TIME MEMORY TRADEOFF ATTACK ON DATA ENCRYPTION STANDARD (DES)

Project report submitted in fulfilment of the requirement for the degree of

Bachelor of Technology
In
Computer Science and Engineering
By

Anmol Mahajan (151222)

Under the supervision of

Dr. Suman Saha

to

Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology Waknaghat, Solan-173234,

Himachal Pradesh

Certificate

Candidate's Declaration

I hereby declare that the work presented in this report entitled TIME MEMORY TRADEOFF ATTACK ON DATA ENCRYPTION STANDARD(DES) in fulfilment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering/Information Technology submitted in the department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology, Waknaghat is an authentic record of my own work carried out over a period from August 2018 to December 2018 under the supervision of Dr. Suman Saha, Assistant Professor (Senior Grade), Computer Science and Engineering/Information Technology.

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Anmol Mahajan, 151222

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

Dr. Suman Saha
Assistant Professor (Senior Grade)

Computer Science and Engineering / Information Technology

Dated: 1/12/2018

ACKNOWLEDGEMENT

I have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. I would like to extend our sincere thanks to all of them.

I am highly indebted to Dr. Suman Saha for their guidance and constant supervision as well as for providing necessary information regarding the project and also for their support in completing the project.

I would like to express our gratitude towards our parents and Jaypee University of Information Technology for their kind cooperation and encouragement which helped us in completion of this project.

Mine thanks and appreciations also go to our colleague in developing the project and people who have willingly helped us out with their abilities.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Cryptography plays a very important role in a wide range of areas from preserving our privacy on the online world to providing military a secure methodology to undertake their operations. In the vast history of this field the art of encoding or encrypting the messages has held the utmost importance. We have come so far. Ahead of using the secret hieroglyphics to the digitally advanced automated computational phase.

Today digital security is taken for granted. But there are also some more aspects that need to be considered. These algorithms always tend to have some inherent sense of imperfection in them which can be exploited in one way or other to break the system. This could risk the whole system that we have built and can have some serious consequences.

The art and science of finding such faults is called crytanalysis.DES is also one such encryption algorithm that had its share of inbuilt faults and many cryptanalytic methods have been built to break it. Some of these include Time Memory Data Trade-Off Attack, Differential Analysis, Linear Analysis and Differential-Linear Attack. Today, breaking DES is a task of only about a few hours but the cost of the required hardware is very high. In this project we are going to cryptanalyse the DES algorithm using the famous Time Memory Trade-off Attack.

1. Chapter 1

1. Introduction

Humans the most civilized and advanced species on the planet has progressed due to his knowledge and the ability to successfully pass on this knowledge to his future generations. Communication was the primary means of achieving the same. Initially humans exchanged information using gestures. Slowly, he started carving information into objects in the form of images, sculpures etc. to remember it for a longer term. Speech was a golden gift endowed to humans.

But with time the need to protect the information, from leaking to the unwanted sources also emerged. The science corresponding to adding the confusion and diffusion to the message has also developed a lot and is now called cryptography. Cryptography is actually an old science which has seen about 4000 years of development. The credits to the origin of the word “Cryptography” go to the Greeks. The word means “secret writing”. In the earliest times the use of this science was mainly limited to the military purposes. The earliest example or proof that exists today is from 2000 B.C. found in Egypt. This is the ancient hieroglyphic which was considered as the sacred writing since people were unable to decode the images and symbols embedded on it. Later a different tablet was found on which the meaning of these symbols was mentioned against their native language. Turned out that this was a simple substitution cipher. The military has used cryptography since a long time. It has been proposed that the brave army of Spartans also used this art to communicate messages to their generals. They shaved their slave’s head, wrote the message on the bald head and waited for his hair to regrow. Then they sent the slave to the required place, where receiver shaved his head and recovered the message. The Romans are also posited to know something about the art of cryptography. King Caesar himself is credited with the invention of the famous Caesar cipher. This is a simple substitution cipher which involves the shifting to the plaintext by the number mentioned in the key. Hence, it can be said that the substitution ciphers ruled the classical age of cryptography.

As military used this science to exchange important messages the opponents wanted to break the cryptographic systems and gain the hidden information. This lead to the development of something called cryptanalysis. Cryptanalysis can be defined as the analysis of the existing cryptographic systems and using the knowledge gained to break the system with the gain of access to the secret information. The Arabs are credited with the invention of cryptanalysis. The most prominent method being the one developed by the famous mathematician Al-Kindi as earlier as in 820 AD. He was the first to analyse the monoalphabetic substitution ciphers and invented the frequency based analysis. This advancement turned out to be most important till Second World War. As most of the existing systems were just substitution ciphers most of them were susceptible to the above attack. So need to develop more advanced cryptosystems emerged. Such system did not release until mid 1400's.A polyalpahbetic cipher was created by Leon Arberti. He is often referred as "The father of western cryptography". His method involved the use of a cipher disk which consisted of a movable inner disk. With movement this disk could map the inner text to any required plaintext. Until now the use of a secret key was obscure and unknown. In the mid 1500's Blaise De Vigenere came up with an idea that would revolutionize the way cryptography was done. His scheme involved repeating the key to adjust to the length of the plaintext and performing simple addition modulo 26.The simplicity of the method along with the confusion and diffusion it offered was remarkable. Vigenere Cipher was secure late until 1800's.In 1865 Friedrich Kasiski developed a method called KASISKI TEST through which the Vigenere cipher was broken .The cryptography remained approximately the same till the twentieth century.

But by the advent of the industrial revolution and the technological era, the amount of information exchanged experienced an exponential growth. Hence, an urgency in raising the level of the technology involved in the exchange of messages. Moreover the World Wars played an important role in the advancements in this field. In 1917 the British Army intercepted a message intercepted a message sent by the German Army to the Mexican Government. The encrypted message was soon decrypted by the British intelligence. In the message the German Foreign Minister Arthur Zimmerman addressed the Mexican ministers ,pleading them to join the war against the United States Of America. This instigated the American leaders and lead them to participate in the World War. Now the Americans were aware of the importance of maintaining secrecy and developed a better cryptosystem, One Time Pad. One Time Pad consisted of generation of a random key with length equal to the length of the plaintext and XORing the two to get the required ciphertext. This scheme was proposed by Gilbert Sandford Vernam in 1917 only. Similar things also happened in the second world war. The Japanese had invented a machine called PURPLE and encrypted their messages using this machine only.USA had developed the decryption algorithm. Unaware of this the Japanese messaged their fleet posted near USA about the arrival of their military chief. The American army seized this opportunity and killed a very important ,passionate leader. But the enigma machine saga is the most popular and heroic.In 1932 German engineer Arthur Scherbius developed a machine called Enigma which was capable of performing various encryption operations. Machine turned out to be quite handy in the world war for communicating messages, secretly. The British were able to intercept many messages but were unable to break these messages. So, they employed a team of great cryptographers and mathematicians consisting of greats like Alan Turing to break the codes. Eventually, made many observations and figured out many weak points. They found out that a letter in plaintext will never map to the same alphabet in the ciphertext. Using this and many similar facts the British were able to break the enigma machine and gathered massive amount of information about the actions of the German army, which they used to eventually defeat the German propaganda.

Since the world war the science of cryptography has improved exponentially. Computers have become part and parcels of our life. We all have a life online and it has become very important to secure it against any potential threats. Hence, many popular cipher schemes have been developed. This includes many cryptosystems. The old block ciphers, stream ciphers to the more advanced elliptical curve cryptography, lattice based cryptography and even quantum cryptography. All of these provide us with a secure online life. The algorithms like DES (Data Encryption Standard) were used initially to encrypt the data online. But the processing power and the limitless resources are the challenges it was unable to keep up to. Having a key size of only 56 bits until 1990's the scheme was broken using brute force and differential analysis. Due to the importance that these algorithms hold, we require them to be impervious against any kind of attack. Hence, we analyse these algorithms against all possible threats.

One such common threat is the brute force attack. This involves plugging in every possible key and looking for the right key. But due the large size of the key involved our processing power turns out to be feeble. So, cryptanalysts always wanted to develop a better method to brute force. A method more efficient and something that could be used universally to crack any kind of scheme. In 1980 Martin E. Hellman proposed something that seemed like the answer. A simple proposal claiming to establishing a middle way between the memory used and the processing power. A trade of between the time and memory. His solution claimed to reduce the time of brute force from N to $N^{1/3}$. The proposal was eventually accepted and is a famous method to cryptanalyze any algorithm or cryptographic scheme. Cryptanalysis is an important part of cryptology. We need our solutions to be perfect and want to be the first to discover any fault in them, if it does. What will happen if a person having evil intents finds out some faults or loopholes in our cryptographic scheme? Financial crisis, inconsistent information, leakage of information related to national security.DES has a similar history. DES was implemented as the security algorithm in every system from 1970's.The Hellman proposal though practical took a long period to successfully challenge DES and force it to be replaced. The DES served as the encryption standard late until 1990's when NIST finally called for replacing it.

Types of Attacks on DES

1. **Brute Force Attack**

   This is the most basic type of attack on any kind of cipher in which we try every possible key in turn. Number of possible keys are determined by the length of the key and this also determines how feasible this approach will be. Even before DES was adopted as a standard, a lot of questions were raised regarding adequacy of its key size because its key size was too small. This showed the need for a replacement algorithm.

2. **Differential Cryptanalysis**

   Earlier known to IBM and NSA and kept secret, Differential Cryptanalysis was rediscovered in 20^{th} century, 1980s by Eli Biham and Adi Shamir. Chosen plaintexts are needed to exploit all 16 different rounds by Differential Cryptanalysis. Data Encryption Standard wa made resistant towards Differential Cryptanalysis.

3. **Linear Cryptanalysis**

   This attack was discovered by Mitsuru Matsui and required 2^43 known plaintexts. This was the first ever Linear Cryptanalysis on DES. In 1994, Multiple Linear Cryptanalysis was proposed and was further refined by Biryukov. In 2000, a chosen plaintext variant was also introduced of linear cryptanalysis with reduced data complexity.

4. **Improved Davies' Attack**

   It was a special technique suggested by Donald Davies which was further improves by Biham and Biryukov. The most overwhelming kind of DA needs 2^50 plaintexts and has a success probability of 0.51.

1.2 Problem Statement

Data Encryption Standard served as the encryption standard for nearly 30 years. Explain how this algorithm was finally broken and demonstrate a Time Memory Data Trade Off Attack on it.

1.3 Objectives

GOAL: To perform Time Memory Tradeoff Attack on DES with high success probability.

METRICS:

Time taken by the system to perform the attack.

The success probability involved.

The amount of memory required.

Objective 1:

Implement and optimize the implementation of Data Encryption Standard.

Deadline: October 2018

Expected Speed: 2^{20} encryptions per second on normal machine (Intel i5 2.7 Ghz clock speed).

Objective 2:

Implementation of the offline phase of Time Memory Trade Off on Data Encryption Standard.

Deadline: December 2018

Memory Required: 34357938369*4bytes [140 GB for a success probability of about .86]

Objective 3:

Extend the implementation using rainbow and perfect tables.

Deadline: February 2019

Objective 4:

Implementation of the online phase of Time Memory Trade Off on Data Encryption Standard.

Deadline: April 2018

Expected Time: O(log(m)+t)

1.4 **Methodology**

In the final step of our project we will create a Graphical application and merge it with the attack's implementation. The visuals will contain a button for preparing a table for a new plaintext or for performing the online phase. After performing the attack the key will be displayed. High processing power is required for our system. Hence, we will be needing GPU's which will be from some cloud online services.

