# University of Technology
# الجامعة التكنولوجية

## Computer Science Department
## قسم علوم الحاسوب

# Cryptanalysis
# تحليل الشفره


## Prof.Dr.Hala Bahjat AbdulWahab
## أ.د.هاله بهجت عبدالوهاب


Prof.Dr.Hala Bahjat Abdulwahab

cs.uotechnology.edu.iq

# *Cryptanalysis*

## ***Introduction:***

**** Cryptanalysis*** is the science of making encrypted data unencrypted.

A cryptographer will use cryptography to convert plaintext into ciphertext and a cryptanalyst will use cryptanalysis to attempt to turn that ciphertext back into plaintext. Both the cryptographer and the cryptanalyst are cryptologists. Cryptography and cryptanalysis are the two sides of cryptology.

It without use of the key. The other side of cryptography, cryptanalysis is used to break codes by finding weaknesses within it. In addition to being used by hackers with bad intentions, cryptanalysis is also often used by the military. Cryptanalysis is also appropriately used by designers of encryption systems to find, and subsequently correct, any weaknesses that may exist in the system under design.

There are several types of attacks that a cryptanalyst may use to break a code, depending on how much information they have. A ciphertext-only attack is one where the cryptanalyst has a piece of ciphertext (text that has already been encrypted), with no plaintext (unencrypted text). This is probably the most difficult type of cryptanalysis, and calls for a bit of guesswork. In a known-plaintext attack, the cryptanalyst has both a piece of ciphertext and the corresponding piece of plaintext.

Other types of attacks may involve trying to derive a key through trickery or theft. The "man-in-the-middle" attack is one example. In this attack, the cryptanalyst places a piece of surveillance software in between two parties that communicate. When the parties' keys are exchanged for secure communication, they exchange their keys with the attacker instead of each other.

Prof.Dr.Hala Bahjat Abdul wahab

The ultimate goal of the cryptanalyst however, is to derive the key, so that all ciphertext can be easily deciphered. A brute-force attack is one way of doing so. In this type of attack, the cryptanalyst tries every possible combination until the correct key is identified. Although using longer keys make the derivation less statistically likely to be successful, faster computers, continue to make brute-force attacks feasible. Networking a set of computers together in a grid, combines their strength; their cumulative power can be used to break long keys. The longest keys used, 128-bit keys, remain the strongest, and less likely to be subject to a brute-force attack.

At its core, cryptanalysis is a science of mathematics, probability and fast computers; cryptanalyst's also usually require some persistence, intuition, guesswork and some general knowledge of the target.

Cryptanalysis also has an interesting historical element; the famous Enigma machine, used by the Germans to send secret messages, was ultimately cracked by members of the Polish resistance and transferred to the British.

There are two general approaches in order to attacking a conventional encryption scheme:

- **Cryptanalysis:** Cryptanalytic attacks rely on the nature of the algorithm plus perhaps some knowledge of the general characteristics of the plaintext or even some sample plaintext-cipher text pairs. This type of attack exploits the characteristics of the algorithm to attempt to deduce a specific plaintext or to deduce the key being used. If the attack succeeds in deducing the key, the effect is catastrophic: All future and past messages encrypted with that key are compromised.
- **Brute-force attack:** The attacker tries every possible key on a piece of ciphertext until an intelligible translation into plaintext is obtained. On average, half of all possible keys must be tried to achieve success.

Prof.Dr.Hala Bahjat Abdul wahab

Table (1) summarizes the various types of cryptanalytic attacks, based on the amount of information known to the cryptanalyst. The most difficult problem is presented when all that is available is the *ciphertext only*. In some cases, not even the encryption algorithm is known, but in general, we assume that the opponent does know the algorithm used for encryption. One possible attack under these circumstances if the brute-force approaches of trying all possible keys.

If the key space is very large, this becomes impractical. Thus, the opponent must rely on an analysis to the cipher text itself, general applying various statistical tests to it. To us this approach, the opponent must have some general idea of the type of plaintext that is concealed, such as English or French text, a Window EXE file, a Java source listing, an accounting file, and so on.

Table (1): Types attacks on encrypted messages.

| ***Type of Attack*** | ***Known of Cryptanalyst*** |
|---|---|
| **Ciphertext only** | • Encryption algorithm.<br>• Ciphertext to be decoded. |
| **Known plaintext** | • Encryption algorithm.<br>• Ciphertext to be decoded.<br>• One or more plaintext-ciphertext pairs formed with the secret key. |
| **Chosen plaintext** | • Encryption algorithm.<br>• Ciphertext to be decoded.<br>• Plaintext message chosen by cryptanalyst, together with its corresponding cipher text generated with secret key. |

Hala Bajat Abdul wahab

| | |
|---|---|
| **Chosen cipher text** | • Encryption algorithm<br>• Cipher text to be decoded<br>• Purported cipher text chosen by cryptanalyst, together with its corresponding decrypted plain generated with the secret key |
| **Chosen text** | • Encryption algorithm<br>• Cipher text to b decoded<br>• Plaintext massage chosen by cryptanalyst, together with its corresponding cipher text generated with the secret key.<br>• Purported cipher text chosen by cryptanalyst, together with its corresponding decrypted plaintext generated with the secret key |

Table (1) lists two other types of attacks: chosen ciphertext and chosen text. These are less commonly employed as cryptanalytic techniques but are possible avenues of attack.

1- Cipher text – only attack: the cryptanalyst has the cipher text of several message , all of which have been encrypted using the same encryption algorithm, the cryptanalyst job is to recover the plain text of many message as possible, or better yet to deduce the key ( or keys) used to encrypted the message. In order to decrypted other message encrypted with the same keys.

Given: $C_1=E_k(p_1)$, $C_2=E_k(p_2)$,….. $C_i=E_k(p_i)$

Deduce: either $p_1, p_2, p_i, k$; or an algorithm.

To infer $p_{i+1}$ from $C_i=E_k(p_{i+1})$

Prof.Dr.Hala Bahjat Abdul wahab

2- Known – plain text attack: the cryptanalyst has access not only to the cipher text of several message, but also to the plain text of those message. His job is to deduce the key (or keys) used to encrypt the message or an algorithm to decrypt any new message encrypted with the same key, (or keys).

Given: $P_1, C_1=E_k(p_1)$, $P_2, C_2=E_k(p_2)$,….. $P_i, C_i=E_k(p_i)$

Deduce: either k or an algorithm.

To infer $P_{i+1}$ from $C_i=E_k(P_{i+1})$

3- chosen– plain text attack: the cryptanalyst not only has access to the cipher text and associated plain text for several message. But be also chooses the plain text that gets encrypted.

