An Exploration of Modern Cryptography

Siddharth
Mahendraker

April 23, 2012

Abstract

Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus. Maecenas sed diam eget risus varius blandit sit amet non magna. Duis mollis, est non commodo luctus, nisi erat porttitor ligula, eget lacinia odio sem nec elit. Vivamus sagittis lacus vel augue laoreet rutrum faucibus dolor auctor. Aenean eu leo quam. Pellentesque ornare sem lacinia quam venenatis vestibulum. Nulla vitae elit libero, a pharetra augue. Donec id elit non mi porta gravida at eget metus.

Contents

Introduction

Suppose two people, Alice and Bob, wish to communicate by mail and do not want their mailwoman, Eve, to be able to read their messages. Alice and Bob are military personal of the same country, but they have never met each other before. Because Eve is the mailwoman, she will be able to read all of the messages passing between Alice and Bob, but her obligation to the postal service prevents her from tampering with these messages[1].

The question is, is it possible for Alice and Bob to communicate securely in these circumstances? Astoundingly, the answer is yes!

Cryptography is the science of securely sending messages over insecure channels. Using cryptographic techniques, Alice and Bob can be sure that their communications are illegible to Eve.

0.1 Terminology and Basic Concepts

0.1.1 Alice and Bob

Unless specified otherwise, Alice and Bob are two parties attempting to communicate over an insecure channel, and Eve is their adversary trying to read their messages.

0.1.2 Encryption and Decryption

Messages in cryptography are formally called plaintext. When messages are scrambled, or made "illegible", they are encrypted. The encrypted form of these messages is called ciphertext. The reverse process of encryption, decryption, accepts ciphertext as input and returns plaintext. We can describe this mathematically as:

$$E(M) = C$$
$$E'(C) = D(C) = M$$

Where M denotes plaintext, C denotes cipher text, E is the encryption function and D is the decryption function, or the inverse of E. Also note

[1]In reality, Eve would not be bound by such a petty obligation, however, for the sake of simplicity, let us assume this is true.

the following identity:

$$D(E(M)) = M$$
$$E(D(C)) = C$$

0.1.3 Ciphers and Keys

A cryptographic algorithm, or cipher, is a function used for encryption and decryption.

If the workings of a cipher are made public, then the messages of anyone who is known to use the cipher can quickly be compromised by simply implementing an inverse of the cipher. Therefore, cryptographers introduced a key. A key is a piece of secret, private information upon which the cipher depends. It often takes the form of a number. The total number of possible values a key can take on is called they keyspace. Because ciphers depend on the key to encrypt and decrypt plaintext, encryption and decryption is often denoted:

$$E_K(P) = C$$
$$D_K(C) = P$$

The key is denoted by K and the keyspace is by $\mathcal{K}$. Note that the identity mentioned in 0.1.2 still holds true for ciphers.

0.1.4 Symmetric and Public-Key Ciphers

There are two distinct kinds of ciphers, symmetric ciphers and public-key (or asymmetric) ciphers.

Symmetric ciphers are ciphers for which the key used to decrypt ciphertext and encrypt plaintext is the same. In most symmetric ciphers, this means that Alice and Bob will need to agree on a key before they can begin sending messages. Symmetric ciphers can be further categorized as stream ciphers or block ciphers. Stream ciphers operate on only one bit (or byte) of plaintext at a time, where as block ciphers operate on a large number of bytes at once.

Public-key ciphers are ciphers for which the the key used for encrypting plaintext is different from the the key used for decrypting ciphertext. Further, these keys should be independent of each other, meaning the decryption key can not be calculated[2]from the encryption key. The design of this cipher is such that the encryption key can be published for anyone to use, but only the owner of the decryption key can decrypt the message. This is why is encryption key is referred to as the public key and the decryption key is referred to as the private key.

0.1.5 Cryptanalysis

Crytanalysis is the study of obtaining the plaintext from encrypted messages without the knowledge of the key. An attempt to cryptanalyse a cipher is called an attack. Successful attacks often reveal either the plaintext, the secret key, or both.

The only assumption made in cryptanalysis is that the only piece of information the users of the cipher, Alice and Bob know that the adversary Eve does not is the secret key. This means that all other information, including communications and the workings of their cryptographic algorithm are available to anyone. This assumption implies that the security of the algorithm rests only in the key, and nothing else.

There are three main cryptanalysis techniques we will be focusing on in this report. Listed in decreasing order of difficulty they are; ciphertext only attacks, known-plaintext attacks and chosen plaintext attacks.