2. Chapter-2 LITERATURE SURVEY

2.1 A Cryptographic Time Memory Trade Off

2.1.1 Author: Martin E. Hellman

2.1.2 Year: 1980

2.1.3 Summary:

This was the first paper to introduce the idea of finding a middle way between the amount of memory used and the time required to brute force the entire scheme. This paper proposed to break down the task of brute forcing into two parts that is firstly the pre computation step to make a table offline and then the online phase to search for the key actually used.

2.1.4 Advantages:

2.1.4.1 First paper to introduce the concept of using Time Memory Trade Off.

2.1.4.2 Reduced the brute force time by a significant amount.

2.1.5 Disadvantages:

2.1.5.1 False Alarms in the online phase.

2.1.5.2 Merged chains and loops created problems and increased the amount of total memory required.

2.2 Rigorous time / space tradeoffs for inverting functions

2.2.1 Author: Amos Fiat, MoniNaor

2.2.2 Year: 1991

2.2.3 Summary: The initial proposal of Hellman claimed that TMTO could be extended and used for inverting one way functions also. But TMTO remained limited upto block ciphers only and its use for inversion was realised in this paper, which proposed an effective methodology for achieving this.

2.2.4 Advantages:

2.2.4.1 First paper to extend the concept of using Time Memory Trade Off upto inversion of the one way functions.

2.24.2 Reduced the brute force time by a significant amount.

2.2.5 Disadvantages:

2.2.5.1 The proposal was just a conceptual one and they were unable to prove it with practical results.

2.3Cryptanalytic Time / Memory / Data Tradeoffs for Stream Ciphers

2.3.1 Author: Alex Biryukov and Adi Shamir

2.3.2 Year: 2000

2.3.3 Summary:

This was the first paper to propose a method for implementing TMTO successfully on stream ciphers. The idea was to use low sampling resistance for tradeoff attacks on stream ciphers.

2.3.4 Advantages:

2.3.4.1 First paper to introduce the concept of using Time Memory Trade Off on the stream ciphers.

2.3.4.2 The time for the pre computation phase decreased. Moreover, the total need for the data decreased.

2.4 A Time-Memory Trade off by use of Distinguished Points

2.4.1 Authors: Francois-Xavier Standaert, GaelRouvroy, Jean-Jaques Quisquater

2.4.2 Year: 2002

2.4.3 Summary:

The first paper to implement the proposal of Rivest of using distinguished points. In this method rather than generating the chains of same length, we fix the end points of chain and keep on generating the chain until any of these endpoint is achieved.

2.4.4 Advantages:

2.4.4.1 First paper to extend the concept of using distinguished points for performing
. Time Memory Data Trade Off.

2.4.4.2 The algorithm gave us better parameters and reduced the total memory references and made the implementation easier.

2.5 Breaking Ciphers with COPACOBANA-A Cost-Optimized Parallel Code Breaker

2.5.1 Authors: SandeepKumar, ChristofPaar, JanPelzl, Gerd Pfeiffer

2.5.2 Year: 2003

2.5.3 Summary: This paper was the first to propose the design of a specialised machine called COPACOBANA to break DES. The machine was capable of doing so in about 2 days. The estimated cost of the machine was about $10,000 which was considered more than affordable.

2.5.4 Advantages:

2.5.4.1 First paper to propose an adequate design for a specialised machine.

2.5.4.2 The designed turned out to be very efficient and cost effective.

2.6 Preparing a Faster Cryptanalytic Time-Memory Trade-Off

2.6.1 Author: Philippe Oechslin

2.6.2 Year: 2003

2.6.3 Summary: Introduced the concept of rainbow table. The rainbow table is a variant of the time memory trade off method which deals with the various problems its ancestor had by using different reduction function at each step in the chain. Philippe Oechslin showed that performance of the TMTO can be increased drastically using different reduction function.

The probability of collision between two chains decreased to 1/t. Philippe Oechslin further extended his work and worked to find the optimum values of the parameters m, t and l and worked to create something called perfect tables. Perfect tables have 0 merges and no memory is wasted.

2.6.4 Advantages:

2.6.4.1 First paper to propose the idea of rainbow tables. Perfect table was also introduced in this paper only.

2.6.4.2 Gave proper procedure to choose the various parameters involved in performing Time Memory Data Trade off.

2.6.4.3 Tackled the problems of false alarms, merges and loops originally prominent in the Hellman paper.

2.7 Time Memory Tradeoff Implementation on Copacobana

2.7.1 Author: Stefan Spitz

2.7.2 Year: 2007

2.7.3 Summary: Major objective of this research work was to provide a systematic method which can lower the time required to break the cipher and extract the original key from the block cipher. Its implementation was on the popular Data Encyption Standard (DES). Stefan used specifically designed hardware Copacobana to attack on the DES in the least time possible.

2.7.4 Advantages:

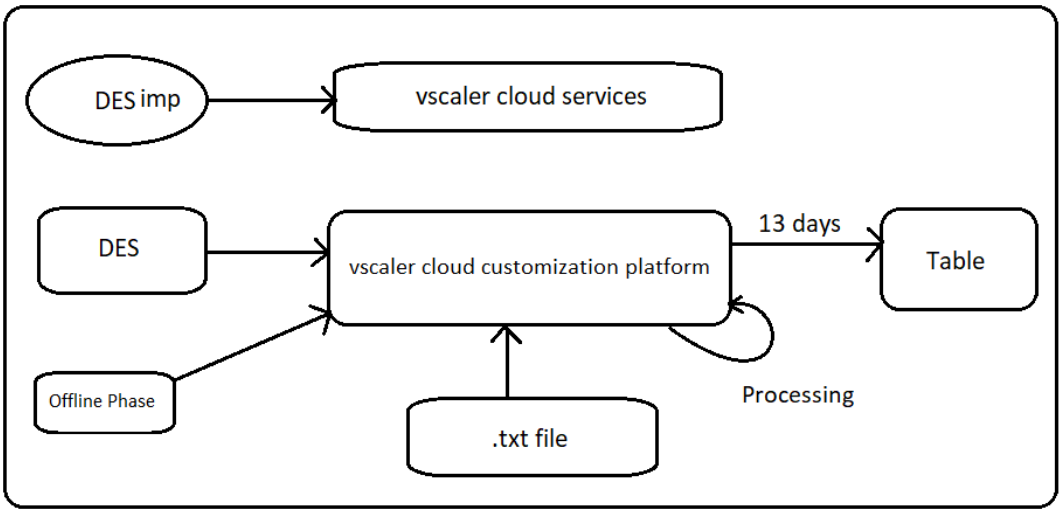
2.7.4.1 This particular utilization of only one task gives the odds to make this progressively difficult procedure an easier one and expends less time of breaking the DES.

2.7.4.2 Provided the predictability to improvise the structure of some specific tasks which can be included using FPGA's.

3. Chapter-3 SYSTEM DEVELOPMENT

3.1. Computational

The computational model will consist of the use of cloud computing services.The algorithm will be uploaded to the cloud where high performance hardware will run our algorithm till the creation of the tables.



3.1.1

3.2. Mathematical

Hellman Analysis

The equation for each step :

$C=S_k(P)$

The output generated in this step will be of 64 bits. So we reduce it to the key size i.e 56 bits.

$f(K)=R[S_k(P)]$

The probability of success is given by the formula:

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

The approximate count of false alarms per table is given as:

$E(F)<=mt(t+1) / 2*N$

Hellman suggested to follow the relation:

$M=m*t$

$mt^{2}=N$

$m=t=N^{(1/3)}$

Breaking the DES via this method was reduced to 2^38.

m - number of starting points

t - length of chain

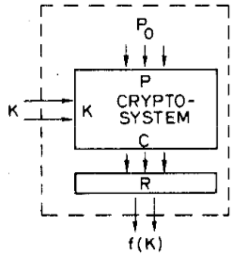


Fig. 1. Construction of the function *f*.

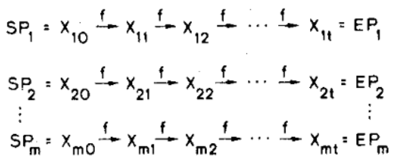


Fig. 2. Matrix of images under *f*.

3.2.1

Alterations in DES:

DES (Data Encryption Standard) is one of the initial encryption algorithms that involved 64 bits of plain text being encrypted using key of 56 bits. DES involves total of 16 rounds with partition of plaintext into two equal halves named left and right part respectively. Similarly, key is also divided, permutation and combination boxes are used and finally halves are XORed and stored respectively for the next round.

I have implemented DES in C++14 and to improve its speed, I have stored S- boxes, Permutation boxes and Combination boxes, instead of calculating them every time. Such an implementation has improved the speed of our DES algorithm to 2^20 encryptions.

Sorting Function:

In my approach of implementation of Time Memory Trade Off Attack, I used the already present inbuilt sorting function of C++14.The inbuilt C++14 sorting function is comprised of an introspective sorting which is the overall mix of three types of sorting namely quick, heap and insertion sort. Such sorting is formulated in such a way that it overall tries to perform in a better way, covering all kinds of cases, that is ranging from best case to the worst possible case.

Insertion sort If elements in array or list are less than or equal to 16, Insertion Sorting is used

Heap sort performs its basic functions such as heapify in the algorithm

Quick sort is mainly used to find the pivot element, thatis the point from where partition will happen

Therefore, Introsort is an optimized combination of all the above mentioned sorting algorithms for dealing with average complexities provided as solution in best and worst possible cases.

Algorithm:

Sort(A, A+size)

```
depLimit = 2*floor(log(len(A)))
insort(A, depLimit)

insort(A, depLimit):
b = len(A)
if b<=16:
insertionSort(A)
if (depLimit == 0):
heapsort(A)
else:

// using quick sort, the
// partition point is found
c = partition(A)
insort(A[0:c-1], depLimit - 1)
insort(A[c+1:b], depLimit - 1)
```

Why binary searching algorithm is preferred over other searching algorihms?

Binary search:

Let us assume that we are provided with a array which is already in a sorted fashion comprising of n values in total and we are looking for a particular value in this array. Therefore, in binary searching, we divide the provided list of elements in two partitions, and look whether the middle values is bigger or smaller than the value we are looking for. If target value is large then look in the right portion of the list and if target value is lesser then look in the left portion of the list. Repeat the above procedure until we find the required target value in the list.

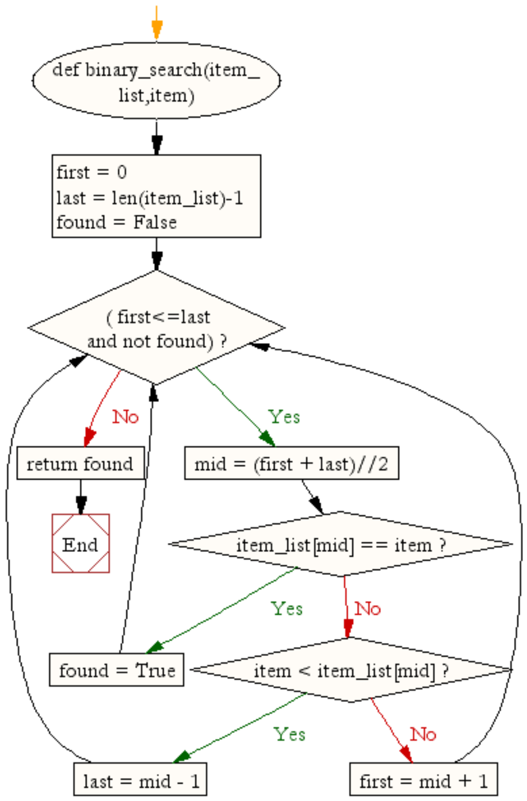


Figure 3.2.1 (a)

Why we preferred binary searching algorithm?

