-memory-data trade-off attacks on stream ciphers. *Inf. Process. Lett.*, 107(5):133–137, 2008.

[27] M. Dworkin. Recommendation for block cipher modes of operation. NIST Special Publication 800-38A, 2001.

[28] eSTREAM. the ecrypt stream cipher project, 2004-2008.

[29] A. Fiat and M. Naor. Rigorous time/space trade-offs for inverting functions. *SIAM J. Comput.*, 29(3):790–803, 1999.

[30] P. Flajolet and A. M. Odlyzko. Random mapping statistics. In *EUROCRYPT*, pages 329–354, 1989.

[31] J. D. Golic. Cryptanalysis of alleged a5 stream cipher. In *EUROCRYPT*, pages 239–255, 1997.

[32] T. Good and M. Benaissa. Hardware performance of estream phase-iii stream cipher candidates. SASC 2008, 2008.

[33] B. Harris. Probability distributions related to random mappings. *The Annals of Mathematical Statistics*, 31(4):1045–1062, 1960.

[34] M. E. Hellman. A cryptanalytic time-memory trade-off. *IEEE Transactions on Information Theory*, 26(4):401–406, 1980.

[35] J. Hong, K. C. Jeong, E. Y. Kwon, I.-S. Lee, and D. Ma. Variants of the distinguished point method for cryptanalytic time memory trade-offs. In L. Chen, Y. Mu, and W. Susilo, editors, *ISPEC*, volume 4991 of *Lecture Notes in Computer Science*, pages 131–145. Springer, 2008.

[36] J. Hong and W.-H. Kim. Tmd-tradeoff and state entropy loss considerations of streamcipher mickey. Cryptology ePrint Archive, Report 2005/257, 2005.

[37] J. Hong and P. Sarkar. New applications of time memory data tradeoffs. In B. K. Roy, editor, *ASIACRYPT*, volume 3788 of *Lecture Notes in Computer Science*, pages 353–372. Springer, 2005.

[38] J. Hong and P. Sarkar. Rediscovery of time memory tradeoffs, 2005.

[39] K. Khoo, G. Chew, G. Gong, and H.-K. Lee. Time-memory-data tradeoff attack on stream ciphers based on maiorana-mcfarland functions. http://eprint.iacr.org/2007/242.pdf, 2008.

[40] I.-J. Kim and T. Matsumoto. Achieving higher success probability in time-memory trade-off cryptanalysis without increasing memory size. *IEICE Transactions on Fundamentals*, E82-A(1):123–129, 1999.

[41] A. Klimov and A. Shamir. A new class of invertible mappings. In *Proceedings of CHES 2002*, pages 460–483. LNCS, Springer-Verlag, 2002.

[42] A. Klimov and A. Shamir. Cryptographic applications of t-functions. In *Proceedings of SAC 2003*. LNCS, Springer-Verlag, 2003.

[43] A. Klimov and A. Shamir. New cryptographic primitives based on multiword t-functions. In *Proceedings of FSE 2004*, volume 3017, pages 1–15. LNCS,Springer-Verlag, 2004.

[44] D. E. Knuth. *Art of Computer Programming, Volume 2: Seminumerical Algorithms (3rd Edition)*. Addison-Wesley Professional, November 1997.

[45] V. F. Kolchin. *Random Mappings*. Springer ,-Verlag, 1986.

[46] K. Kusuda and T. Matsumoto. Optimization of time-memory trade-off cryptanalysis and its application to des, feal-32 and skipjack. *IEICE Transactions on Fundamentals*, E79-A(1):35–48, 1996.

[47] D. W. Marc Briceno, Ian Goldberg. A pedagogical implementation of a5/1. 1999.

[48] G. Marsaglia. The marsaglia random number cdrom including the diehard battery of tests of randomness, 1995.

[49] A. M. Martin Hell, Thomas Johansson and W. Meier. A stream cipher proposal: Grain-128. In *IEEE International Symposium on Information Theory*, pages 1614–1618, 2006.

[50] T. J. Martin Hell and W. Meier. The grain family of stream ciphers. pages 179–190, 2008.