In ciphertext only attacks, the cryptanalyst (or attacker) Eve has access to several different ciphertexts. The attack is considered successful if Eve successfully retrieves the plaintexts corresponding to the ciphertexts or the key used in encryption.

In known-plaintext attacks, Eve has access to the ciphertexts as well as their corresponding plaintexts. The attack is successful if Eve finds the key (or keys) used to encrypt each plaintext.

In chosen plaintext attacks, Eve can not only access the ciphertexts, and their corresponding plaintext, but can also choose which plaintexts are encrypted and which are decrypted. The attack is successful if Eve retrieves the key (or keys) used to encrypt each plaintext.

Note that in all of the cases above, the adversary, Eve, had to know some amount of "information" about the ciphertext, plaintext or the relationship between the two. The only other technique which can yield the key is a brute force attack or exhaustive search attack, in which Eve checks the

[2]In a reasonably amount of time ofcourse.

ciphertext against all possible keys in the keyspace until one of the keys reveals the plaintext.

1 Substitution Ciphers

A substitution cipher is a cipher in which each character or byte in the plaintext is substituted with a character or byte in the cipher text.

The substitution cipher we will be analyzing is called the Caesar cipher. The Caesar cipher operates on one byte (or character) of the plaintext at a time. Briefly, the value of the key is added to each byte of the plaintext. If this sum is a byte which does not corresponds to a character in the English alphabet, the sum is set modulo 26. Decryption works the opposite way. The value of the key is subtracted from each byte of the ciphertext. If the sum is not in the English alphabet, the sum is set modulo 26.

This can be more concisely explained in pseudocode as:

Algorithm 1 Caesar cipher

```
$i \leftarrow getByte()$
$i \leftarrow i + k$
if $i > 26$ then
    $i \leftarrow i - 26$
end if
return $i$
```

1.1 Information Theory and Languages

Before we begin a deconstruction of the Caesar cipher, there are a few assumptions we make that must be explained.

Firstly, we must clarify the definition of information we used in section 0.X.X. Information can be rigorously defined as the least number of bits it would take to represent all possible meanings of a message, assuming all messages are equally likely.

For example, suppose we are trying to determine the amount of information in a list of possible sexes:

```
1. Male
2. Female
```

Clearly, this data can be represented using one bit, where the 1 represents male and 0 represents female. Therefore, we can say that there is only one bit of information present in this list.

Now, if we take a look at words used in the English language, we clearly see that English does not represent this information very succintly, e.i there is lots of redundance per character.

For example, the sentence "met u tmrw @ 9" conveys the same information as the sentence "meet you tomorrow at nine", yet does do much more succinctly. Therefore, we could say that many of the character in the latter sentence are redundant or useless.

Although this may not seem related at all to cryptography, it is. This redundancy in languages causes sentences to "leak" more information than they need to. As we shall soon see, this often manifests itself as discrepancies in the frequency and location of certain characters in relation to others, and makes breaking the Caesar cipher a piece of cake.

1.2 Cryptanalysis of the Caesar Cipher

If we take the most nave cryptanalytic approach, a brute force attack, the Caesar cipher appears quite strong. Indeed the keyspace of the cipher is $26!$ or approximately 4×10^{26}! This means that even if we were to check even a million keys per second, it would still take us around 1.27×10^{13} years to check every possible key! That longer than the estimated age of the universe!!

However, we know from our understanding of redundancy in language that there is information being leaked here.

Because the output of this algorithm merely "switches" one letter with another, the letters in any particular ciphertext will continue to follow the known statistic rules regarding English text. Particularly, they will maintain certain distributions of characters, bi-grams and tri-grams over the message.

Letter	Frequency (%)		
E	13.11	M	2.55
T	10.47	U	2.45
A	8.15	G	1.95
O	8.05	Y	1.95
N	7.15	P	1.96
R	6.85	W	1.55
I	6.35	B	1.45
S	6.15	V	0.95
H	5.25	K	0.45
D	3.75	X	0.15
L	3.35	J	0.15
F	2.95	Q	0.15
C	2.75	Z	0.05

Figure 1: General frequency of English characters in decreasing order

Therefore, if we are given the following ciphertext:

```
ofobiyxocryevnkvcyexnobcdkxndrovswsdkdsyxcypmbizdyqbkz
rikckdyyvgroxeconsxmyxtexmdsyxgsdrcyvsnzbyqbkwwsxqzbkm
dsmockxnpsbwwkdrowkdsmkvmyxtomdebocsdmkxlorsqrvioppomd
sforygofobspwscecondrobowkilonsckcdobyecmyxcoaeoxmocsx
mvensxqwkccnkdkdropdybcobfobrsqrtkmusxq
```