This is more powerful than a known-plain text attack, because the cryptanalyst can choose specified plaintext blocks to encrypted, ones that might more information about the key. His job is to deduce the key (or keys) used to encrypt the message or an algorithm to decrypt any new message encrypted with the same key (or keys).

Given: $P_1, C_1=E_k(p_1)$, $P_2, C_2=E_k(p_2)$,….. $P_i, C_i=E_k(p_i)$

where the cryptanalyst gets to choose $p_1$, $p_2$, $p_i$

Deduce: either k or an algorithm.

To infer $P_{i+1}$ from $C_i=E_k(P_{i+1})$

Other

4- chosen– cipher text attack.

5- Adaptive-chosen-plaintext attack: this is a special case of a chosen-plaintext attack.

Note only can the cryptanalyst choose the plaintext that is encrypted , but he can also modify his choice based on the result of pervious encryption- in a chosen-plain text attack,a cryptan-alyst might just be able to choose one large block of plain text to be encrypted ; in an adaptive-chosen-plaintext attack he can choose

Prof.Dr.Hala Bahjat Abdul wahab

a smaller block of plaintext and then choose another based on the result of the first, and so forth.

Only relatively weak algorithms fail to withstand a ciphertext-only attack. Generally, an encryption algorithm is designed to withstand a known-plaintext attack.

Two more definitions are worth to be noted. An encryption scheme is **unconditionally secure** if the ciphertext generated by the scheme does not contain enough information to determine uniquely the corresponding plaintext, no matter how much ciphertext is available, that is no, matter how much time an opponent has, it is impossible for him or her to decrypt the ciphertext, simply because the required information is not there. With the exception of a scheme known as the one-time pad, there is no encryption algorithm that is unconditionally secure. Therefore, all that the uses of an encryption algorithm can strive for is an algorithm that meets one or both of the following criteria:

- The cost of breaking the cipher exceeds the value of the encrypted information.
- The time required to break the cipher exceeds the useful lifetime of the information.

An encryption scheme is said to be **computationally secure** if the foregoing two criteria are met. The rub is that it is very difficult to estimate the amount of effort required to cryptanalyze ciphertext successfully. Table (2) shows how much time is involved for various key spaces. Results are shown for four binary key sizes.

For each key size, the result is shown assuming that it takes $1\mu s$ to perform a single decryption, which is a reasonable order of magnitude for today's machines. The final column of table (2) considers the results for a system that can process 1 million keys per microsecond.

Prof.Dr Hala Bahjat Abdul wahab

Table (2): Average time required for exhaustive key search.

| ***Key Size (bits)*** | ***Number of Alternative Keys*** | ***Time required at 1 encryption / μs*** | ***Time required at $10^6$ encryptions / μs*** |
|---|---|---|---|
| 32 | $2^{32} = 4.3 \times 10^9$ | $2^{31}\ \mu s = 35.8$ **minutes** | 2.15 **milliseconds** |
| 56 | $2^{56} = 7.2 \times 10^{16}$ | $2^{55}\ \mu s = 1142$ **years** | 10.01 **hours** |
| 128 | $2^{128} = 3.4 \times 10^{38}$ | $2^{167}\ \mu s = 5.9 \times 10^{36}$ **years** | $5.9 \times 10^{30}$ **years** |
| **26 characters (permutation)** | $26! = 4 \times 10^{26}$ | $2 \times 10^{26}\ \mu s = 6.4 \times 10^{12}$ **years** | $6.4 \times 10^{6}$ **years** |

Prof.Dr.Hala Bahjat Abdul Wahab

# *Transposition Ciphers*

- ***intorduction***

Transposition ciphers jumble the letters of the message in a way that is designed to confuse the attacker, but can be unjumbled by the intended recipient. The concept of transposition is an important one and is widely used in the design of modern ciphers, as will be seen in subsequent chapters. Notethat the key must provide sufficient information to unscramble the ciphertext.

*** Scytale**

One of the earliest recorded uses of cryptography was the Spartan scytale(circa 500 B.C.). **A** thin strip of parchment was wrapped helically around a cylindrical rod and the message was written across the rod, with each letter on a successive turn of the parchment. The strip was unwound and delivered to the receiver. The message could then be decrypted with the use of an identical cylindrical rod. To anyone who intercepted the message, and did not understand the encryption technique, the message would appear to be a jumble of letters. **A** clever cryptanalyst with access to a number of rods of various diameters will soon recover the plaintext.

For the scytale cipher, which is an example of a transposition cipher, the key is the rod (or its diameter). This is a very weak cipher since the system could be easily broken by anyone who understands the encryption method.

Prof.Dr.Hala Bahjat Abdul wahab

- **Scytale**

bdul wahab

# Example:

# Encrypting

Suppose the rod allows one to write 4 letters around in a circle and 5 letters down the side of it. The plaintext could be: "Help me I am under attack"

To encrypt one simply writes across the leather...

```
_____________________________________________________________
    |   |   |   |   |   |   |
    |   | H | E | L | P | M |
    |___| E | I | A | M | U |__
        | N | D | E | R | A |   |
        | T | T | A | C | K |   |
        |   |   |   |   |   |   |
_____________________________________________________________
```

so the ciphertext becomes, "HENTEIDTLAEAPMRCMUAK" after unwinding.

# Decrypting

To decrypt all one must do is wrap the leather strip around the rod and read across. The ciphertext is: "HENTEIDTLAEAPMRCMUAK" Every fifth letter will appear on the same line so the plaintext becomes

```
HELPM...return to the beginning once the end is reached
...EIAMUNDERATTACK
```

Insert spaces and the plaintext is revealed: "Help me I am under attack"

***Columnar Transposition.***

Suppose we have plaintext **SEETHELIGHT** and we want to encrypt this using a columnar transposition cipher. We first put the plaintext into the rows of an array of some given dimension. Then we read the ciphertext out of the columns. The key consists of the the number of columns in the array. For example, suppose we choose the key to be four, which means that we write the plaintext in four columns as

$$\begin{bmatrix} S & E & E & T \\ H & E & L & I \\ G & H & T & X \end{bmatrix},$$

Where the final **X** is used as to fill out the array. The ciphertext is then read from the columns, which in this case yields **SHGEEHELTTIX.** The intended recipient, who knows the number of columns, can put the ciphertext into an appropriate-sized array and read the plaintext out from the rows. Not surprisingly, a columnar transposition is not particularly strong. To perform a ciphertext only attack on this cipher, we simply need to test all possible decrypts using c columns, where c is a divisor of the number of characters in the ciphertext.