[51] C. C. Meltem Sönmez Turan, Ali Doğanaksoy. Statistical analysis of synchronous stream ciphers. SASC06, 2006.

[52] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of applied cryptography*. CRC Press, Boca Raton, Florida, 1996.

[53] N. Mentens, L. Batina, B. Preneel, and I. Verbauwhede. Time-memory trade-off attack on fpga platforms: Unix password cracking. In K. Bertels, J. M. P. Cardoso, and S. Vassiliadis, editors, *ARC*, volume 3985 of *Lecture Notes in Computer Science*, pages 323–334. Springer, 2006.

[54] N. Metropolis and S. Ulam. A property of randomness of an arithmetical function. *The American Mathematical Monthly*, 60(4):252–253, Apr. 1953.

[55] M. Mihaljevic and H. Imai. Cryptanalysis of toyocrypt-hs1 stream cipher. *IEICE Trans.Fundamentals*, E85-A, 2002.

[56] J. Mitra and P. Sarkar. Time-memory trade-off attacks on multiplications and t-functions. In P. J. Lee, editor, *ASIACRYPT*, volume 3329 of *Lecture Notes in Computer Science*, pages 468–482. Springer, 2004.

[57] S. Mukhopadhyay. *A Study on Time/Memory Trade-Off Cryptanalysis*. PhD thesis, Indian Statistical Institute, 2006.

[58] S. Mukhopadhyay and P. Sarkar. Application of lfsrs in time/memory trade-off cryptanalysis. In J. Song, T. Kwon, and M. Yung, editors, *WISA*, volume 3786 of *Lecture Notes in Computer Science*, pages 25–37. Springer, 2005.

[59] NESSIE. New european schemes for signatures, integrity, and encryption, 2000-2003.

[60] NIST. A statistical test suite for random and pseudorandom number generators for cryptographic applications, 2001.

[61] P. Oechslin. Making a faster cryptanalytic time-memory trade-off. In D. Boneh, editor, *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 617–630. Springer, 2003.

[62] J.-J. Quisquater and J.-P. Delescaille. How easy is collision search? application to des (extended summary). In *EUROCRYPT*, pages 429–434, 1989.

[63] J.-J. Quisquater, F.-X. Standaert, G. Rouvroy, J.-P. David, and J.-D. Legat. A cryptanalytic time-memory tradeoff: First fpga implementation. In M. Glesner, P. Zipf, and M. Renovell, editors, *FPL*, volume 2438 of *Lecture Notes in Computer Science*, pages 780–789. Springer, 2002.

[64] M.-J. O. Saarinen. A time-memory tradeoff attack against lili-128. In J. Daemen and V. Rijmen, editors, *FSE*, volume 2365 of *Lecture Notes in Computer Science*, pages 231–236. Springer, 2002.

[65] N. Saran and A. Doğanaksoy. Variant constructions for tmto based on random mapping statistics. In *Information Security and Cryptology Conference*, 2008.

[66] N. Saran and A. Doğanaksoy. Choosing parameters to achieve a higher success rate for hellman time memory trade off attack. In *The Fourth International Conference on Availability, Reliability and Security*, page to appear, 2009.

[67] R. Sedgewick, T. G. Szymanski, and A. C. Yao. The complexity of finding cycles in periodic functions. *SIAM Journal on Computing*, 11(2):376–390, 1982.

[68] L. R. Simpson, E. Dawson, J. D. Golic, and W. Millan. Lili keystream generator. In *SAC '00: Proceedings of the 7th Annual International Workshop on Selected Areas in Cryptography*, pages 248–261, London, UK, 2001. Springer-Verlag.

[69] F.-X. Standaert, G. Rouvroy, J.-J. Quisquater, and J.-D. Legat. A time-memory tradeoff using distinguished points: New analysis & fpga results. In B. S. K. Jr., C. Paar, editors, *CHES*, volume 2523 of *Lecture Notes in Computer Science*, pages 593–609. Springer, 2002.

[70] M. S. Turan. *On Statistical Analysis of Synchronous Stream Ciphers*. PhD thesis, METU, 2008.