The main reason behind using binary search algorithm is its overall lower complexity in comparison with linear searching algorithms. The overall time complexity of binary search is O(log(n)) and for linear searching it is O(n*n). In best case situations, binary search provides the results in O(1). Moreover, binary search uses no extra space for finding the target value in the list.

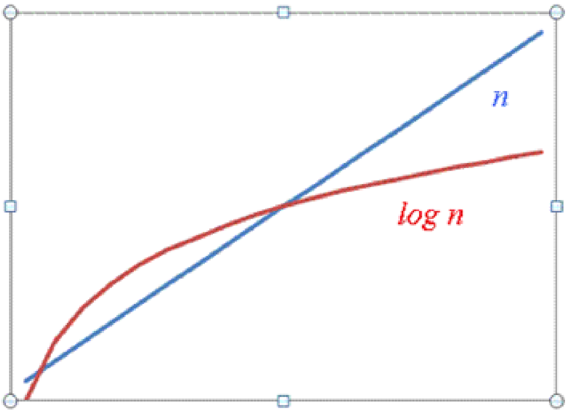


Figure 3.2.1 (b)

Why Hashing Algorithms were avoided for this purpose?

We know that hashing algorithms are fast as there have overall linear time complexity. But there are two parameters due to which we cannot use Hashing in Time Memory Trade Off Attack, that is large number of key value pairs stored and the manner in which these chains are stored for dealing with memory and time constraints. During the execution of Time Memory Trade Off Attack, we keep track of millions or billions of encrypted texts in pre-calculation part which overall leads to a lot of values to be stored and kept track of. If we include hashing in TMTO then we again need to keep track of hashvalues of billions of calculations which is not at all feasible. Such a procedure will surely hike up the space complexity to a whole new stage and keeping track of such a large number of values only for looking for one encrypted key is not at all a good practical approach.

Why we preferred iteration over recursion in our binary searching approach?

Although iterative as well as recursive binary search have similar time complexity (O(log(n))) but there is major difference in other parameters such a overall length of written code, overall complexity of the written code and of course, space complexity. The major difference between space complexities of iterative and recursive methods is that the space complexity of iterative solution is O(1) but in case of recursive implementation it can go up to O(n).

In Time Memory Trade Off Attack, we need to evenly go for both time as well as space complexities for our calculations. Therefore, it is advantageous to implement iterative binary searching algorithm and prefer it over recursive binary searching algorithm.

Expansion and reduction function in implementation of rainbow tables:

Now, there was a major problem with the solution proposed by Martin Hellman on TMTO that there was always same reduction function being used for at each point for storing values of encrypted texts in chains. In Implementation of TMTO over DES, we were using the reduction function as ignoring last 8 bits of 64 bits output of DES and storing only 56 bits. But, using same reduction function over and over again at each point in all chains can lead to different problems in real world scenario such as loops, circles ad merging of chains.

In rainbow tables, we use reduction function as an iterative function on the outcome of DES to reduce the number of sample output bits from 64 bits to 56 bits and passing these 56 bits as the key for net DES function in chain, keeping the plain text same all the time.
For implementation of TMTO on other encryption algorithms such as triple DES, we can use an expansion function in same manner.

In expansion function, we use an iterative function on the outcome of 3DES for the expansion of the size of bits from 64 bits to 112 bits and passing these 112 bits as input for next 3DES function in the chain, keeping the plaintext same all the time.

Why different reduction / expansion functions on each point in rainbow tables?

If different reduction or expansion functions in rainbow tables are not used, then the properties of rainbow tables are compromised, that is rainbow tables will be same as the tabular forms of hash chains that are overall not so much good in efficiency for tabular forms of larger sizes.

Distinguished Points Analysis

The original algorithm involves fixing the length of the chain and also the number of the starting points for the table. We pre compute r tables by picking r distinctive cover capacities. For each veil work m distinctive begin focuses (recognized) will be chosen

randomly. For every begin point a particular chain will be registered till the point that a Distinguish Point is experienced or the point that the chain length is t + 1. Just begin focuses emphasizing to a Distinguished Point in under t cycles will be given away with the relating chain length, the others will be disposed of. In addition, if a similar DP is a last point for a lot of chains, at that point just the chain of maximal length will be given away.
This includes less amount memory unpredictability in comparison with Hellman's trade off.
Precomputation algo: Create (r) tables with (SP, EP, l)-triples, sorted on the basis of endpoints.

1. First a distinguish point property is to be chosen .The property will have d as its order.

2. Choose r number of different mask functions. Each of these mask function will further generate a different function which can be used for reduction.

3. As the probability of achieving the specific endpoint or distinguished point is quite less the length of the chain can be very large. Hence, we choose a max length variable t.
4. Start a continuous loop from 1 to r
(a) Fix, or say, still any m initial starting point
(b) For j = 1 to m, l = 1 to t

i. Calculate f (Starting Point j).

ii. If f (Starting Point j) is a distinguish point then keep track of triple (Starting Point j , End Point j = f (Starting Point j), l)

and iteratively consider next j.

iii. If l > t "let go" Starting Point j and consider following j.

(c) Now arrange the given triplets in sorted fashion on the basis of endpoints. More than one such triplets can be achieved. So, choose the one which has the maximum length.

(d) for every table store the value of the maximum l: lmax

Searching algo: Given Cipher = Encryption K (Plain text) find K.
1. Start a continuous loop from 1 to r
(a) Find lenmax
(b) Y = gi (C).
(c) For I from 1 to lenmax

i. If the given Y is a Distinguised Point

A. If Y is present in table number i, then do

– Consider the givenSP (i) and len l in that particular table.

– But if j is less than l

• Calculate parent K̃ = f−1−j (Starting Point l).

- If C = Encryption K̃ (Plain text) then of course K = K̃: STOP.

• If C 6= Encryption K̃ey (Plain text), consider the upcoming value of i.

B. Otherwise consider upcoming value of i.

ii. Declare Y = f (Y).

The probability of reaching DP is given by the formula:

$P_2(l)=\prod^{l-1}_{i=0}(1-2^{k-d}/2^{k}-i)$

Choosing the average chain length β

$$P_2(l) \simeq (1 - \frac{2^{k-d}}{2^k - \frac{l-1}{2}})^l$$

$$P_1(l) = 1 - \prod_{i=0}^{l-1}(1 - \frac{2^{k-d}}{2^k - i})$$

$$P_1(l) \simeq 1 - (1 - \frac{2^{k-d}}{2^k - \frac{l-1}{2}})^l$$

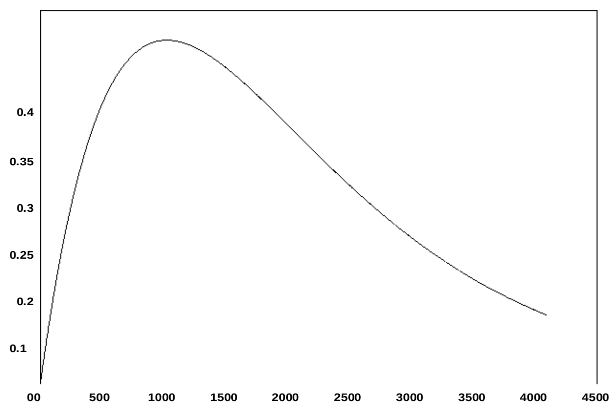


Figure 3.2.2

Average length of chain can be computed as :

$$\beta = \frac{\sum_{l=t_{min}}^{t_{max}} l.P(DP.in.exactly.l.iterations)}{\sum_{l=t_{min}}^{t_{max}} P(DP.in.exactly.l.iterations)}$$

$$\gamma = \sum_{l=t_{min}}^{t_{max}} P(DP.in.exactly.l.iterations) = P_1(t_{max}) - P_1(t_{min} - 1)$$

Numerator can be estimated as follows:

$$\sum_{l=t_{min}}^{t_{max}} l.P(DP.in.exactly.l.iterations)$$
$$= \sum_{l=t_{min}}^{t_{max}} l.(\prod_{i=0}^{l-2}(1 - \frac{2^{k-d}}{2^k - i}) - (\prod_{i=0}^{l-1}(1 - \frac{2^{k-d}}{2^k - i}))$$
$$\simeq \sum_{l=t_{min}}^{t_{max}} l.((1 - \frac{2^{k-d}}{2^k - \frac{t}{2}})^{l-2} - (1 - \frac{2^{k-d}}{2^k - \frac{t}{2}})^{l-1})$$

Where t is the average of maximum t and minimum t

Equation can be rewritten in simpler terms as follows:

$$\sum_{l=t_{min}}^{t_{max}} l.((1-x)^{l-2} - (1-x)^{l-1})$$

Here, $x = \frac{2^{k-d}}{2^k - \frac{t}{2}}.$

$$\sum_{l=t_{min}}^{t_{max}} l.((1-x)^{l-2} - (1-x)^{l-1})$$
$$= t_{min}.(1-x)^{t_{min}-2} - t_{max}.(1-x)^{t_{max}-1} + \sum_{l=t_{min}-1}^{t_{max}-2} (1-x)^l$$
$$= (1-x)^{t_{min}-2}.(t_{min} + \frac{1-x}{x}) - (1-x)^{t_{max}-1}.(t_{max} + \frac{1}{x})$$

At last, the average length of chain can be given as follows :

$$\beta \simeq \frac{(1-x)^{t_{min}-2}.(t_{min}+\frac{1-x}{x})-(1-x)^{t_{max}-1}.(t_{max}+\frac{1}{x})}{\gamma}$$

Firstly the property of distinguished points can be evaluated as follows,

DP-property	Length region (log_2)	Experimental β (log_2)	Theoretical β (log_2)
DP-11	9-13	11.2140	11.2140
DP-12	10-14	12.2137	12.2139
DP-13	12-14	13.0965	13.0966
DP-14	13-15	14.0967	14.0966
DP-15	11-18	15.0771	15.0836

3.2.3

Similarly, influence of chains can be observes as

DP-property	Length region (log_2)	Experimental β (log_2)	Theoretical β (log_2)
DP-13	10-13	11.9790	11.9987
DP-13	12-14	13.0965	13.0966
DP-13	13-16	14.0107	13.9955