***Keyword Columnar Transposition***

The columnar transposition cipher can be strengthened by using a keyword, where the keyword determines the order in which the columns of ciphertext are transcribed. We refer to this as a keyword columnar transposition cipher.

For example, consider encrypting the plaintext **CRYPTOISFUN** using a keyword columnar transposition cipher with keyword **MATH,** again using four columns. In this case, we get the array

$$\begin{array}{c} \begin{matrix} M & A & T & H \end{matrix} \\ \hline \begin{bmatrix} C & R & Y & P \\ T & O & I & S \\ F & U & N & X \end{bmatrix} \end{array}.$$

Prof.Dr.Hala Bahjat Abdul wahab

The ciphertext is read from the columns in alphabetical order (as etermined by the keyword), so that, in this example, the ciphertext is

**RO UPSXCTFYIN**

Is it possible to conduct a ciphertext-only attack on a keyword columnar transposition cipher? It is certainly not as straightforward as attacking a non-keyword columnar cipher. Suppose we obtain the ciphertext ,

**VOESA IVENE MRTNL EANGE WTNIM HTMEE ADLTR NISHO DWOEH**

Which we believe was encrypted using a keyword columnar transposition. Our goal is to recover the key and the plaintext. First, note that there are 45 letters in the ciphertext. Assuming the array is not a single column or row, the array could have any of the following dimensions: 9 x 5. 5 x 9. 15 x **3**
or **3** x 15. Suppose that we first try a 9 x ***5*** array. Then we have the ciphertext array in Table 1.1. We focus our attention on the top row of the array in Table 1.1. If we permute the columns as shown in Table 1.2, we see the word **GIVE** in the first row and we see words or partial words in the other rows. Therefore, we have almost certainly recovered the key.

Prof.Dr.Hala Bahjat Abdul wahab

Table 1.1: Ciphertext Array

| 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| V | E | G | M | I |
| O | M | E | E | S |
| E | R | W | E | H |
| S | T | T | A | O |
| A | N | N | D | D |
| I | L | I | L | W |
| V | E | M | T | O |
| E | A | H | R | E |
| N | N | T | N | H |

Table 1.2: Permuted Ciphertext Array

| 2 | 4 | 0 | 1 | 3 |
|---|---|---|---|---|
| G | I | V | E | M |
| E | S | O | M | E |
| W | H | E | R | E |
| T | O | S | T | A |
| N | D | A | N | D |
| I | W | I | L | L |
| M | O | V | E | T |
| H | E | E | A | R |
| T | H | N | N | N |

Prof.Dr. wahab

This method is somewhat ad hoc, but the process could be automated, provided we can automatically recognize likely plaintexts. In this example, we have recovered the encryption key 24013 and the plaintext is

**GIVE ME SOMEWHERE TO STAND AND I WILL MOVE THE EARTH.**

There are many ways to systematically mix the letters of the plaintext.

For example, we can strengthen the columnar transposition cipher by allowing the permutation of columns and rows. Since two transpositions are involved, this is known as a double transposition cipher, which we briefly describe next.

***Double Transposition Cipher***

To encrypt with a double transposition cipher, we first write the plaintext into an array of a given size and then permute the rows and columns according to specified permutations. For example, suppose we write the plaintext **ATTACKATDAWN** into a **3** x 4 array:

$$\begin{bmatrix} A & T & T & A \\ C & K & A & T \\ D & A & W & N \end{bmatrix}.$$

Now if we transpose the rows according to (0,1,2) →(2,1,0) and then transpose the columns according to (0,1,2,3) → (3,1,0,2), we obtain

$$\begin{bmatrix} A & T & T & A \\ C & K & A & T \\ D & A & W & N \end{bmatrix} \longrightarrow \begin{bmatrix} D & A & W & N \\ C & K & A & T \\ A & T & T & A \end{bmatrix} \longrightarrow \begin{bmatrix} N & A & D & W \\ T & K & C & A \\ A & T & A & T \end{bmatrix}.$$

The ciphertext is read directly from the final array:

**NADWTKCAATAT.**

For the double transposition, the key consists of the size of the matrix and the row and column permutations. The recipient who knows the key can simply put the ciphertext into the appropriate sized matrix and undo the permutations to recover the plaintext.If Trudy happens to know the size of the matrix used in a double transposition, she can insert the ciphertext into a matrix of the appropriate size. She can then try to unscramble the columns to reveal words (or partial words).

Once the column transposition has been undone, she can easily unscramble the rows; This attack illustrates the fundamental principle of divide and conquer. That is, Trudy can recover the double transposition key in parts, instead of attacking the entire key all at Once. There are many examples of divide and

Prof.Dr.Hala Mahmoud Abdul wahab

conquer attacks throughout The remainder of this book. In spite of the inherent divide and conquer attack, the double transposition cipher is relatively strong---at least in comparison to many other classic cipher. The interested reader is directed to for a thorough cryptanalysis of the double transposition.

# Substitution Ciphers

## Introduction

Like transposition, substitution is a crucial concept in the design of modern ciphers. in fact, Shannon's [133] two fundamental principles for the design of symmetric ciphers are confusion and diflusion, which, roughly, correspond to the classic concepts of substitution and transposition, respectively. These are still the guiding principles in the design of symmetric ciphers. In this section we discuss several classic substitution ciphers. We high light, some of the clever techniques that can be brought to bear to attack such ciphers.

### Caesar's Cipher

in 50 R.C., Gaius Julius Caesar described the use of a specific cipher that, goes by the name of Caesar's c2pher.l In Caesar's cipher, encryption is accomplished by replacing each plaintext letter with its corresponding "shiftby- three" letter, that is, **A** is replaced by D, B is replaced by E, C is replaced by F, and so on. At the end of the alphabet, a wrap around occurs, with X replaced by **A, Y** replaced by B and Z replaced by C. Decryption is accomplished by replacing each ciphertext letter with its corresponding left-shift-by-three letter, again, taking the wrap around into account. Suppose we assign numerical values 0 , 1 , . . . ,25 to the letters **A,** B, . . . , Z, respectively, Let pi be

Prof.Dr Hala Bahjat Abdul wahab

the ith plaintext letter of a given message, and ci thecorresponding ith ciphertext letter. Then Caesar's cipher can be mathematically stated as ci = ***pi*** + **3** (mod 26) and, therefore, pi = ci - **3** (mod 26). In Caesar's cipher, the key is ***"3",*** which is not very secure, since there is only one key-anyone who knows that the Caesar's cipher is being used can immediately decrypt the message.