[71] M. S. Turan, C. Calik, N. B. Saran, and A. Doğanaksoy. New distinguishers based on random mappings against stream ciphers. In *SETA '08: Proceedings of the 5th international conference on Sequences and Their Applications*, pages 30–41, Berlin, Heidelberg, 2008. Springer-Verlag.

[72] E. Zenner and M. B. . How secure is secure? on message and iv lengths for synchronous stream ciphers. ECRYPT Stream Cipher Project, Report 2005/039, 2005.

APPENDIX A

Description of Related Ciphers

A.1 A5/1

A5/1 is the stream cipher that is used for GSM communications in most of the European countries and the United States. A5/1 is using a 64-bit key and a known 22-bit public key. A GSM conversation is sent as a sequence of frames every 4.6 millisecond and produces frames of length 228 bit each; 114 bits for sending and 114 bits for receiving information.

A5/1 is contains three maximal length linear feedback shift registers (LFSR) with irregular clocking (See FigureA.1[1]).

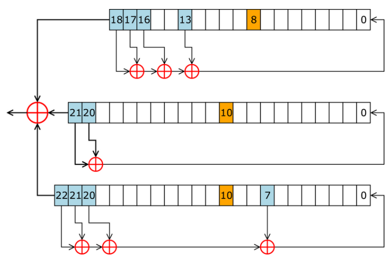

Figure A.1: A5-1 Stream Cipher

The least significant bit (LSB) is labeled as bit zero. The specification of the three LFSRs are :

The majority function of the clocking taps is calculated in each clock and a register is clocked if its clocking taps agree with the majority bits. Two / three registers are clocked at each step, and each register go with probability 3/4 and stop with probability 1/4.

[1] Figure is taken from Wikipedia

Table A.1: A5/1 Specificitaion

	R1	R2	R3
Lengths	19	22	23
Clocking bits	8	10	10
Tap points	13,16,17,18	20,21	7,20,21,22

Key Setup procedure can be summarized as following: Firstly, the registers are set to zero. Then for 64 cycles, the 64-bit secret key is loaded to the least significant bit of each register by XORing. By the same way, the 22-bits of the frame number are loaded. Each register is clocked in each cycle with out majority clocking. Then using the majority clocking mechanism, registers are clocked for 100 cycles and the output discarded.

A.2 LILI

LILI-128 is a stream cipher designed by Dawson, Clark, Golic, Millan, Penna and Simpson [68], and submitted to NESSIE (New European Schemes for Signatures, Integrity, and Encryption[59]).It was accepted as one of six candidate stream ciphers, but was rejected from the second round [64]. The LILI-128 keystream generator is a synchronous stream cipher with a 128 bit key (See FigureA.2[2]).

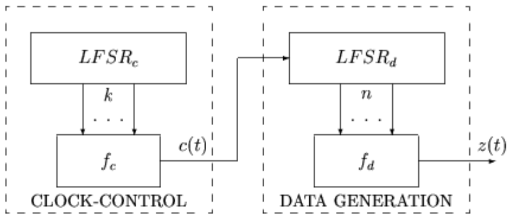

Figure A.2: Lili -128 Stream Cipher

It is a clock-controlled nonlinear filter generator. It uses two binary LFSRs and two functions. $LFSR_d$ is clocked at least once and at most four times depending on $LFSR_c$. The feedback polynomials of $LFSR_c$ and $LFSR_d$ are chosen to be the primitive polynomials as

$$x^{39} + x^{35} + x^{33} + x^{31} + x^{17} + x^{15} + x^{14} + x^{2} + 1$$

[2] Figure is taken from original paper [68]

for $LFSR_c$ and

$$x^{89} + x^{83} + x^{80} + x^{55} + x^{53} + x^{42} + x^{39} + x + 1$$

for $LFSR_d$.

$$f_c(x_{12}, x_{20}) = 2x_{12} + x_{20} + 1$$

is chosen to be a linear clock control function so that the distribution of c(t) is close to uniform. f_d is a highly nonlinear, balanced boolean function that inputs ten bits produced from $LFSR_d$.