3.2.4

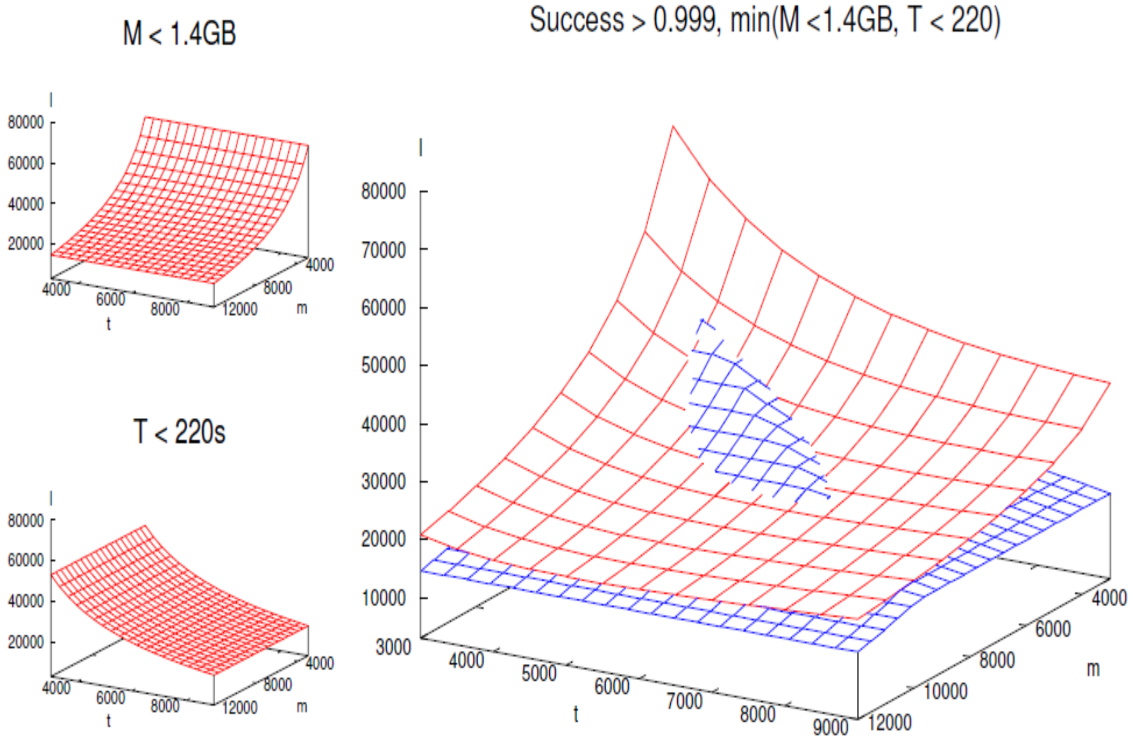
Probability of finding key using table t different keys with m different rows is

$$P_{table} \geq \frac{1}{N}\sum_{i=1}^{m}\sum_{j=0}^{t-1}(1-\frac{it}{N})^{j+1}$$

The overall probability of success with use of 1table only is provided as

$$P_{success} \geq 1-\left(1-\frac{1}{N}\sum_{i=1}^{m}\sum_{j=0}^{t-1}(1-\frac{it}{N})^{j+1}\right)^{\ell}$$

	classic with DP	rainbow
t, m, ℓ	4666, 8192, 4666	4666, 38'223'872, 1
predicted coverage	75.5%	77.5%
measured coverage	75.8%	78.8%



3.2.5

Solution space for probability of success with 99.9%, maximum size of 220 seconds and memory size of 1.4GB

Table os size m x t is provided as

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

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

$$
m \downarrow \overset{t}{\longrightarrow}
\begin{bmatrix}
k^1_{1,1} \xrightarrow{f_1} & \xrightarrow{f_1} & \dots & \xrightarrow{f_1} k^1_{1,t} \\
k^1_{m,1} \xrightarrow{f_1} & \xrightarrow{f_1} & \dots & \xrightarrow{f_1} k^1_{m,t}
\end{bmatrix}
$$

$$
m \downarrow \overset{t}{\longrightarrow}
\begin{bmatrix}
k^2_{1,1} \xrightarrow{f_2} & \xrightarrow{f_2} & \dots & \xrightarrow{f_2} k^2_{1,t} \\
k^2_{m,1} \xrightarrow{f_2} & \xrightarrow{f_2} & \dots & \xrightarrow{f_2} k^2_{m,t}
\end{bmatrix}
$$

$$
\vdots \qquad\qquad \vdots
$$

$$
m \downarrow \overset{t}{\longrightarrow}
\begin{bmatrix}
k^{t-1}_{1,1} \xrightarrow{f_{t-1}} & \xrightarrow{f_{t-1}} & \dots & \xrightarrow{f_{t-1}} k^{t-1}_{1,t} \\
k^{t-1}_{m,1} \xrightarrow{f_{t-1}} & \xrightarrow{f_{t-1}} & \dots & \xrightarrow{f_{t-1}} k^{t-1}_{m,t}
\end{bmatrix}
$$

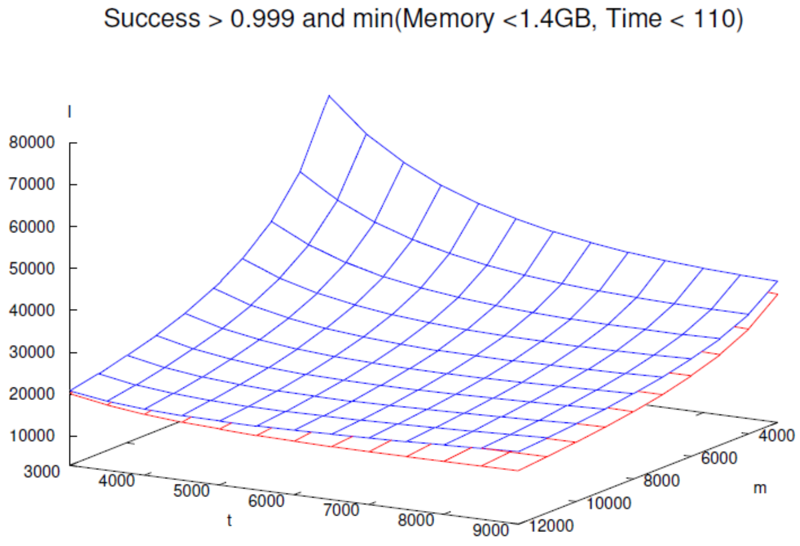
$$
m \downarrow \overset{t}{\longrightarrow}
\begin{bmatrix}
k^t_{1,1} \xrightarrow{f_t} & \xrightarrow{f_t} & \dots & \xrightarrow{f_t} k^t_{1,t} \\
k^t_{m,1} \xrightarrow{f_t} & \xrightarrow{f_t} & \dots & \xrightarrow{f_t} k^t_{m,t}
\end{bmatrix}
$$

3.2.6

And rainbow table of overall size mt x t can be constructed as

$$
m \times t \downarrow \overset{t}{\longrightarrow}
\begin{bmatrix}
k_{1,1} \xrightarrow{f_1} & \xrightarrow{f_2} & \dots & \xrightarrow{f_{t-1}} k_{1,t} \\
 & & & \\
k_{mt,1} \xrightarrow{f_1} & \xrightarrow{f_2} & \dots & \xrightarrow{f_{t-1}} k_{mt,t}
\end{bmatrix}
$$

3.2.7



3.2.8

Comparison between success rates of classical and rainbow tables

Following are the estimations for classic tables with different end points and for rainbow tables after calculating keys for 500 different password hashes

	classic with DP	rainbow
t, m, ℓ	4666, 8192, 4666	4666, 38'223'872, 1
predicted coverage	75.5%	77.5%
measured coverage	75.8%	78.8%

3.2.8

Cryptanalysis statistics with tables yielding 99.9% success rate. We can observe from the middle column easily that rainbow table notes a reduction of 12 times in terms of calculations.

	classic with DP	rainbow	ratio	rainbow sequential	ratio
t, m, ℓ	4666, 7501, 23330	4666, 35M, 5	1	4666, 35M, 5	1
cryptanalysis time	101.4s	66.3	1.5	13.6s	**7.5**
hash calculations	90.3M	7.4M	**12**	11.8M	7.6
false alarms (fa)	7598	1311	5.8	2773	2.7
hashes per fa	9568	4321	2.2	3080	3.1
effort spent on fa	80%	76%	1.1	72%	1.1
success rate	100%	100%	1	100%	1

3.2.9

To know the number of distinguishable chains, we just need to observe overall count of distinct particular points in the ending segment

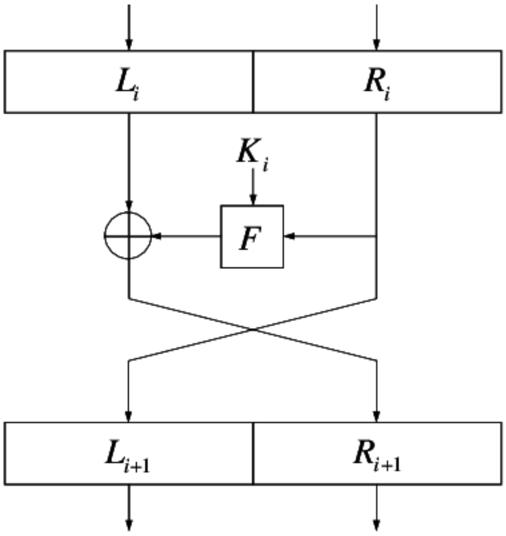
$$\hat{P}_{table} = 1 - e^{-t\frac{m_t}{N}} \quad \text{where} \quad m_1 = N \quad \text{and} \quad m_{n+1} = N\left(1 - e^{-\frac{m_n}{N}}\right)$$

Algorithms:

In this project we are working to break a classical encryption technique called DES. DES stands Data Encrypt Standard. It is a symmetric key algorithm which is partially software and hardware implementation. The algorithm was proposed by Horst Fiestel who at the time worked at IBM in 1970's.The algorithm incorporates Fiestel networks and is a modification of Lucifer cipher scheme. The algorithm served as the standard encryption block cipher algorithm from 1977 to 1998 when it was replaced by AES. The major problem faced was the small size of the key and the possible differential cryptanalysis.

DES is the original square figure—a calculation that takes a settled length string of plaintext bits and changes it through a progression of convoluted activities into another ciphertext bitstring of a similar length. On account of DES, the square size is 64 bits. DES likewise utilizes a key to redo the change, with the goal that decoding can apparently just be performed by the individuals who realize the specific key used to scramble. The key apparently comprises of 64 bits; in any case, just 56 of these are really utilized by the calculation. Eight bits are utilized exclusively to check equality, and are from that point disposed of. Thus the viable key length is 56 bits. DES is never used natively. Hence, some mode of operation like Cipher Block Chaining, output feedback etc. are always used. The process of decryption is similar with the only difference being the opposite order of the used keys.

DES is based on something called the Feistal network. The network consists of smaller blocks which individually instil some confusion and diffusion. The total input bits are divided into two equal parts (here 32 bits each).They are both treated differently with the left bits going undisturbed to the right portion of the next Feistel block. The right bits are first fed into a function called Feistel function and are then XORed with the left 32 bits. This output is then take to the left block of the next round.



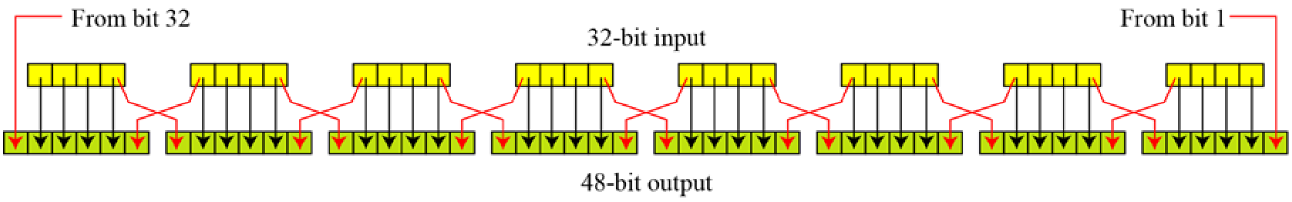
A Classical Feistel Network

4.1

Feistel function:

The feistel function works on half of the bits at a time. Each undergoes through four main steps.

1. In this step the input's 32 bits are expanded into 48 bits. This is done by the use of an permutation expansion box. The input is first divided into eight four bits blocks. Now, each of these four bits are expanded into 6 bits.