## Simple Substitution

A simple substitution (or mono-alphabetic substitution) cipher is a generalization of the Caesar's cipher where the key can be any permutation of the alphabet. For the simple substitution, there are 26! = 288 keys available. This is too many keys for any attacker to simply try them all, but even with this huge number of keys, the simple substitution cipher is insecure. Before we discuss the attack on the simple substitution, we consider a few special types of related ciphers that have been used in the past.

## Poly-alphabetic Substitution

During the Renaissance, the first poly-alphabetic substitution cipher was invented by one Leon Battista Alberti (1404-1472). Such a cipher is essentially a variable simple substitution cipher, that is, a different substitution alphabet is used for different parts of the message. In Alberti's cipher, this was accomplished by use of a device that included an inner and outer cipher wheel with the alphabet written in

Prof Dr.Hala Bahjat Abdul wahab

particular ways on each wheel. The inner wheel freely rotated allowing the two alphabets to be aligned in any fashion, with each alignment generating a different (simple) substitution. As the message was encrypted, differing substitution alphabets could be used, as determined by both parties in advance, or as specified within the message itself. In his book ***Traict6*** des Chaffres, Blaise de Vigenkre (1585) discusses a poly-alphabetic substitution that uses a 26 x 26 rectangular array of letters. The first row of the array is **A,** B, C, . . . , Z, and each succeeding row is a cyclic left shift of the preceding one. A keyword can then be used to determine which of the cipher alphabets to use at each position in the text. In this way, all "shift-by-n" simple substitutions are readily available for use. The Vigenkre cipher, and its cryptanalysis, is discussed below.

## Affine Cipher

An ajJine cipher is a simple substitution where *ci* = api + b (mod 26). Here,the constants a and b are integers in the range 0 to 25 (as are p , and ci).To decrypt uniquely--always a nice feature for a cipher system--we must have gcd(a, 26) = 1. Consequently, there are 26.4(26) = **312** affine ciphers for the English language, where **4** is the Euler-phi function (see the Appendix fora definition of the **4** function). The decryption function for the affine cipher is pi = *aP1(ci* - b) (mod 26), where aa-l = 1 (mod 26), that is, up1 is the multiplicative inverse of a, modulo 26.Affine ciphers are weak for several reasons, but the most obvious problem is that they have a small keyspace. A ciphertext only attack can be performed by conducting a brute force search of all 312

Prof Dr.Hala Bahjat Abdul wahab

possible key pairs ( a , b). This attack is trivial, provided we can recognize the plaintext when we see it (or, better yet, automatically test for it).

**Simple Substitution Cryptanalysis**

Trying all possible keys is known as an exhaustive key search, and this attack is always an option for Trudy. If there are N possible keys, then Trudy will,on average, need to try about half of these, that is; N/2 of the keys, before she can expect to find the correct key. Therefore, the first rule of cryptography is that any cipher must have a large enough keyspace so that an exhaustive search is impractical. However, a large keyspace does not ensure that a cipher is secure. To see that this is the case, we next consider an attack that will work against any simple substitution cipher and, in the general case, requires far less work than an exhaustive key search. This attack relies on the fact that statistical information that is present in the plaintext language “leaks” through a simple substitution .Suppose we have a reasonably large ciphertext message generated by a compiled from a 7834-letter sample of written English. By simply computing letter frequency counts on our ciphertext, we can make educated guesses as to which plaintext letters correspond to some of the ciphertext letters.For example, the most common ciphertext letter probably corresponds to plaintext **E.** We can obtain additional statistical information by making use of digraphs (pairs of letters) and common trigraphs (triples). This type of statistical attack on a simple substitution, is very effective. After a few letters

Prof.Dr.Hala Bahjat Abdul wahab

have been guessed correctly, partial words will start to appear and the cipher should then quickly unravel.

English Letter Frequencies as Percentages

| Letter | Relative Frequency | Letter | Relative Frequency |
|---|---|---|---|
| A | 8.399 | N | 6.778 |
| B | 1.442 | O | 7.493 |
| C | 2.527 | P | 1.991 |
| D | 4.800 | Q | 0.077 |
| E | 12.15 | R | 6.063 |
| F | 2.132 | S | 6.319 |
| G | 2.323 | T | 8.999 |
| H | 6.025 | U | 2.783 |
| I | 6.485 | V | 0.996 |
| J | 0.102 | W | 2.464 |
| K | 0.689 | X | 0.204 |
| L | 4.008 | Y | 2.157 |
| M | 2.566 | Z | 0.025 |

## Vigenere Cipher

Recall that a poly-alphabetic substitution cipher uses multiple simple substitutions to encrypt a message. The Vigenkre cipher is a classic poly-alphabetic substitution cipher. The World War I1 cipher machines discussed in Chapter 2 are more recent examples of poly-alphabetic substitutions. In the Vigenkre cipher, a key of the form K = (ko, k l ; . . . , ***k n - l ) ,*** where each ki E {0,1,. . . ,25}, is used to encipher the plaintext. Here, each ***ki***represents a particular shift of the alphabet.

***To encrypt a message, CZ = Pi + ki (mod n) (mod 26) and***

***To decrypt Pi = CZ - ki (mod n) (mod 26)***

For example, suppose K = (12,0,19,7), which corresponds to the keyword **MATH** (since **M** corresponds to a shift of 12, **A** corresponds to a shift of 0, and so on). Using this keyword, the the plaintext

Prof Dr.Hala Bahjat Abdul wahab

**SECRETMESSAGE** is encrypted as **EEVYQTFLESTNQ.** Next, we cryptanalyze the Vigenkre cipher. But first, note that a polyalphabetic substitution (such as the VigenBre cipher) does not preserve plaintext letter frequencies to the same degree as a mono-alphabetic substitution. Furthermore, if the number of alphabets is large relative to the message size,the plaintext letter frequencies will not be preserved at all. Therefore, the generic simple substitution attack discussed above will not work on a polyalphabetic substitution.However, the VigenBre cipher is vulnerable to a slightly more sophisticated statistical attack. To see how this works, first consider a VigenBre cipher with a small keyword. Suppose that the following ciphertext was created using a VigenBre cipher with a three-lettered keyword:

**RLWRV MRLAQ EDUEQ QWGKI LFMFE XZYXA QXGJH FMXKM QWRLA LKLFE LGWCL SOLMX RLWPI OCVWL SKNIS IMFES JUVAR MFEXZ CVWUS MJHTC RGRVM RLSZS MREFW XZGRY RLWPI OMYDB SFJCT CAZYX AQ.** (1.1)

To recover the key and decrypt the message, we can make use of the fact that the ciphertext is composed of three simple substitutions. To accomplish t,his,we tabulate the letter frequencies for the sets SO = {co,c:~c,g ,. . . }, 5.1 = {Q, c 4 , ~ 7 , .. . }, and 5.2 = {c2,c g, ex.. . . }.where c, is the ith ciphertext letter. Doing so, we obtain the results in Tables 1.4, 1.5, and 1.6, respectively.