Key Setup procedure can be summarized as following: 128 bit key is loaded directly to form the initial values of the two shift registers, from left to right, the first 39 bits in $LFSR_c$ then the remaining 89 bits in $LFSR_d$. LFSRd is clocked at least once and at most four times between the production of consecutive keystream bits.

A.3 Toyocrypt

Toyocrypt is a 128-bit stream cipher proposed to the Japanese government Cryptrec call for cryptographic primitives. It consists 128-bit regularly linear feedback shift register which is filtered by a 128-bit Maiorana-McFarland function. The keystream is generated by applying the nonlinear function f to the contents of 127 of the 128 stages.

A.4 eStream Project

European Network of Excellence for Cryptology(ECRYPT) is a 4-year work funded within the Information Societies Technology (IST) Programme of the European Commission's Sixth Framework Programme (FP6). Ecrypt announced a new project for a public development of efficient, secure stream ciphers (eSTREAM [28])in 2004 and received 34 stream ciphers. ECRYPT was finalized with 7 stream ciphers on September 2008. eStream ciphers that are analyzed with TMTO method are listed below.

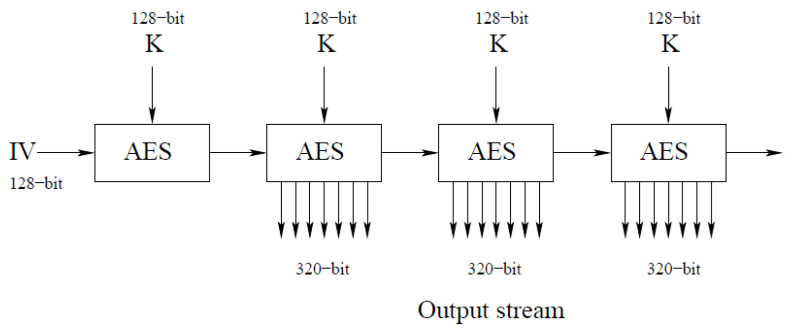

Figure A.3: Lex Stream Cipher

A.4.1 LEX

LEX is stream cipher based on AES (Advanced Encryption Standard) by Biryukov submitted to ECRYPT. The keystream is generated by applying AES in the OFB (Output Feedback Block **??**) mode of operation. In each round, 32 bits of the intermediate state are extracted after the application of each full AES round (See FigureA.3[3]). IV is changed after 500 encryptions and the process is repeated. The secret key is changed after 2^{32} different IVs.

A.4.2 MICKEY

Mickey (Mutual Irregular Clocking KEYstream generator) is a stream cipher proposed to eSTREAM by Babbage and Dodd [7]. Mickey is designed for restricted hardware environments. It supports key size of 80 bit and IV size of between 0 and 80 bit. It uses two 80 bit shift registers with linear feedback (R) and nonlinear feedback (S) each of which has two modes of clocking selected by a control bit (See FigureA.4[4]). Keystream is produced by XORing two bits of shift registers. With a single (K,IV) pair, authors suggested to prodeuce 2^{40} bits keystream.

The xor of two bits, $s_{27} \oplus r_{53}$ for register R and $s_{53} \oplus r_{26}$ for register S, are set as control bits.

[3] Figure is taken from original paper [12]
[4] Figure is taken from original paper [7]