4.2

2. The above 48 bits are XORed with the 48 bits of the key. It is to be noted that the input key size was only 56 bits. So, here we use a pseudo random generator to actually produce the required key. The step is known as the key-mixing step.

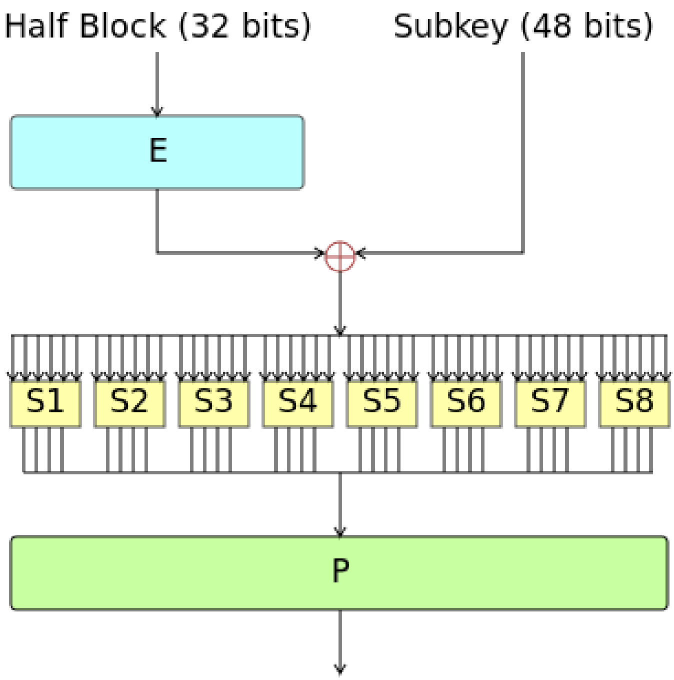
In this step a substitution box is used. This is used to convert the 48 bits output to 32 bits. The input size is divided into 6 blocks, each of 8 bits. Now we have a two dimensional matrix containing 4rows and 16 columns. The first and the last bit are used to find the row and the middle four bits are used to locate the column. Each cell has a 4 bit value and we substitute the given 6 bits with these 4 bits. After doing this for all 8 blocks we will have a resulting size of 8*4 i.e. 32 bits. It is to be kept in mind that we need to prevent any kind of linearity in the construction of these s-boxes. Moreover, for all 8 blocks of data different s-box is used.

4. Now the output 32 bits are spread out using a permutation block. The basic functionality of a p-box is to scramble the output and to distribute the effect of the s-boxes to a wider range. This is called the permutation step.

R1=L0

L1=f(r)^L0

Here, R1 that is the right portion stores the value of previous left part of plaintext and L1 stores the output of previous left part XORed with overall computation of function f(r).



4.3

The Key-Mixing step

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	04	13	01	02	15	11	08	03	10	06	12	05	09	00	07
1	00	15	07	04	14	02	13	10	03	06	12	11	09	05	03	08
2	04	01	14	08	13	06	02	11	15	12	09	07	03	10	05	00
3	15	12	08	02	04	09	01	07	05	11	03	14	10	00	06	13

Sample Substitution Box

4.4

Key Scheduling Algorithm

For each function in the feistel round we need a key of 48 bits. As a lot of such steps involving different keys can be there we cannot use the input key directly. Hence, we need a key scheduling algorithm.

Steps involved in key scheduling:

1. The 56 bits input key bits are separated into two parts. These keys have a size of 28 bits.

The bits are shifted by some positions. The first, second, ninth and sixteenth positions are shifted by 1 bit to the right. All the remaining bits are shifted by two bit positions to right.

3. Now a D-box is used. The bits convert 28 bits of the key into 24 bits. The left and the right part are now combined to give us the 48 bits which are used for the encryption process.

14	17	11	24	01	05	03	28
15	06	21	10	23	19	12	04
26	08	16	07	27	20	13	02
41	52	31	37	47	55	30	40
51	45	33	48	44	49	39	56
34	53	46	42	50	36	29	32

Figure 4.5

The first three rows of the table are used for the compression of the left part of the key while the lower three keys are used to reduce the size of the right part.

Data Encryption Standard

This algorithm is built by the use of Feistel network. Sixteen Feistel blocks are used in DES. The block size for this algorithm is 64 bits. Hence, DES is nothing but Feistel rounds applied 16 times.

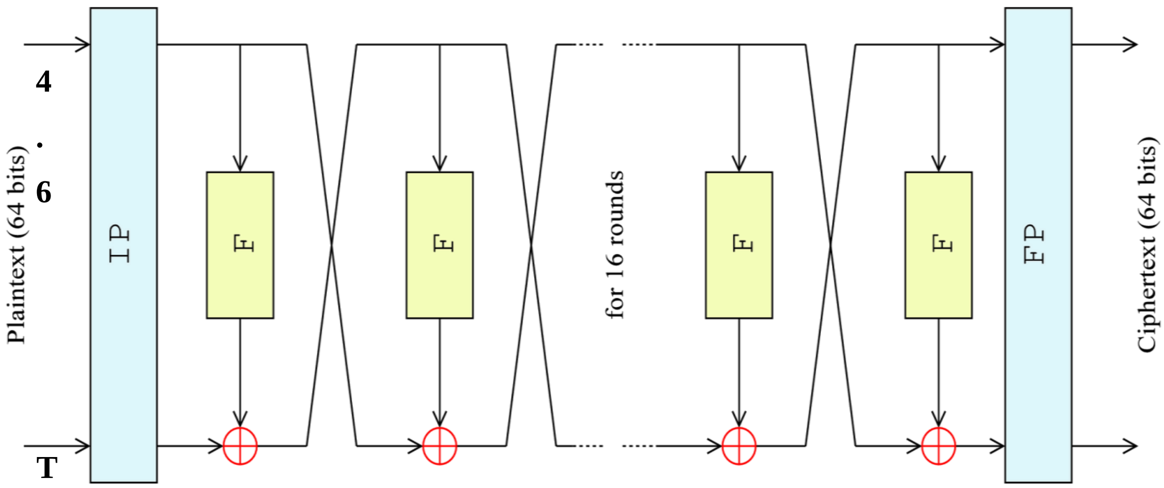


Figure 4.6

Trudy's Perfect World

Initially, when Time Memory Trade off Attack was just conceptualised, nobody thought about the problems that can occur within the chains, such as merging of chains, detection of loops and circles in chains.

Trudy, a researcher provided his thoughts in this proposed solution, which are now considered as Trudy's Perfect World. According to this, there cannot be any overlaps in the chains and every cipher text is in different chain.

Trudy gave a simple concept that suppose we consider DES. Here, length of chain is 56 bits and suppose we find 2^28 chains of lengths also ranging to 2^28. So, easily we can say that it will acquire a total memory of space for 2^28 different pairs.

Also, time required to find the solution will also be around 2^28 for each attack. Every time, we can find an endpoint from the starting point in 2^27 steps and in the same way we can look for K in 2^27 steps again. So, in this way, there will be no chance of failure of attack.

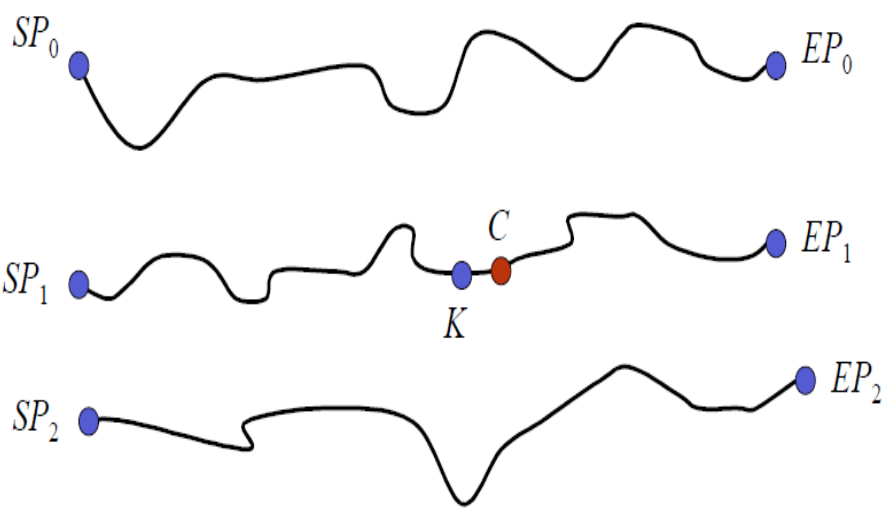


Figure 4.6.1

As described in the picture above, according to Trudy's Perfect World scenario, there cannot be any loops, overlaps, merges or circles in the chains. Every chain is perfect and every cipher text exists only in one chain, and only one single time

Real World Scenario (exactly opposite to Trudy's perfect world)

In real world implementation of chains, chains are not so well behaving. There can be loops, circles, overlaps and also merging in chains. Due to these real world exceptions, we can find the encrypted text we are looking for in the endpoints (EP) but this doesn't guarantee that we will also find the solution, that is, the key in that particular chain.

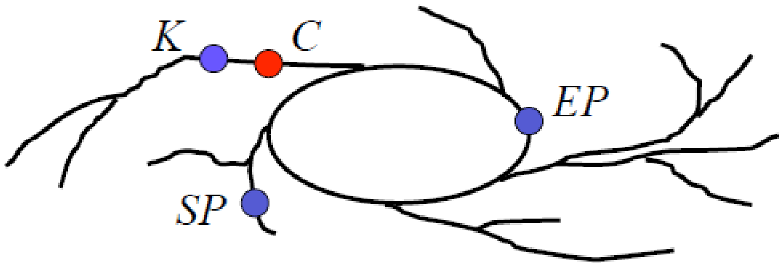


Figure 4.6.2

In real world, to avoid such situations, one can reduce merging by opting for different reduction functions at each point in the chains. In this way, there may be possibilities of intersection in the chains, but they will not overlap and go on merging and winding with each other.

Reduction function can be of any type. Say, one reduction function can add some binary string format to the output before storing the output in the chain. Other type of reduction function can be possibly the one that shuffles the output in random fashion at each point in the chain.

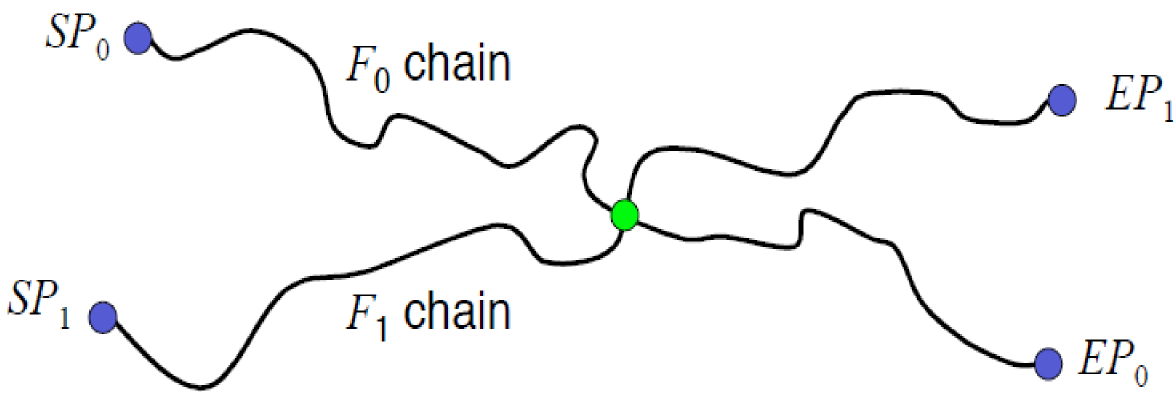


Figure 4.6.3

Hellman Analysis

In this approach we choose m start points and fix a chain length represented by size t. Now as discussed before due to the problems like merges and loops we need to construct large number of tables.