We would first construct a table of the character present in the text and their respective frequencies, as so:

Character	o	s	c	k	d	x	y	b	m	r	n	e	w	v	q	p	i	f	z	g	t
Frequency	28	21	19	19	19	18	17	15	14	13	10	9	9	8	7	6	5	4	4	3	3

Figure 2: Frequency of English characters in the ciphertext

Then we would map the most frequent characters to each other. Clearly, the character “o” appears to represent the character “e”. Based on our knowledge of the algorithm, we know the key is used as a shift for the integer values of each character. A quick glance at the ASCII character table reveals that e = 101 and o = 111 as integers. Therefore, we can conclude that the key used to encrypt this plaintext is the number 10.

A quick check reveals that we were indeed correct.

```
everyoneshouldalsounderstandthelimitationsofcryptograp
hyasatoolwhenusedinconjunctionwithsolidprogrammingprac
ticesandfirmmathematicalconjecturesitcanbehighlyeffect
ivehoweverifmisusedtheremaybedisasterousconsequencesin
cludingmassdatatheftorserverhighjacking
```

With proper punctuation and capitalization the plaintext becomes:

```
Everyone should also understand the limitations of cry
ptography as a tool. When used in conjunction with sol
id programming practices and firm mathematical conject
ures, it can be highly effective. However, if misused,
there may be disasterous consequences, including mass
data theft or server highjacking.
```

1.3 Advantages and Disadvantages of the Caesar Cipher

At this point, you may be asking yourself why the Caesar cipher would ever be considered a viable method of encrypting data, considering we have been able to break it quite easily using a ciphertext only attack.

Although this cipher is very deeply flawed, it still has certain advantages which make it practical in certain situations.

For example, if speed and memory are your main concerns, and your plaintext only has to be superficially secure, then this cipher is one of your best options. The Caesar cipher has a time complexity of $\Theta(1)$ and a memory complexity of $\Theta(1)$. This means that the cipher's speed and memory usage stay within a constant range, and do not grow (or shrink) in relation to the size of the cipher's input. This is because the cipher operates on only one character (or byte) at a time, and performs the same constant time operation on each byte.

Furthermore, the Caesar cipher may also be practical in situations where only a small amount of plaintext is being encrypted. The statistical analysis which was used is only relevant to plaintexts of sufficient length. It has been shown that highly competent cryptanalysts can break the Caesar cipher using only 25 English characters of plaintext. Therefore, the Caesar cipher might be practical for messages shorter than 25 characters.

2 Block Ciphers

Block ciphers, unlike substitution ciphers, operate on several bytes at a time. This ensures that redundancies in the data are not easy to analyse using statistical methods.

The two basic methods of obfuscating redundancies in plaintext messages are confusion and diffusion. Confusion obscures the relationship between the plaintext and the ciphertext. This can be most simply achieved through substitution, as was done in the Caesar cipher. Diffusion, on the other hand, disperses the redundancy in the plaintext over the ciphertext. A simple example of this is a permutation box, where the position of the input bits are swapped with each other.

Substitution ciphers can only use confusion based techniques, whereas block ciphers are capable of using both confusion and diffusion based techniques. This is why modern block ciphers are generally more secure than modern substitution ciphers.

The block cipher we will be analyzing is a heavily modified version of the GHOST cipher used by the former Soviet Union as encryption standard. The cipher operates on 64 bits of input and accepts a 256 bit key.

Algorithm 2 GHOST cipher

```
$b \leftarrow getBlock()$
$i \leftarrow i + k$
if $i > 26$ then
    $i \leftarrow i - 26$
end if
return $i$
```

2.1 Construction of Block Ciphers

Before we begin the analysis of GHOST, let us briefly look at some important ideas behind the design and construction of block ciphers.

2.1.1 Feistel Networks

Most blocks ciphers, including this version of GHOST, are Feistel networks. In Feistel networks, the plaintext is divided into two separate halfs, the right half, R, and the left half, L. Encryption is performed by repeatedly iterating a function, f, over the plaintext, using a different subkey of K each time:

$$\begin{aligned} L_i &= R_{i-1} \\ R_i &= L_i \oplus f(R_i, K_i) \end{aligned}$$

The wonderful property of this construction, is that decryption does not require f to be invertible, it can be as complex as desired, so long as the input to each round can be reconstructed. This can be easily seen:

$$L_{i-1} \oplus f(R_i, K_i) \oplus f(R_i, K_i) = L_{i-1}$$

Therefore, decryption is simply performing encryption with the subkeys in the opposite order.