Prof.Dr.Hala Bahjat Abdul wahab

Table 1.4: Letter Frequencies in $S_0$

| Letter | R | Q | U | K | F | E | Y | J | M | L | G | P | C | N | I | Z | W | B |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Frequency | 10 | 4 | 3 | 1 | 2 | 3 | 2 | 3 | 3 | 4 | 2 | 2 | 4 | 1 | 1 | 1 | 1 | 1 |

Table 1.5: Letter Frequencies in $S_1$

| Letter | L | V | E | W | I | M | X | Q | K | S | H | R | Y | C | A |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Frequency | 6 | 5 | 4 | 2 | 4 | 4 | 7 | 1 | 1 | 6 | 1 | 2 | 1 | 1 | 1 |

From the So ciphertext in Table 1.4, we might reasonably guess that ciphertext R corresponds to plaintext E. T, N, 0, R, I. A or S. which gives us

Table 1.6: Letter Frequencies in $S_2$

| Letter | W | M | A | D | Q | G | L | F | Z | K | C | O | X | S | J | T | Y |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Frequency | 6 | 4 | 5 | 2 | 1 | 3 | 3 | 5 | 4 | 2 | 1 | 3 | 1 | 2 | 1 | 2 | 1 |

candidate values for ko, namely ko E {13,24,4,3,0,9,17,25}. Similarly, for set S1, ciphertext **X** niight correspond to plaintext **E, T, N, 0, R, I, A** or **S,**from which we obtain likely values for k l , and from set Sz, ciphertext W likely correspond to plaintext **E, T, N, 0, R, I, A** or **S.** The corresponding likely keyword letters are tabulated in Table 1.7. candidate values for ko, namely ko E {13,24,4,3,0,9,17,25}. Similarly, for set S1, ciphertext **X** niight correspond to plaintext **E, T, N, 0, R, I, A** or **S,** from which we obtain likely values for k l , and from set Sz, ciphertext W likely correspond to plaintext **E, T, N, 0, R, I, A** or **S.** The corresponding likely keyword letters are tabulated in Table 1.7.

Table 1.7: Likely Keyword Letters

| $k_0$ | $k_1$ | $k_2$ |
|---|---|---|
| N | T | S |
| Y | E | D |
| E | K | J |
| D | J | I |
| A | G | F |
| | | |

Prof.Dr.Hala Bany

The conibinations of likely keyword letters in Table 1.7 yield 83 = 2’ putative keywords. By testing each of these putative keyword on the first few letters of the ciphertext, we can easily determine which, if any, is the actual keyword. For this example, we find that (ko, k l , k2) = (24,4,18), which corresponds to **YES,** and the original plaintext is

**THE TRUTH IS ALWAYS SOMETHING THAT IS TOLD, NOT SOMETHING THAT IS KNOWN. IF THERE WERE NO SPEAKING OR WRITING, THERE WOULD BE NO TRUTH ABOUT NYTHING.THERE WOULD ONLY BE WHAT IS.**

This attack provides a significant shortcut, as conipared to trying all possible 263 **M** 214 keywords. Knowing the length of the keyword used in a Vigenkre cipher helps greatly in the cryptanalysis. If the keyword is known, and the message is long enough, we can simply perform letter frequency counts on the associated sets of ciphertext to begin solving for the plaintext. However, it is not so obvious how to determine the length of an unknown keyword. Next, we consider two Friederich W. Kasiski (1805-1881) was a major in the East Prussian infantry regiment and the author of the cryptologic text Die Geheimschriflen und die Dechiger-kunst. Kasiski developed a test

Prof.Dr Hala Bahjat Abdul wahab

(amazingly, known as the Kasiski Test), that can sonietimes be used to find the length of a keyword used in a cipher such as the Vigenkre. It relies on the occasional coincidental alignment of letter groups in plaintext with the keyword. To attack a periodic cipher using the Kasiski Test, we find repeated letter groups in the ciphertext arid tabulate the separations between them. The greatest common divisor of these separations (or a divisor of it) gives a possible length for the keyword.For example, suppose we encrypt the plaintext

**THECHILDISFATHEROFTHEMAN**

with a Vigenkre cipher using the keyword **POETRY.** The resulting ciphertest is

**IVIVYGARMLMYIVIKFDIVIFRL.**

Notice that the second Occurrence of the ciphertext letters **IVI** begins exactly 12 letters after the first, and the third occurrence of **IVI** occurs exactly six letters after the second. Therefore, it is likely that the length of the keyword is a divisor of six. In this case, the keyword length is exactly six.

## Index of Coincidence

While working at the Riverbank Laboratory, William F. Friedman (1891L 1969) developed the index of coincidence. For a given ciphertext, the index of coincidence I is defined to be the probability that two randomly selected letters in the ciphertext represent, t.he same plaintext symbol. For a given ciphertext, let no, 121,. . . ,*1225* be the respective letter counts of **A,** B, C, . . . , Z in the ciphertext, and set ***71***

Prof.Dr.Hala Bahjat Abdul wahab

= n o + **111** + . . . + r125. Then, theindex of coincidence can be computed as

$$I = \frac{\binom{n_0}{2} + \binom{n_1}{2} + \cdots + \binom{n_{25}}{2}}{\binom{n}{2}} = \frac{1}{n(n-1)} \sum_{i=0}^{25} n_i(n_i - 1). \qquad (1.2)$$