How is the table constructed?

Pre-computation Phase

TMTO is specifically a known plaintext attack. So we have a value P for plaintext. We consider these m start points as the key. So at start point we encrypt the plaintext using this key.

$C=S_k(P)$

Now the output of the above step is of 64 bits. But the size of the key is 56 bits. So, we use a reduction function.

$f(K)=R[S_k(P)]$

S is the encryption algorithm.

R is the reduction algorithm.

Hellman proposed this function as the one dropping the last eight bits of the ciphertext produced.

Now, this algorithm is applied t times and the result is obtained correspondingly. As storing all of these pairs will be a tremendous memory overhead we only store the first and the last column as pair.

Diagram below shows a pictorial view of how chains are being created, and the creation of chains starting from a starting point say SP, to the end that is the end point of the chain, named as EP.

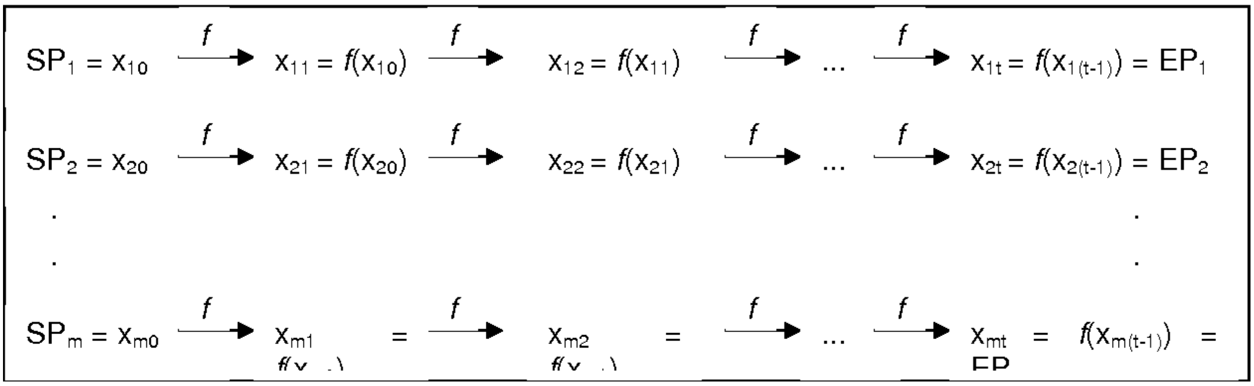


Figure 4.7

Algorithm:

The algorithm for offline phaseis as follows:

●M random points of 56 bits are selected. These serve as the starting point or key as in DES.

● For each starting point a chain of length T is formed.

● The starting point is taken and is fed into the algorithm.

● If the output size is different than the required size (56 bits as in DES) the output bits are truncated.

● The same procedure is repeated T times.

● The starting and the end point are saved.

● The above steps are repeated for M different start points.

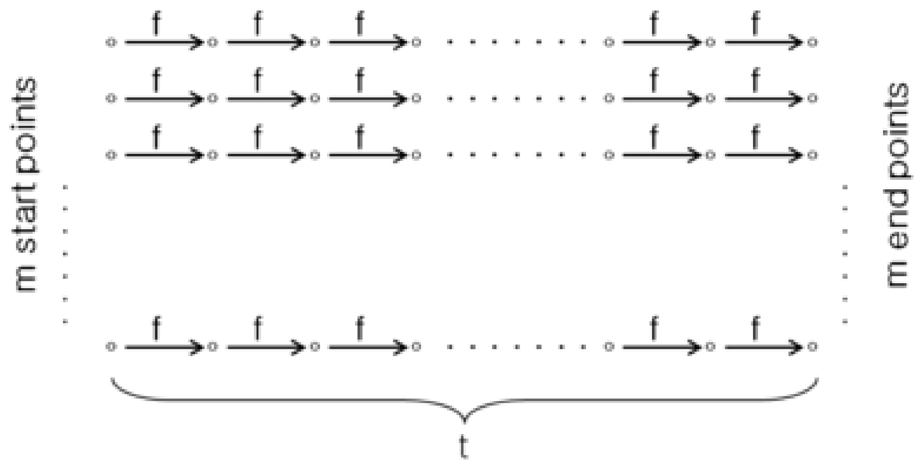


Figure 4.8

Online Phase:

Algorithm

Algorithm for online phase is explained as follows

- This phase involves searching the table for the required ciphertext.
- Firstly , the ciphertext is truncated to 56 bits.

- Now the end points of the various chains are searched for the same ciphertext bits.

- If the same bits are encountered the chain is regenerated to find the t entry and the output of this t after applying the reduction function is the key.

- If the bits don't match then the reduced ciphertext bits are encrypted and reduced p time 1<=p<=T.

- Each time they are checked against the required ciphertext's bits.

- When a match is found the chain is generated T-p times and we derive the key from there.

Hellman suggested to follow the relation:

M=m*t

mt^2=N

m=t=$N^{(1/3)}$

Breaking the DES via this method was reduced to 2^38.

Improvements by Perfect Table:

In 2003 Philippe Oechslin showed that the performance of the TMTO can be increased drastically b using different reduction function.

The probability of collision between two chains decreased to 1/t.

Rainbow chains have the following advantages:

- There is an overall reduction by a factor of t in table look ups using rainbow tables, if compared with original implementation proposed by Martin Hellman.

●Therefore, merge free tables can be created using rainbow tables. Still, there can be collisions in the chains.

● There cannot be any loops in rainbow tables as reduction function changes iteratively at every point in the chain. This saves our time as we don’t have to look for loops in our chains and remove them, and also, all values of encrypted texts can still be covered, and that too, without any loops or circles.

Philippe Oechslin further extended his work and worked to find the optimum values of the parameters m, t and l and worked to create something called perfect tables.Perfect tables have 0 merges and no memory is wasted.

Given M (memory available), N and P(required probability of success), following are the optimal metrics that lowers down the overall time of cryptanalysis:

l=-ln(1-P)/2

m=M/l

t=(-N*ln(1-p))/M

Using the above parameters we tried implemented TMTO for DES :

P=.86

l=1

m=34359738368

t=262144

Working of our Offline phase implementation

In offline phase, I created a C++14 function of Data Encryption Standard (DES) that takes input of plain text and key in string format and outputs the result, that is encrypted text in string format to the main function.

Along with DES function, I have also implemented other useful functions in my code such as binary to hexadecimal conversion and vice versa, conversion from string to integer and also, iterative binary search algorithms.

I have prepared a double dimensional matrix of 200 rows and 200 columns which represents 200 chains of 200 lengths each. After all this computational work, I created another array of 200 rows and 2 columns to store the starting and ending points of the chains. Then I dumped the 200 chains to save memory and sorted the array with respect to ending points of the chains so that I can perform binary search later.

Now, there is a field present to enter the encrypted text whose key you want to search for and using binary searching algorithm, one can easily find in logarithmic time that whether

encrypted text is present in last column or not, and on which index. This is the overall implementation of offline phase of time memory trade off attack.

Creation of 200 chains with each chain containing 200 values are shown below

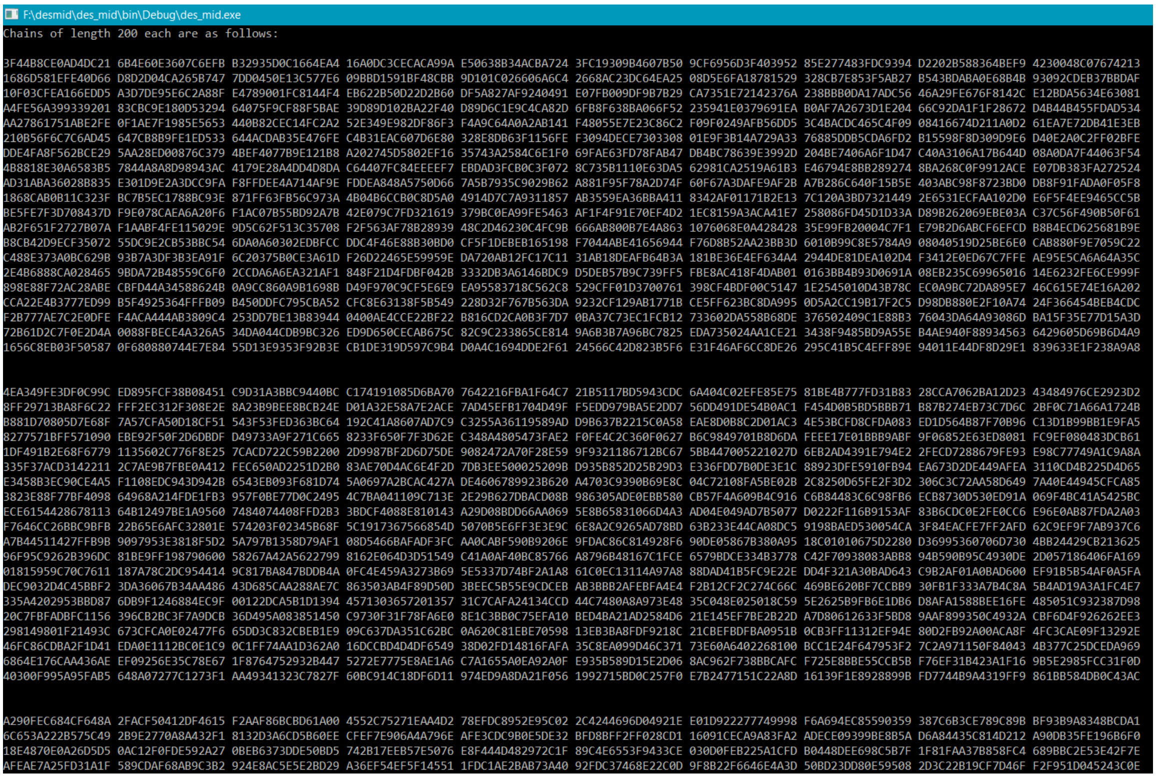


Figure 4.8(a)

Computation of binary search over the array containing first and last elements of chains in a sorted fashion based on end points

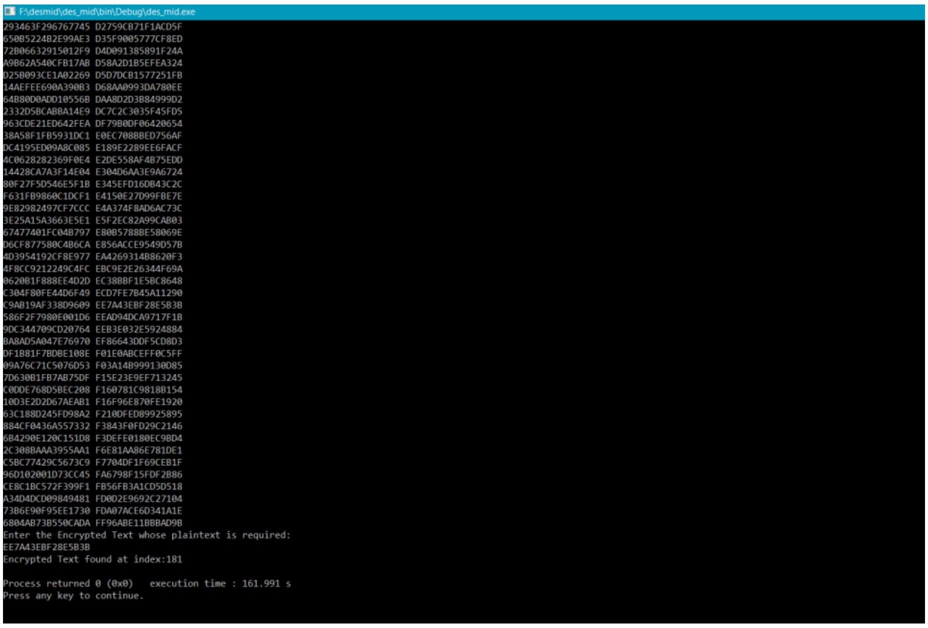


Figure 4.8(b)

Working of our Online Phase implementation

In online phase, file handling is used to transfer the data from offline phase to online phase. Here, one can enter the index at which the encrypted text is located to compute that particular chain from starting point to (t-1) length as the (t-1)th index will hold the key for that particular encryption.