2.1.2 S-Boxes and P-Boxes

The confusion and diffusion properties of block ciphers come from their substitution boxes and their permutation boxes, abbreviated to S-boxes and P-boxes respectively. S-boxes output a transformation of their input bits, while P-Boxes output a permutation of their input bits.

Some ciphers do not have P-boxes, but rather permute the input or output bits in a different fashion. The GHOST cipher, for example uses an 11-bit left circular shift, rather than a proper permutation box.

S-boxes are by far the most important part of most block ciphers because they are its only non-linear component. If the S-boxes were linear, (or close to linear) the entire cipher would just be one big linear transformation of bits, and hence trivially simple to break.

Therefore, if these boxes are improperly designed, that is, they perform close to linear transformations of bits, the the security of the entire cipher can be compromised.

As we shall soon see, if these boxes are improperly designed, their overarching cipher can be suceptible to attack.

2.2 Linear Cryptanalysis

Suppose you were given a random sequence of bits X and another random sequence of bits Y. Then the probability that the linear expression,

$$a_1X_1 \oplus a_2X_2 \oplus \cdots \oplus a_nX_n \oplus b_1Y_1 \oplus b_2Y_2 \oplus \cdots \oplus b_mY_m = 0 \tag{1}$$

Is exactly $1/2$ when $\{a\}$ and $\{b\}$ are random sequences of bits.

Now suppose that X and Y are the respective sequences of input and output bits of a substitution box. If the S-box is a good one, then for any random sequence $\{a\}$ and $\{b\}$, the probability of (1) being 0 should be less than some $1/2 + \epsilon$, where ϵ is some negligibly small number. However, if there are certain combinations of $\{a\}$ and $\{b\}$ which result in a probability of obtaining 0 significantly greater or significantly less than $1/2$, then the S-box is said to be biased. The linear probability bias (or just bias), is the amount from which the probability of the linear expression deviates from $1/2$. When the bias is positive, the expression is said to be linear and when it is negative it is said to be affine.

Linear cryptanalysis is a known-plaintext attack against block ciphers, which attempts to take advantage of highly probable or improbable linear expressions occurring at key points in the cipher, most importantly in S-boxes. The attack uses these linear expressions to approximate portions of the cipher (accurate to some non-negligible probability) and eventually obtain the key bits. In general, linear cryptanalysis does not reveal the entire key, but rather makes exhaustive search an option by finding several key bits.

For example, suppose given a 4×4 S-box, every combination of the four input bits is inputed and every output is recorded. It is found that the XOR of the first and second bit of input and the fourth bit of the output is equal to 0 exactly $12/16$ times. This can be stated as:

$$X_1 \oplus X_2 \oplus Y_4 = 0$$

Or equivalently:

$$X_1 \oplus X_2 = Y_4$$

With a probability of $3/4$.

Therefore, the probability bias of this expression is $3/4 - 1/2 = 1/4$. Now suppose that the input to this S-box is the plaintext, P, XOR'd with the key

K. Then we know that $(P_1 \oplus K_1) \oplus (P_2 \oplus K_2) = V_4$, where V is the vector of output bits, with a probability $3/4$. Now, if we fix values for the bits K_1, K_2, and then XOR these hypothesized values with the output, we obtain our original plaintext bits.

$$\begin{aligned} P_1 \oplus K_1 \oplus P_2 \oplus K_2 \oplus K_1 \oplus K_2 &= V_4 \oplus K_1 \oplus K_2 \\ P_1 \oplus P_2 &= V_4 \oplus K_1 \oplus K_2 \end{aligned}$$

If we try each combination of K_1 and K_2 over several plaintext ciphertext pairs, we will see one pair of K_1 and K_2 which when added give us the plaintext bits most often, ergo with the highest probability. These values are the values of the first and second bits of the key. The rest of the key can then be acquired by exhaustive search of the remaining key bits.

In this example, the savings were quite trivial because the keyspace was only 2^4, however, in larger keyspaces, the advantages become more significant.

2.3 Cryptanalysis of the GHOST Cipher

Now that we know how linear cryptanalysis works, we can begin analyzing the GHOST cipher. The GHOST cipher is a 8 round Feistel network which uses a different S-box for each round. During the round, the subkey corresponding to that round is XOR'd with the input and that is then circularly left shifted 11 bits.