To see why the index of coincidence gives us useful information, first note that the empirical probability of randomly selecting two identical letters from a large English plaintext is

$$\sum_{i=0}^{25} p_i^2 \approx 0.065,$$

where po is the probability of selecting an **A,** p l is the probability of selecting a B, and so on, and the values of p , are given in Table 1.3. This implies that an (English) ciphertext having an index of coincidence I x 0.065 is probably associated with a mono-alphabetic substitution cipher, since this statistic will not change if the letters are simply relabeled (which is the effect of encrypting with a simple substitution). The longer and more random a Vigenkre cipher keyword is, the more evenly the letters are distributed throughout the ciphertext. With a very long and very random keyword, we would expect to find

$$I \approx 26 \left(\frac{1}{26}\right)^2 = \frac{1}{26} \approx 0.03846.$$

Prof.Dr Hala Bahjat Abdul wahab

Therefore, a ciphertext having I **E** 0.03846 could be associated with a polyalphabetic cipher using a large keyword. Note that for any English ciphertext, the index of coincidence I must satisfy 0.03846 5 I 5 0.065. The question remains as to how to determine the length of the keyword of a Vigenkre cipher using the index of coincidence. The main weakness of the Vigenkre (or any similar periodic cipher) is that two identical characters occurring a distance apart that is a multiple of the key length will be encrypted identically. In such cryptosystems, the key length k can be approximated by a function involving the index of coincidence I and the length of the ciphertext R. The following example illustrates this technique. Suppose an English plaintext containing n letters is encrypted using a VigenBre cipher, with a keyword of length k , where, for simplicity, we assume R is a multiple of k . Now suppose that we arrange the ciphertext letters into a rectangular array of n / k rows and k columns, from left to right, top to bottom. If we select two letters from different columns in the array, this would be similar to choosing from a collection of letters that is uniformly distributed, since the keyword is more-or-less “random”.

Prof.Dr.Hala Bahjat Abdul wahab

## *Statistical cryptanalysis*

***Introduction:***

There are a number of aids to identification and solution available to help you as a cryptanalyst. By preparing character frequency counts, performing statistical

tests, and recording observed repetitions and patterns in messages, you can compare the data to established norms for various systems and languages.

***Language Characteristics***

Each language has characteristics that aid successful cryptanalysts.

***a.*** The individual letters of any language occur with greatly varying frequencies. Some letters are used a great deal. Others are used only a small percentage of the time. In English, the letter E is the most common letter used. It occurs about 13 percent of the time, or about once in every eight letters. In small samples, other letters may be more common, but in almost any sample of 1,000 letters of text or more, E will be the most frequent letter. In other languages, other letters sometimes dominate. In Russian, for example, O is the most common letter. The eight highest frequency letters in English, shown in descending order, are **E, T, N, R, O, A,I** and **S**. The eight highest frequency letters make up about 67 percent of our language. The remaining 18 letters only make up 33 percent of English text. The lowest frequency letters are **J, K, Q, X,** and **Z.** These five letters makeup only a little over 1 percent of English text. The vowels, **A, E, I, O, U** and **Y**, make up about 40 percent of English text. In many cryptographic systems, these frequency relationships show through despite the encryption. The analysis techniques explained in the following lecture make repeated use of these frequency relationships. In particular, you should remember the high frequency letters, **ETNROAIS**, and the low frequency letters, **JKQXZ**, for their repeated application. The word **SENORITA**, which includes the high frequency letters is one way to remember them. Some people prefer to remember the pronounceable ETNORIAS as a close approximation of the descending frequency order. Choose the method you prefer. The high frequency letters are referred to frequently.

Prof.Dr.Hala Bahjat Abdul wahab

**b**. Just as single letters have typical frequency expectations, multiple letter combinations occur with varying, but predictable frequencies, too. The most common pair of letters, or digraph, is EN. After EN, RE and ER are the most common digraphs. There are 676 different possible digraphs in English, but the most common 18 make up 25 percent of the language. Some cryptographic systems do not let individual letter frequencies show through the encryption, but let digraphic frequencies come through. The systems explained in Part Three of this manual show this characteristic.

**c**. frequency expectations for sets of three letters (trigraphs) and four letters (tetragraphs). Each of these can be useful when studying cryptograms in which three and four letter repeated segments of text occur.

**d.**Repeated segments of two to four letters will often occur because they are common letter combinations, whether or not they are complete words by themselves. Longer repeated segments readily occur when words and phrases are reused in plaintext. When words are reused in plaintext, they may or may not show up as repeated segments in ciphertext. For a word to show through as a repeat in ciphertext, the same keys must be applied to the same plaintext more than once. Even complex systems which keep changing keys will sometimes apply the same keys to the same plaintext and a repeated ciphertext segment will result. Finding such repeats gives many single messages to all messages that you believe may have been encrypted with the same set of keys. If computer support is available to search for repeats for you, a great deal of time can be saved. If not, time spent scanning text to search for repeats will reward you for your time when you find them.

We first consider the weakest type of attack, namely a ciphertext-only attack. We also assume that the plaintext string is ordinary English text, without punctuation

Prof.Dr.Hala Bahjat Abdul wahab

or “spaces.” (This makes cryptanalysis more difficult than if punctuation and spaces were encrypted.)

Many techniques of cryptanalysis use statistical properties of the English language. A among these is the letter frequency distribution, which gives the percentage frequency of the characters in the given text. Various people have estimated the relative frequencies of the 26 letters by compiling statistics from numerous novels, magazines, and newspapers. The estimates in Table bellow were obtained by Beker and Piper.

On the basis of the above probabilities, Beker and Piper partition the 26 letters into five groups as follows:

1. E, having probability about 0.120
2. T, A, O, I, N, S, H, R, each having probabilities between 0.06 and 0.09
3. D, L, each having probabilities around 0.04
4. C, U, M, W, F, G, Y, P, B, each having probabilities between 0.015 and 0.028
5. V, K, J, X, Q, Z, each having probabilities less than 0.01.

It may also be useful to consider sequences of two or three consecutive letters called digrams and trigrams, respectively. The 30 most common digrams are (in decreasing order) TH, HE, IN, ER, AN, RE, ED, ON, ES, ST, EN, AT, TO, NT, HA, ND, OU, EA, NG, AS, OR, TI, IS, ET, IT, AR, TE, SE, HI, and OF. The twelve most common trigrams are (in decreasing order) THE ING, AND, HER, ERE, ENT, THA, NTH, WAS, ETH, FOR, and DTH.