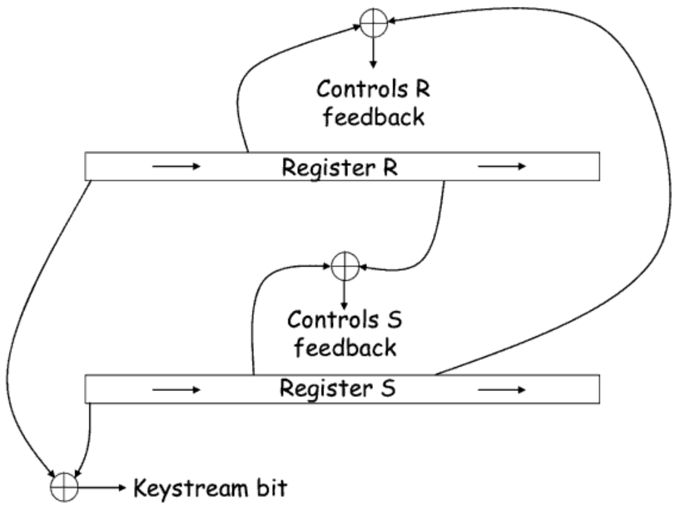

Figure A.4: Mickey Stream Cipher

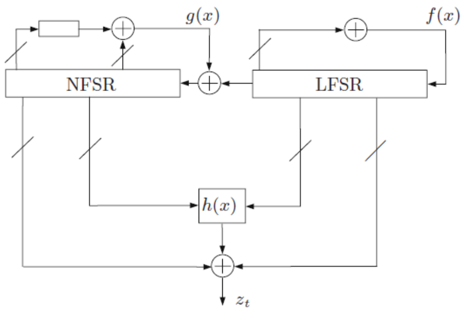

Figure A.5: Grain Stream Cipher

A.4.3 GRAIN

Grain is a stream cipher proposed to eSTREAM by Hell, Johansson and Meier [50](See FigureA.4.3[5]). Grain is designed for restricted hardware environments and has attracted a lot of attention due to its high speed, low gate count and low power consumption [32]. Grain version 1 supportes key size of 80 bits and IV size of 64 bits which is not feasible to exhaustively search with modern computers.

It is based on two shift registers with linear feedback (LFSR) with primitive polynomial ($f(x)$ of degree 80 and with nonlinear feedback (NFSR) with nonlinear boolean function ($g(x)$) of

[5] Figure is taken from original paper [50]

degree 6 given as:

$$f(x) = 1 + x^{18} + x^{29} + x^{42} + x^{57} + x^{67} + x^{80}$$

and

$$\begin{aligned} g(x) = 1 &+ x^{18} + x^{20} + x^{28} + x^{35} + x^{43} + x^{47} + x^{52} + x^{59} + x^{66} + x^{71} + x^{80} + \\ &+ x^{17}x^{20} + x^{43}x^{47} + x^{65}x^{71} + x^{20}x^{28}x^{35} + x^{47}x^{52}x^{59} + x^{17}x^{35}x^{52}x^{71} + \\ &+ x^{20}x^{28}x^{43}x^{47} + x^{17}x^{20}x^{59}x^{65} + x^{17}x^{20}x^{28}x^{35}x^{43} + x^{47}x^{52}x^{59}x^{65}x^{71} + \\ &+ x^{28}x^{35}x^{43}x^{47}x^{52}x^{59} \end{aligned}$$

Let LFSR state at time t is denoted by $l_i, \cdots, l_{i+79}$ and NFSR state at time t is denoted by $n_i, \cdots, n_{i+79}$ Grain outputs a single bit z_i at each clock cycle. It is computed by equation

$$z_i = n_{i+1} + n_{i+2} + n_{i+4} + n_{i+10} + n_{i+31} + n_{i+43} + n_{i+56} + h(l_{i+3}, l_{i+25}, l_{i+46}, l_{i+64}, n_{i+63})$$

where

$$h(x) = x_1 + x_4 + x_0x_3 + x_2x_3 + x_3x_4 + x_0x_1x_2 + x_0x_2x_3 + x_0x_2x_4 + x_1x_2x_4 + x_2x_3x_4$$

Key Setup procedure can be summarized ad following: NLFSR is loaded by 80 key bits and LFSR is loaded by 66 IV bits, remaining 16 bits of LFSR are all set to 1.

Although TMD method (See in Section 3.4.5) does not lead to practical attacks on Grain due to the cost of the precomputation. Grain-128 [49] is designed to meet this requirement, it supports key size of 128 bits and supports IV size of 96 bits.

A.5 SHA

Secure Hash Algorithm (SHA) published by NIST (National Institute of Standards and Technology) as a U.S. government standard. SHA is a hash function family consisting of five algorithms; SHA- 1, SHA-224, SHA-256, SHA-384, and SHA-512. SHA-1 is widely used in various applications such as TLS and SSL, PGP, SSH, S/MIME, and IPsec.