Each S-box is a 4×4 S-box, meaning it accepts 4 input bits and 4 output bits. The first S-box, for example looks like this:

Input (hex)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Ouput (hex)	4	A	9	2	D	8	0	E	6	B	1	C	7	F	5	3

Figure 3: Substitution Box 1

To figure out which linear expressions (if any) have high or low biases, we need to test every combination of input bits and output bits for each S-box, and find out which match the most or least often. That is, we need to evaluate

$$a_1X_1 \oplus a_2X_2 \oplus a_3X_3 \oplus a_4X_4 \oplus b_1Y_1 \oplus b_2Y_2 \oplus b_3Y_3 \oplus b_4Y_4 = 0$$

Or equivalently,

$$a_1X_1 \oplus a_2X_2 \oplus a_3X_3 \oplus a_4X_4 = b_1Y_1 \oplus b_2Y_2 \oplus b_3Y_3 \oplus b_4Y_4$$

For every possible combination of $a_i = \{0, 1\}$ and $b_j = \{0, 1\}$.

⊕	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	-2	4	-2	-2	0	2	0	4	2	0	2	-2	0	2	0
2	0	0	-2	2	-2	2	0	0	-2	2	0	0	-4	-4	2	-2
3	0	2	-2	0	-4	-2	-2	0	2	0	-4	2	2	0	0	-2
4	0	2	0	-2	2	-4	-2	-4	0	2	0	-2	-2	0	2	0
5	0	0	0	0	0	-4	4	0	0	0	0	0	0	-4	-4	0
6	0	2	-2	0	0	-2	-2	4	2	0	4	2	-2	0	0	2
7	0	4	2	2	-2	2	0	0	2	2	0	-4	0	0	-2	2
8	0	4	2	-2	2	2	0	0	-2	2	0	4	0	0	-2	-2
9	0	-2	2	4	0	-2	-2	0	-2	4	0	2	2	0	0	2
A	0	0	4	0	0	0	-4	0	0	-4	0	0	0	-4	0	0
B	0	-2	0	-2	-2	0	-2	0	0	2	4	-2	2	0	-2	-4
C	0	-2	-2	0	0	2	-2	-4	2	0	0	2	-2	0	-4	2
D	0	0	2	2	-2	-2	0	0	-2	-2	0	0	-4	4	-2	-2
E	0	2	0	2	-2	0	2	-4	0	-2	4	2	2	0	2	0
F	0	0	0	4	4	0	0	0	4	0	0	0	0	0	0	-4

Figure 4: Linear approximation table for S-box 1

This can be done using a linear approximation table. The table shows us the bias of each combination of input and output bits for the S-box. The input and output hex values represent the values of sequences a_i and b_j respectively from right to left. For example, the bias at row `A`, column `2` represents the bias of the combination

$$(1)X_1 \oplus (0)X_2 \oplus (1)X_3 \oplus (0)X_4 = (0)Y_1 \oplus (0)Y_2 \oplus (1)Y_3 \oplus (0)Y_4$$

Or more succinctly,

$$X_1 \oplus X_3 = Y_3$$

As `A` $= 1010,$ `2` $= 0010$ in hex.

By analyzing each S-box, and the way the output of each S-box moves across different S-boxes after each round, we can identify particular chains of linear expressions which will allow us to approximate rounds of the cipher.

Let $U_{i,j}$ and $V_{i,j}$ represents the input and output of the i-th round, at the j-th bit, (where the bits are numbered from left to right, 1 to 32). Further, let $K_{i,j}$ denote the subkey i of the key K at the j-th bit.

Let us begin with the following linear expression which approximates the first round of the cipher with the probability $3/4$.

$$\begin{aligned} V_{1,3} \oplus V_{1,4} &= U_{1,1} \oplus U_{1,2} \oplus U_{1,3} \oplus U_{1,4} \\ &= (P_1 \oplus K_{1,1}) \oplus (P_2 \oplus K_{1,2}) \oplus (P_3 \oplus K_{1,3}) \oplus (P_4 \oplus K_{1,4}) \end{aligned}$$

Note that each S-box maps to a certain range in the output. S-box 1 works for the first four bits, ($U_{i,1} - U_{i,4}$), S-box 2 for the next four ($U_{i,5} - U_{i,8}$) and so forth. Therefore, certain expressions, such as the one above, use only one S-box transformation, as all of the bits belong to one S-box, whereas others use several S-boxes are hence may have different expansions as well as different probabilities.