Prof.Dr.Hala Bahjat Abdul wahab

**Table** Probabilities of Occurrence of the 26 Letters

| letter | probability | letter | probability |
|---|---|---|---|
| *A* | .082 | *N* | .067 |
| *B* | .015 | *O* | .075 |
| *C* | .028 | *P* | .019 |
| *D* | .043 | *Q* | .001 |
| *E* | .127 | *R* | .060 |
| *F* | .022 | *S* | .063 |
| *G* | .020 | *T* | .091 |
| *H* | .061 | *U* | .028 |
| *I* | .070 | *V* | .010 |
| *J* | .002 | *W* | .023 |
| *K* | .008 | *X* | .001 |
| *L* | .040 | *Y* | .020 |
| *M* | .024 | *Z* | .001 |

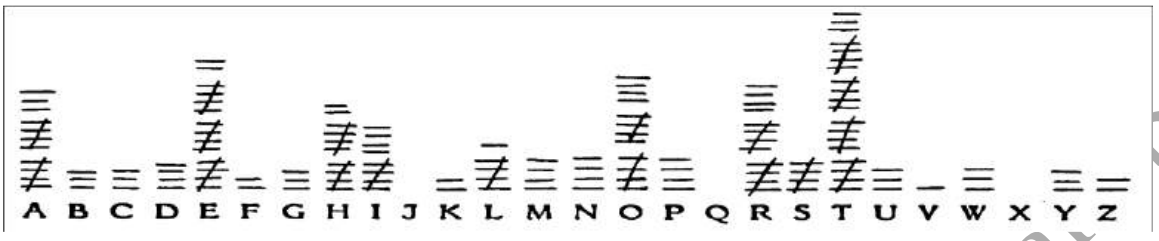
Prof.Dr.Hala Bahjat Abdul...ahab

## ***Unilateral Frequency Distribution***

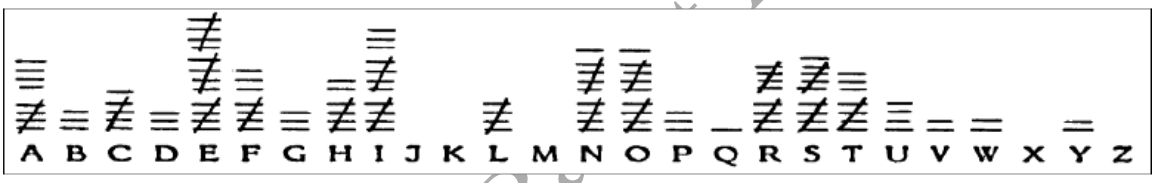
The most basic aid to identification and solution of cipher systems is the unilateral frequency distribution. The term unilateral means one letter at a time. A unilateral frequency distribution is a count of all the letters in selected text, taken one letter at a time.

**a.** The customary method of taking the distribution is to write the letters A through Z horizontally and mark each letter of the cryptogram with a dash above or below the appropriate letter. Proceed through the message from the first letter to the last, marking each letter in the distribution. Avoid the alternate method of counting all the As, Bs, Cs, and so forth, which is very subject to errors. For convenience, each group of five is crossed off by a diagonal slash. The unilateral frequency distribution for the first sentence in this paragraph is shown below.

Forth, which is very subject to errors. For convenience, each group of five is crossed off by a diagonal slash. The unilateral frequency distribution for the first sentence in this paragraph is shown below.

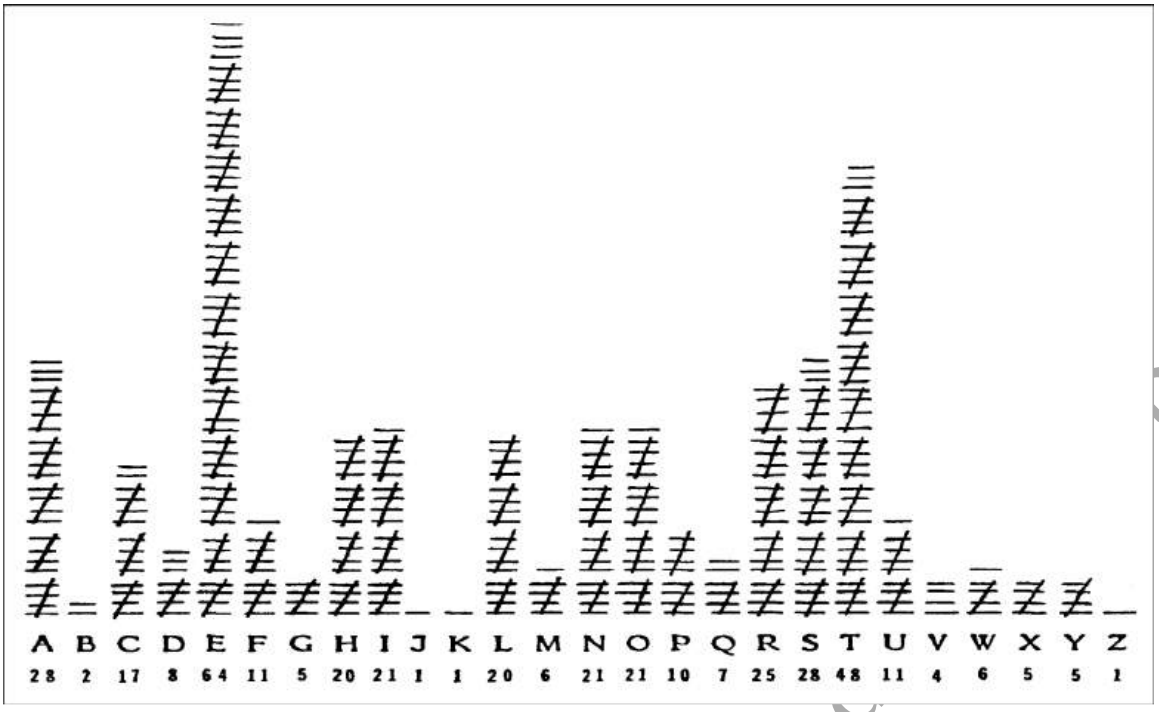



For comparison, the next example shows the frequency count for the fourth and fifth sentences in paragraph 2-9a.




**b**. Although individual letter frequencies differ, the pattern of high and low frequency letters is quite similar. The letters that stand above the others in each tally are, with few exceptions, the expected high frequency letters—ETNROAIS. The expected low frequency letters, JKQXZ, occur once or twice at most. Even in as small a sample as one or two sentences, expected patterns of usage start to establish themselves. Compare this to a frequency count of all letters in this paragraph.

Prof Dr.Hala Bakr Abdul wahab




- When a larger sample is taken, such as the above paragraph, the letters occur much closer to the expected frequency order of ETNROAIS. As expected, E and T are the two highest frequency letters. but the next series of high frequency letters in descending order of occurrence, ASRINO, differs slightly from the expected order of NROAIS. It would take a sample thousands of letters long to produce frequencies exactly in the expected order. Even then, differences in writing style between a field manual and military message texts could produce frequency differences. For example, the word the is often omitted from military message traffic for the sake of brevity. More frequent use of the raises the expected frequency of the letter H.

## Letter Frequencies in Cryptograms

As different cipher systems are explained in this manual, the ways in which letter frequencies can be used to aid identification and solution will be shown. Some basic considerations should be understood now.

**a**. In transposition systems, the letter frequencies of a cryptogram will be identical to that of the plaintext. A cryptogram in which the ciphertext letters

Prof.Dr.Hala Bahjat Abd

occur with the expected frequency of plaintext will usually be enciphered by a transposition system.