APPENDIX B

Relation with Incomplete Euler Gamma Function

Define

$$F(N) = \sum_{k=0}^{M} \frac{N^k}{k!}$$

$$F'(N) = \sum_{k=0}^{M} \frac{N^k}{k!} - \frac{N^M}{M!}$$

Then

$$F'(N) - F(N) = -\frac{N^M}{M!}$$

$$(e^{-N}F)' = -\frac{e^{-N}N^M}{M!}$$

$$e^{-N}F(N) = 1 - \frac{1}{M!}\int_0^N t^M e^{-t} dt$$

where $F(0) = 1$

Since $F(N) = \lambda e^N$ Then

$$\lambda = 1 - \frac{1}{M!}[\int_0^\infty t^M e^{-t} dt - \int_N^\infty t^M e^{-t} dt]$$

And also for positive integer, M
$\Gamma(M) = \int_0^\infty t^{M-1} e^{-t} dt = (M-1)!$ then

$$\int_0^\infty t^M e^{-t} dt \cong M!$$

This gives us a relation among N and $M = N - D$

$$\frac{\sum\limits_{i=0}^{N} \frac{N^i}{i!}}{e^N} \approx \frac{\int\limits_{N}^{\infty} t^M e^{-t} dt}{(M)!}$$

APPENDIX C

Statistical Randomness Test

C.1 Pearson's Chi Square test

The Pearson's Chi Square (χ^2) Test was first proposed by Karl Pearson. If we have frequency distributions of two or more categories on a variable, Pearson's χ^2 test can be used. It tests how well a theoretical distribution fits into an observed frequency distribution. The disadvantage of the test is the value of the chi-square test statistic depend on how the data putted into classes and also it requires a sufficient size of sample.

Firstly, data is grouped into classes and then compared observed frequency to the theoretical distributions under the null distribution. The test statistic is calculated by

$$\chi^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i}$$

where O_i is the observed frequency, E_i is the theoretical frequency for class i.

The chi-square statistic can then be used to calculate a p-value by comparing the value of the statistic to a chi-square distribution (See [70] for details).

VITA

PERSONAL INFORMATION

Name: Ayşe Nurdan Saran

Date of Birth: 07/03/1976

Marital Status: Married

Phone: 2844500-4012

email: buz@cankaya.edu.tr

EDUCATION

Degree	**Institution**	**Year of Graduation**
MS	Cankaya University, Dep. of Computer Engineering	2002
BS	Ankara University, Dep. of Mathematics	1999

EXPERIENCE

Place	**Enrollment**	**Year**
Cankaya University	Research Asistant	2000-2004
Cankaya University	Instructor	2004-

PUBLICATIONS

- Sönmez Turan M., Çalık Ç., Saran N., Doğanaksoy A., ”New Distinguishers Based on Random Mappings against Stream Ciphers”, Sequences and Their Applications - SETA 2008, Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 4203(2008), 30-41.

- Saran M., Saran N., Yıldırım Z., ”The Use Of Audio Playback Rate Control Tool in Multimedia Learning: A Case Study”, Association for the Advancement of Computing in Education- (AACE) E-Learn 2008, Las Vegas, NV, ABD, November 17-21, 2008.

- Saran N., Doğanaksoy A., ”Choosing parameters for time-memory trade offs via new parameters”, WMC 2008, Second Workshop on Mathematical Cryptology, Santander, Spain, October 23-25, 2008. (Poster)

- Saran N., Doğanaksoy A., ”Choosing Parameters to Achieve A Higher Success Rate for Hellman Time Memory Trade Off Attack”, The Fourth International Conference on Availability, Reliability and Security , ARES 2009,IEEE Computer Society, March, 16 - 19 2009 , Japan.

- Saran N., Doğanaksoy A., ”Variant Constructions for TMTO based on Random Mapping Statistics”, Information Security and Cryptology Conference, ISC 2008, December 25-27, Ankara.