We then circularly left shift the output 11 bits, and expand $V_{1,3} \oplus V_{1,4}$. We find that

$$U_{2,24} \oplus U_{2,25} = V_{1,3} \oplus V_{1,4} \oplus K_{2,24} \oplus K_{2,25} = V_{2,22} \oplus V_{2,23} \oplus V_{2,24} \oplus V_{2,28}$$

Is true with a probability of $1/2 + 2^2(3/4)(-1/4)(-6/16)$.

Now we expand $V_{2,22} \oplus V_{2,23} \oplus V_{2,24} \oplus V_{2,28}$, once again after the circular left shift:

$$\begin{aligned} U_{3,11} \oplus U_{3,12} \oplus U_{3,13} \oplus U_{3,17} = \\ V_{2,22} \oplus V_{2,23} \oplus V_{2,24} \oplus V_{2,28} \oplus K_{3,11} \oplus K_{3,12} \oplus K_{3,13} \oplus K_{3,17} \end{aligned}$$

These expansions of the output of the S-boxes continues until we reach the output of the seventh S-box. This final expression gives us the approximation of the first seven rounds of the cipher:

$$P_1 \oplus P_2 \oplus P_3 \oplus P_4 \oplus V_{7,3} \oplus V_{7,6} \oplus V_{7,10} \oplus V_{7,25} \oplus V_{7,31} \oplus V_{7,32} \oplus \Sigma_K = 0$$

Or alternatively:

$$P_1 \oplus P_2 \oplus P_3 \oplus P_4 \oplus U_{8,14} \oplus U_{8,20} \oplus U_{8,21} \oplus U_{8,24} \oplus U_{8,27} \oplus U_{8,31} \oplus \Sigma_K = 0$$

Where Σ_K is the XOR of all the keys picked up along the expansion.

Based on the piling up principle, this expression holds with a probability of

$$1/2 + 2^{23-1}(-1/4)^{1}2(1/4)^{8}(-6/16)(6/16)(5/16)$$

or approximately 0.499999832362 if $\Sigma_K = 0$ and 0.500000167638 if $\Sigma_K = 1$. Although this may seem small, it is actually non-negligible in cryptography.

Because the final input bits are divided into 6 different S-boxes, each of which has a 4 bit input, the total number of bits of subkeys we will have to check by exhaustive search is $2^{6\cdot 4} = 2^{2}4$. One of these keys will match our expression the best, and that will give us 24 bits of the last subkey. Although this is not nearly enough to break the entire cipher, it shows that the cipher is certainly breakable, as these procedures can be repeated with other linear expression chains and the last 8 bits of the subkey can be quite easily retrieved. Then each of the previous subkeys can be analyzed in a similar fashion. Finally, we may retrieve either enough key bits to perform a brute force search, or find the entire key itself.

Theoretically, this cipher is quite broken, however, in practice an attack on the scale of what I have just proposed is often infeasible. Based on Matsui's research, the number of plaintexts required to perform this attack is approximately

$$N_L \approx \frac{1}{\epsilon^2}$$

Where ϵ is the bias of our linear expression and N_L is the number of plaintexts required to perform the linear cryptanalysis attack.

In this case, this means that our cipher would require approximately 3.55×10^{13} plaintext ciphertext pairs, just to recover 24 bits of the key. Since each pair is 2×64 bits of data, the total number of memory needed to store this data would be approximately 520 terabytes. Although this large an amount of data storage is not impossible to obtain, it is very expensive and costly to operate.

Note however, that additional memory need not be purchased to perform subsequent linear cryptanalytic attacks, and therefore this is only an upfront cost. After discovering the entire 8-th subkey, the number of plaintexts required to obtain meaningful results will only become small, as there are fewer and fewer biases to account for (everything past the seventh S-box is now completely linear).

2.4 Advantages and Disadvantages of the GHOST Cipher

Clearly, one of the biggest disadvantages of the GHOST cipher (and for that matter most block ciphers) is that the entire cipher relies on the security of the S-boxes and their construction. A good S-box must have good diffusive properties while still being resistant to linear cryptanalysis and other forms of attack (such as differential cryptanalysis). Finding better S-boxes, however is a very difficult task, and there is still much debate in the scientific community regarding how it should be done. It is known that larger S-boxes are stronger than smaller ones, however, these large tables increase the size of these alogrithms considerably. Furthermore, current mathematical techniques can only create S-boxes which are secure against current attacks, whereas randomly selected S-boxes with adequate properties may be more secure against future attacks.