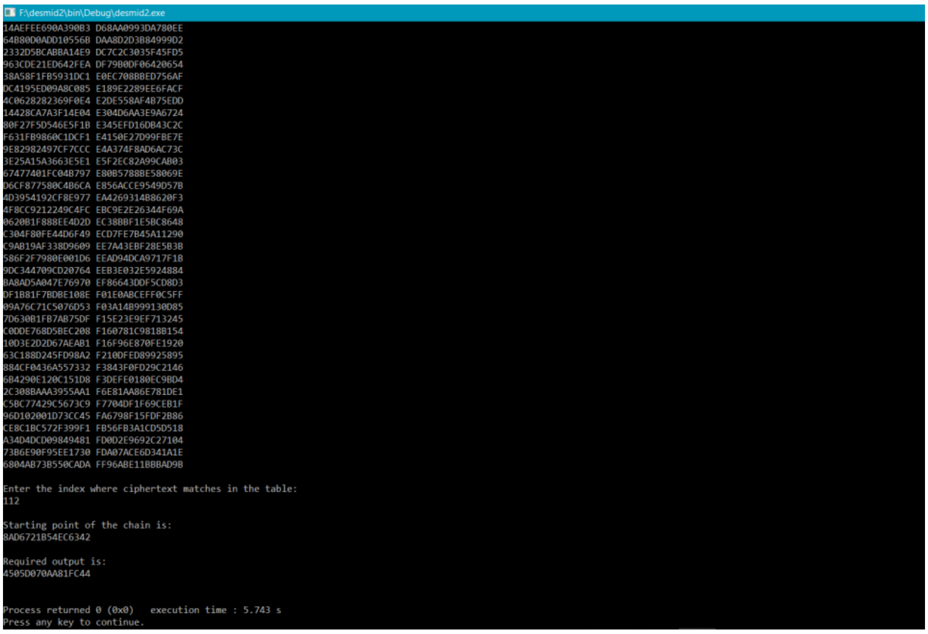


Figure 4.8(c)

But if the user enters -1, that is the encrypted text is not present in the endpoints, then we need to compute DES on that encrypted text again and look for the output in the endpoints.

We have to repeat this step till we find the encryption among the endpoints. Also, we need to keep a track of number of times encryption is computed again and again, say n. Then, we need to compute the chain length up to (t-n)th position as the required key is present at that index. This is the overall implementation of Online Phase in our time memory trade off attack.

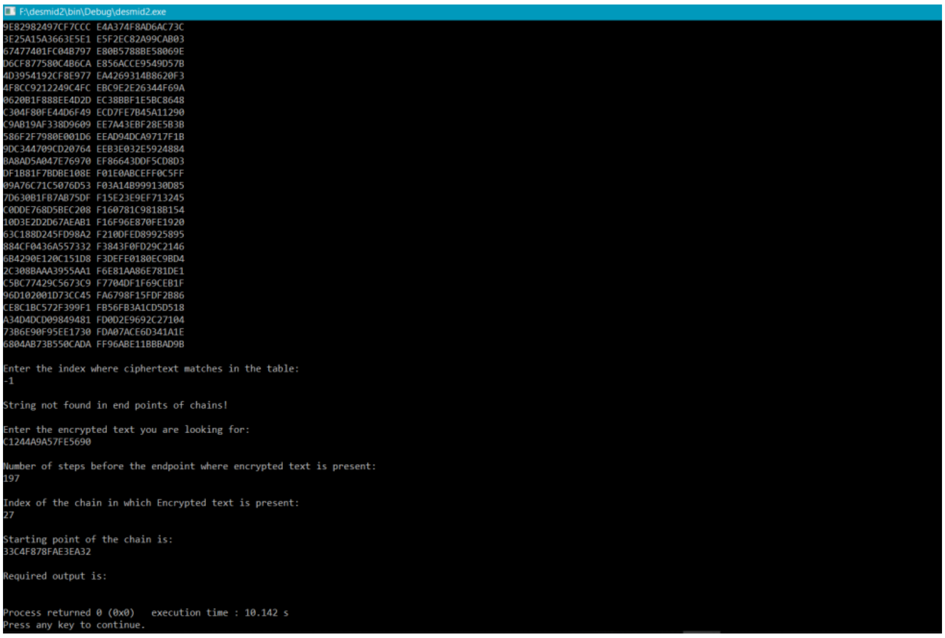


Figure 4.8(d)

Our Rainbow Tables Implementation

In implementation of rainbow tables, the major change as compared to normal solution proposed by Martin Hellman is in the reduction function. In this implementation, we need to change the reduction function at each step so that we cn avoid loops, circles and merging of chains.

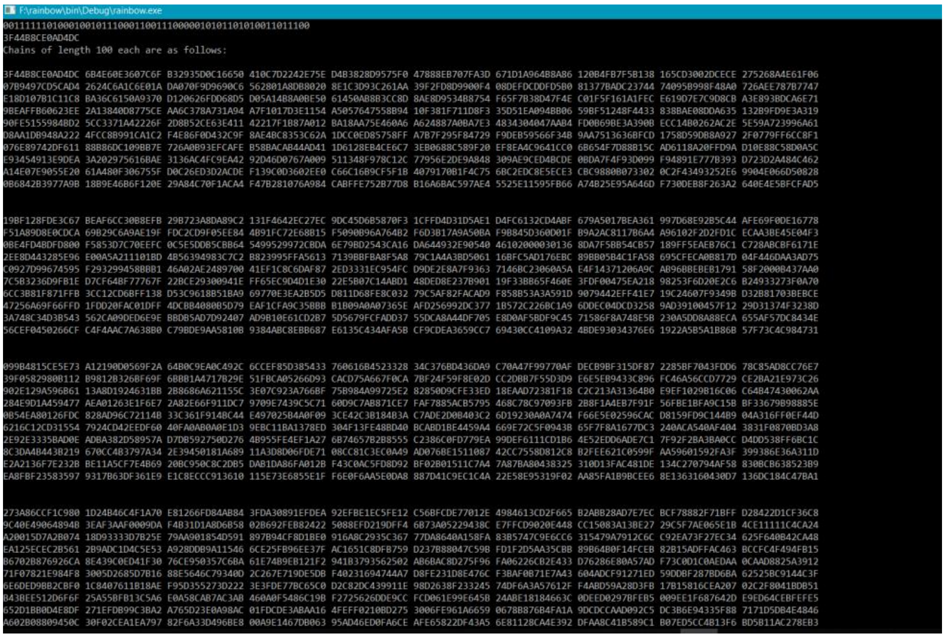


Figure 4.8(e)

Here, we have taken a counter or iterator, and after calculation of encryption at each step, we are adding the binary forms of both iterator and calculated encryption and then we are storing the final result in the array representing chains.

Now, because reduction function is changing at each step, there is no chance that two chains can have same output after reduction function at same point in respective chains, so these chains simply cannot merge. Also, due to different reduction functions, there are minor or less chances of getting loops or circles in the chains.

Success Probability

Consider 3 different parameters say a, b and c where a are the number of random starting points for every F, b stands for number of different encryptions in each chain and c stands for number of different tables or random functions F taken into account.

If a, b and c are chosen as 2^k/3 then the probability of success is near about 0.55 in lower bound. With this lower bound of probability, it requires about ‘a*b’ amount of memory and ‘b*c’ about of time for each TMTO attack. Also, for pre calculation, that is offline phase, it requires about ‘a*b*c’ amount of total work to be done.

Success probability can be calculated using the same approach as used for occupancy problem, that is, if you have to put n different balls in m different boxes, what are the expected numbers of boxes with at least one ball in them?

If the size of the key is considered as k bits length, then the success probability can be given as follows:

Probaility (success) = 1 - (e^-abc/k)
Where a*b*c is the amount of work required to be done in offline phase.

The overall ranging of probability of success is defined in the table below:

mtr	P(success)
0	0
2^{k-5}	0.03
2^{k-4}	0.06
2^{k-3}	0.12
2^{k-2}	0.22
2^{k-1}	0.39
2^{k}	0.63
2^{k+1}	0.86
2^{k+2}	0.98
2^{k+3}	0.99
∞	1.00

Table 4.9

5.Test Plan

The proposed system needs a high processing power. We are interested in implementing the perfect rainbow table method. This method involves creation of the table for which the DES encryption algorithm has to be used several times. As per the proposed scheme we choose the appropriate parameters and create this table. For the creation of the perfect rainbow table approximately about 13 days will be consumed.

The expected encryption rate for our DES implementation is about 2^{36} encryptions/sec. This will generate a large number of encryptions. The total size of these encryptions will be about 2^14 GB. Do we intend to store all of these? No. (Even if we do, we won't be able to) !! We will just be choosing a limited number of starting points with a fixed chain length. We will save only the pairs consisting of the starting point and the end point.

But we have another question in front of us. How are we going to achieve 2^{36} encryptions/sec. Our optimised implementation of DES incorporates bit slicing. A technique suggested by Philipp Grabher, Johann Großsch ̈adl, and Dan Page in 1996 which can significantly increase the speed. With our optimized implementation and bit slicing we achieved a speed of about 2^{22}on a Intel i5,fourth generation machine. Now for increasing the speed we would be needing dedicated GPU's and need to code them for our requirement. The preferred system would be a workstation consisting of 8 GTX 1080i GPU's working simultaneously. Where can we get these? Provided the situation we don't have a workstation

with the requirements mentioned we can hire a cloud system to do this work for us. After considering several services we have decided to go with **vscaler cloud services.** The cost involved certainly is a major factor involved here.

Following metrics for analysis should be used:

1. Money
2. Time
3. Success Probability
4. Availability of resources

Now different approaches could result in different results.

Metrics If Hellman Analysis Is Used

1.**Money**: In Hellman Analysis the problem of merges and loops prevails. So we need to create a large number of tables. Because of this the amount of memory required increases immensely. The cost for the solution as mentioned in the original solution was about

$3.5M.But with time the cost of various hardware components has decreased and the native implemented solution will cost about $12000.

2.**Time**: The pre-processing phase of the Hellman method takes a longer time due to the various problems mentioned above. Nonetheless, the solution decreases the total time required to $N^{2/3}$ for the pre-processing phase. The time consumed in the online phase is equivalent to time of searching. This is about O(logN+t).The time as per today is equivalent to the time involved in the pre computation phase which is about 100 days.

3.**Success Probablity**: The probability of success as proposed in the original paper is (m*t)/N.[m-the number of start points, t-the length of chain, N-The total search space].But merges and loops decrease the success probability to some extent. If we take sufficiently large m and t then a success probability of a range between .63 and .75 can be achieved.