Furthermore, the GHOST cipher (and block ciphers in general) are quite slow in comparison to other kinds of ciphers. For example, on my personal machine, running the modified GHOST cipher took 1192 milliseconds on one run, whereas running the substitution cipher from the previous chapter took only 6 milliseconds! Therefore, is speed takes precedence over security, GHOST (and for that matter block ciphers in general) may not be the best choice.

However, GHOST does have some great benefits. Cryptographically, it is considerably stronger than the substitution cipher we examined in the previous chapter. It is also very simple to implement in software, as it deals mainly with large chunks of data rather than individual bits.

Lastly, although the parameters set for such an attack seem outrageous and infeasible for any normal person, this cipher will be easily broken by more powerful organizations such as governments or criminal organizations, who have the necessary resources to carry out such attacks.

3 Public-Key Ciphers

So far, we have only examined a subset of ciphers called symmetric-key ciphers. These ciphers require both Alice and Bob to know a shared secret key before they can be used and any secure communications can occur.

However, there exist also a set of ciphers which do not require Alice and Bob to know a shared secret key prior to communication. These types of ciphers are called public-key ciphers.

Public-key ciphers usually function using two keys, a public key, K_{pub}, and a private key K_{pri}. Alice sends Bob her private key K_{pub} over a public channel. Bob then encryptes his message using Alice's private key and sends this encrypted message to her. Alice then decryptes the message using her private key, K_{pri}.

The security of these kinds of ciphers assumes two key things. Firstly, that it is very difficult to decrypt Bob's encrypted message without Alice's private key, and secondly, it is very difficult to retrieve Alice's private key from her public key.

Unlike symmetric-key ciphers which usually use information theoretic methods to achieve security, public-key ciphers are based on the intractability of certain mathematical problems, and hence use computational complexity theoretic methods to achieve security. Basically, this means that they rely on the assumption that certain mathematical problems have no efficient solution and leverage that to build strong ciphers.

The cipher we will be examining is a type of public-key cipher, called the elliptic ElGamel cipher. It is based on the difficulty of the elliptic curve discrete logarithm problem. Before we can cryptanalyse this cipher, we will need to cover a little elliptic curve theory.

3.1 Elliptic Curves

Before we can properly describe the cipher we will be analyzing, a little elliptic curve theory is in order.

Elliptic curves are a special kind of equation whose set of solutions have interesting properties.

For our purposes, an elliptic curve can be defined as the set of solutions to the equation:

$$E : Y^2 = X^3 + AX + B$$

Along with the point at infinity $\mathcal{O}$. $\mathcal{O}$ is a special point which we pretend lies on every vertical line. Note that the curves we will be investigating must also satisfy

$$4A^3 + 27B^2 \neq 0$$

To ensure that the curve is non-singular.

3.1.1 Addition Over Elliptic Curves

What makes these curves special is that there is a natural way of "adding" two points on this curve which always results in a third point which is also on the curve! The elliptic curve addition algorithm is as follows:

Algorithm 3 Elliptic curve addition algorithm

```
Ensure: $E \leftarrow Y^2 = X^3 + AX + B$ and $4A^3 + 27B^2 \neq 0$
Require: $P_1, P_2 \leftarrow$ some points on E
 1: if $P_1 = \mathcal{O}$ then return $P_1 + P_2 = P_1$
 2: else
 3:     $P_1 = (x_1, y_1)$
 4:     $P_2 = (x_2, y_2)$
 5:     if $x_1 = x_1$ and $y_1 = -y_2$ then return $P_1 + P_2 = \mathcal{O}$
 6:     else
 7:         $\lambda \leftarrow \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1} & \text{if } P_1 \neq P_2 \\ \dfrac{3x_1^2 + A}{2y_1} & \text{if } P_1 = P_2 \end{cases}$
 8:         $x_3 \leftarrow \lambda^2 - x_1 - x_2$
 9:         $y_3 \leftarrow \lambda(x_1 - x_3) - y_1$
10:         $P_3 = (x_3, y_3)$ return $P_1 + P_2 = P_3$
11:     end if
12: end if
```

For example, consider the elliptic curve $E : Y^2 = X^3 + 8X + 4$, and the points $P_1 = (4, 10), P_2 = (5, 13)$, both of which are on the curve.

If we add the two points together using the algorithm above,

$$\begin{aligned} \lambda &= \frac{13 - 10}{5 - 4} = 3 \\ x_3 &= 3^2 - 4 - 5 = 0 \\ y_3 &= 3(4) - 10 = 2 \\ &P_1 + P_2 = (0, 2) \end{aligned}$$

We obtain the point $P_3 = (0, 2)$, which is also a solution to E!

$$2^2 = 0^3 + 8(0) + 4$$

3.1.2 Elliptic Curves Over Finite Fields

To allow for us to work with elliptic curves in a cryptographic context we need to extend our definition of elliptic curves to include elliptic curves over finite fields. This will allow us to define the mathematically intractable problem which defines the security of this cipher.

Simply, we defined elliptic curves whose points have coordinates in the finite field ("over" the finite field) $\mathbb{F}_p$, with the equation

$$E : Y^2 = X^3 + AX + B \text{ with } A, B \in \mathbb{F}_p \text{ satisfiying } 4A^3 + 27B^2 \neq 0$$

where p is some prime number larger than 3. The points on E with coordinates in $\mathbb{F}_p$ are denoted by

$$E(\mathbb{F}_p) = \{(x, y) : x, y \in \mathbb{F}_p \text{ and } y^2 = x^3 + Ax + B\} \cup \{\mathcal{O}\}$$

The same addition rule we previously defined works here as well, however, because all operations are now in the field $\mathbb{F}_p$, the addition of two points will always yield another point inside $\mathbb{F}_p$! This sort of cyclic property is what this cipher exploits to ensure its security.

For example, consider the addition of the points $P_1 = (0, 2), P_2 = (4, 10)$ on the elliptic curve $E : Y^2 = X^3 + 8X + 4$ over the finite field $\mathbb{F}_11$.

$$\begin{aligned} \lambda &= \frac{10 - 2}{4 - 0} = 2 \pmod{11} \\ x_3 &= 2^2 - 0 - 4 = 0 \pmod{11} \\ y_3 &= 2(0) - 2 = 9 \pmod{11} \\ P_1 + P_2 &= (0, 9) \end{aligned}$$

Although it may not seem obvious,

$$\begin{aligned} 9^2 &\equiv 0^3 + 8(0) + 4 \pmod{11} \\ 81 &\equiv 4 \pmod{11} \end{aligned}$$

The point $(0, 9)$ is indeed a solutioni to E in the field $\mathbb{F}_11$.

3.1.3 Elliptic Curve Discrete Logarithm Problem

As we mentioned earlier, the elliptic curve discrete logarithm problem (ECDLP), is the underlying hard problem upon which ElGamel is based.

The ECDLP is the problem of finding and integer n such that $Q = nP$, where $P, Q \in E(\mathbb{F}_p)$ and the multiplication of n and P is defined as:

$$\underbrace{P + P + P + \cdots + P}_{n \text{ additions over } E} = nP = Q$$

Althougth this may seem simple, remember that the addition occuring over the curve E more complex than normal addition, and that it is being done in the field $\mathbb{F}_p$.

A key point to notice here is that if there exists one n then there must, through the cyclic nature of the addition rule, be an infinite number of n which also solve $Q = nP$. As we shall soon see, this can be exploited to find n.

3.2 Elliptic ElGamel

Now that we have an understanding of the mathematics behind elliptic curves and the ECDLP, we can describe the ElGamel cipher in its entirety.

ElGamel works as follows. A large prime p, an elliptic curve E over $\mathbb{F}_p$ and a point $P \in E(\mathbb{F}_p)$ are chosen by a trusted party. Alice then chooses a private key, n_A, and computes $Q_A = n_A P$ in $E(\mathbb{F}_p)$. She then published Q_A as her public key. If Bob wants to send a message to Alice, he chooses a plaintext $M \in E(\mathbb{F}_p)$ and a temporary key k. He then computes $C_1 = kP$ and $C_2 = M + kQ_A$ in $E(\mathbb{F}_p)$ and sends (C_1, C_2) to Alice. Alice then computes $C_2 - n_A C_1$ in $E(\mathbb{F}_p)$ to obtain M, given that:

$$C_2 - n_A C_1 = (M + kQ_A) - n_A(kP) = M + k(n_A P) - n_A(kP) = M$$

3.3 Cryptanalysis of the ElGamel Cipher

We know that if we solve the underlying hard problem of the ElGamel cipher namely, ECDLP, then we can quite easily break the entire cipher. However, if we had to check every single value of n until we find a match, which in the worst case, would be unfeasible.