Metrics If Distinguished Points Analysis Is Used

1.**Money**: In Distinguished Point the problem of merges and loops still prevails though to a lower extent. We need to create a comparatively smaller number of tables. The cost for the solution as mentioned in the original solution was about $100k. But with time the cost of various hardware components has decreased and the native implemented solution will cost about $9000.

2.**Time**: The pre-processing phase of the Hellman method takes a longer time due to the various problems mentioned above. Nonetheless, the solution decreases the total time required to $N^{2/3}$ for the pre-processing phase. The time consumed in the online phase is equivalent to time of searching. This is about O(logN+t). The time as per today is equivalent to the time involved in the pre computation phase which is about 47 days.

3.**Success Probablity**: The probability of success as proposed in the original paper is (m*t)/N.[m-the number of start points, t-the length of chain, N-The total search space].But merges and loops decrease the success probability to some extent. If we take sufficiently large m and t then a success probability of a range between .7 and .85 can be achieved.

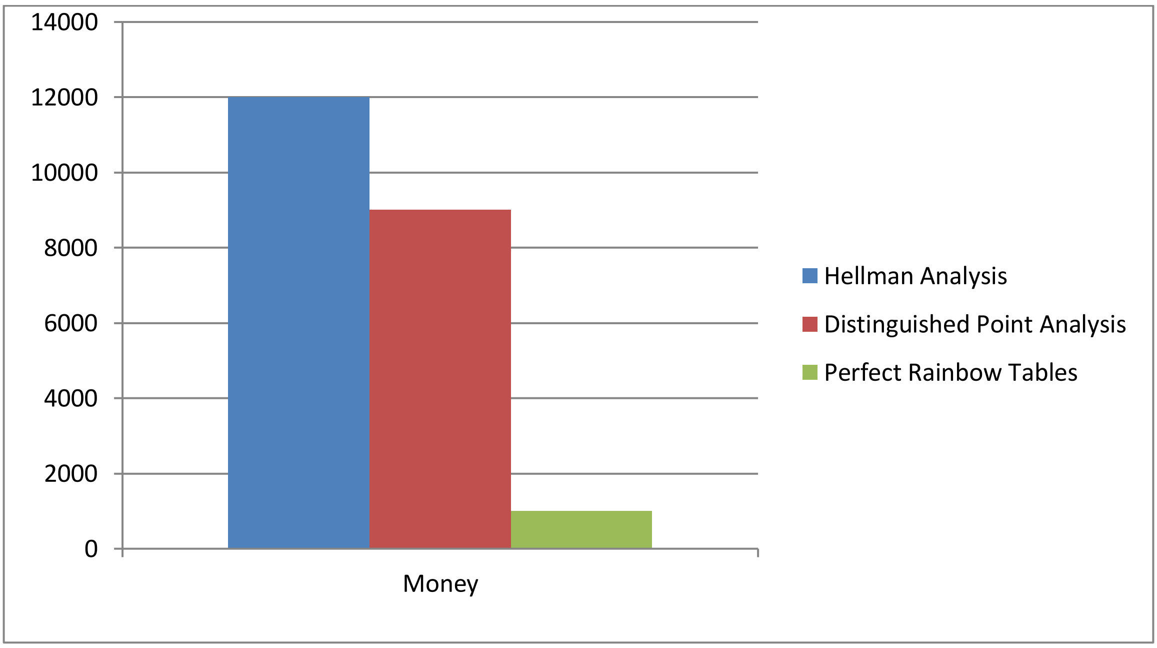
Metrics If Perfect Rainbow Tables Are Used

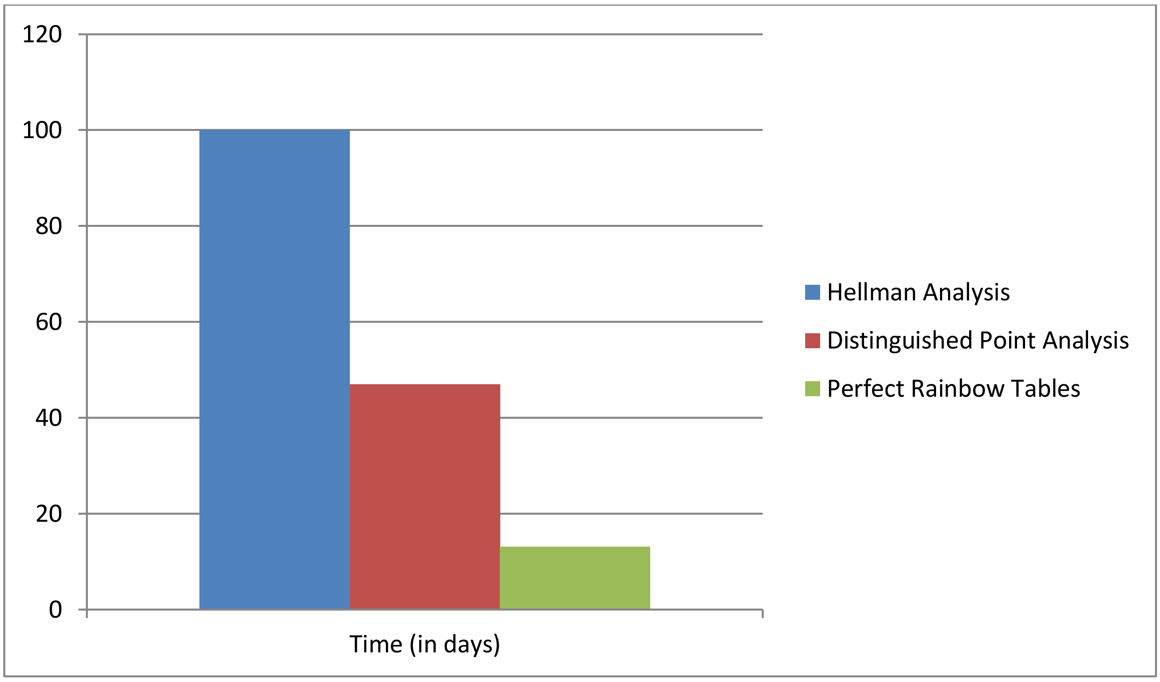
1. Money: Perfect Tables tackle the problem of merges, loops and false alarms. This helps in reducing the amount of memory required by a large amount. Moreover less work needs to be done even in the process of creating the table. The total amount for setup is as minor as 1000$.

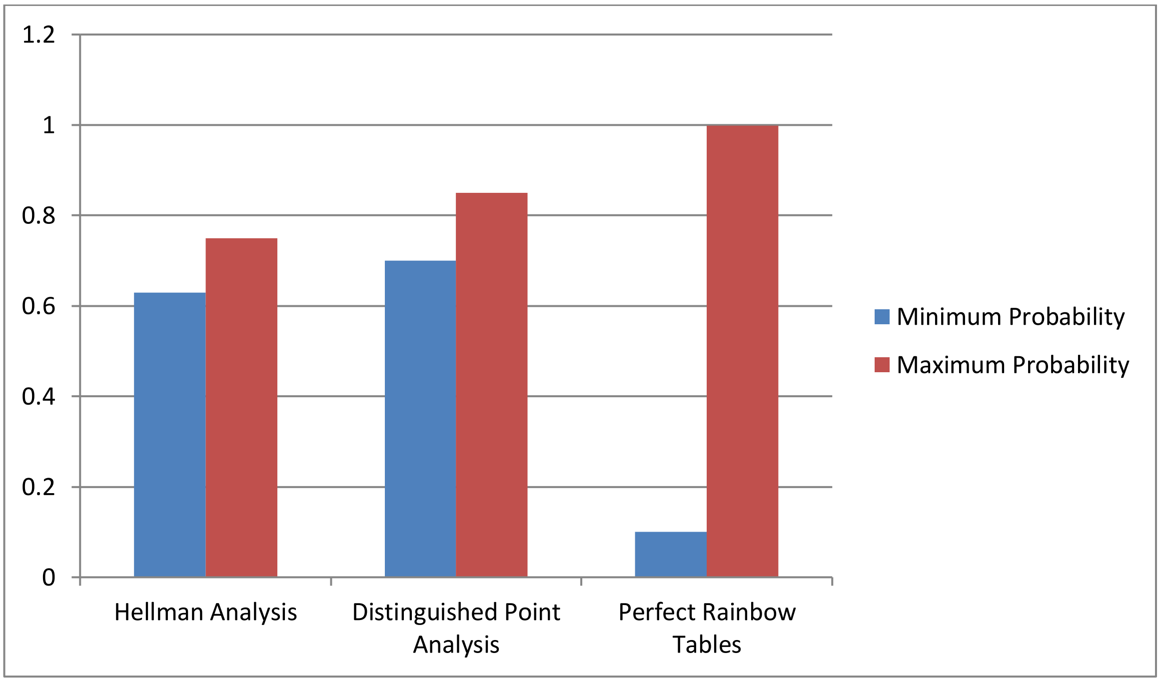
2. **Time**:The pre-processing phase a comparatively less time because of lack of merges and loops. But some time has to be spent on keeping a check on these merges. Yet it is comparatively smaller. The total time consumed in the precomputation phase is about 13 days for a success probability of .86.

3. **Success Probability**: The success in the range of .1 to .999 can be achieved.

Comparison Bargraphs







Data Required:

Time Memory Trade Off is essentially a chosen plaintext attack. So, we need to know a plaintext in advance. Generally we try to exploit our existant knowledge. This included exploiting our knowledge of various protocols. We know that some things are already fixed as the part of the protocol. This helps up to attack all the messages and data falling under a particular protocol.

6. Chapter-6 CONCLUSIONS

6.1 Conclusions

Thus so far in our project we have done and covered the research part. We read and collected all the required information to perform this. Moreover, we have constructed a robust framework which will help us to proactively deal with the various problems. Having developed a good workflow we can perform the project provided we are provided with the necessary resources.

6.2 Future scope

Being a generalized attack, it has already been established that we can perform the Time Memory Tradeoff attack on most of the algorithms present out there. Hence, this approach can be easily applied to any problem with a defined search space.

There are different encryption algorithms present and being used such as 3DES and AES on which naïve attack approaches such as Brute force attacks and attacks storing all possible results do not provide efficient search solutions.

But with Time Memory Trade Off Attack, we can easily compute the time that a much better attack can take to break such encryption algorithms. If found that algorithm can be attacked in a feasible possible time, we can head towards a new, safer and much secure encryption algorithm and should replace the existing one in real world scenario as usage of such an algorithm simply means we are compromising with security provided in real world cases, ranging from banking encryptions to our daily used social media passwords.

REFERENCES

1.https://en.wikipedia.org/wiki/Data_Encryption_Standard

2. https://en.wikipedia.org/wiki/Time/Time_memory_tradeoff

3. https://ee.stanford.edu/~hellman/publications/36.pdf "A cryptanalytic Time-Memory Trade-Off" by Martin E. Hellman

4. https://perso.uclouvain.be/fstandae/PUBLIS/2.pdf "A Time-Memory Tradeoff using Distin. guished Points"

5. https://eprint.iacr.org/2008/054.pdf "Variants of the Distinguished Point Method for Cryptanlytic Time Memory Trade-Off"

6. https://perso.uclouvain.be/fstandae/PUBLIS/74.pdf "Time Memory Trade-offs"

7. https://crypto.junod.info/jacm08.pdf "Characterization and Improvement of Time Memory Trade-Off Based on Perfect Tables"

8. https://lasec.epfl.ch/pub/lasec/doc/Oech03.pdf "Making a faster Cryptanalytic Time Memory Trade-Off"