**b.** In the simplest substitution systems, each plaintext letter has one ciphertext equivalent. The ciphertext letter frequencies will not be identical to the plaintext frequencies, but the same numbers will be present in the frequency count as a whole. For example, if there are 33 Es in the plaintext of a message, and if E is enciphered by the letter K, then 33 Ks will appear in the ciphertext frequency count.

**c.** More complex substitution cipher systems, such as the polyalphabetic systems in ,will keep changing the equivalents. E might be enciphered by a K the first time it occurs and by different cipher letters each time it recurs. This will produce a very different looking frequency count.

**d**. To illustrate the differences in appearance of frequency counts for different types of systems, examine the four frequency counts in Figure 2-1. Each one is a frequency count of the message listed above it. The four messages are different, but each has the same plaintext. The first shows the plaintext and its frequency count. The second shows the frequencies of the same message enciphered by a transposition system. The third shows a simple substitution system encipherment. The fourth shows a polyalphabetic substitution encipherment.

**Roughness**

The four examples in Figure 2-1 show another characteristic of frequency counts which is useful in system identification. The first three distributions all contain the same letter frequencies. In the first two, the plaintext and the transposition examples, there are 16 Es. In the third, where E has been replaced by W, there are 16 Ws. Where there were 9 As, there are now 9 Ls. Where there was 1 K, there is now 1 C. The first three distributions show the same wide differences between the highest frequency letters and the lowest. The fourth

Prof.Dr.Hala Bahjat Abdul wahab

distribution is very different. The distribution lacks the wide differences between the highest and lowest frequency letters. Where the first three showed distinct highs and lows, or peaks and troughs, in the distributions, the fourth is relatively flat.

**a**. Frequency counts which show the same degree of difference between peaks and troughs as plaintext are considered to be rough distributions. Systems which suppress the peaks and troughs of plaintext letters by changing their equivalents produce flatter distributions. If letters were selected randomly from the 26 letters of the English alphabet, the resulting distribution would look very much like the fourth example. Random selection will not produce a perfectly level distribution, but it will appear quite flat in comparison to plaintext.

bdul wahab

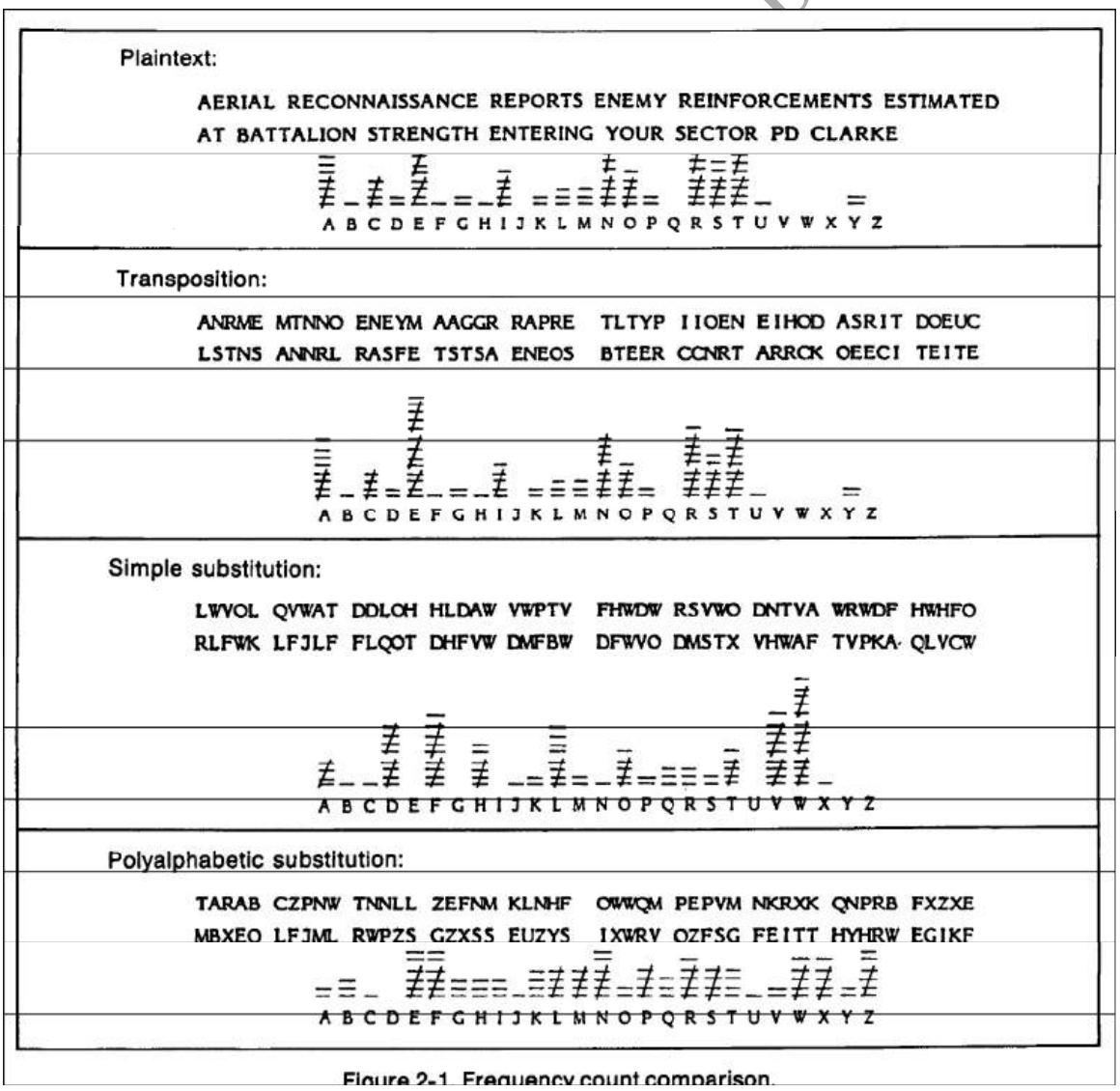



Figure 2-1. Frequency count comparison.

**b.** The simplest substitution systems tend to produce rough distributions. The most secure tend to produce flat distributions. Many other systems tend to fall in between. You can use the degree of roughness as one of the aids to system identification.

Prof.Dr.Hala Bahjat Abdul wahab

***Cryptanalysis of the Affine Cipher***

As a simple illustration of how cryptanalysis can be performed using statistical data, let's look first at the Affine Cipher. Suppose Oscar has intercepted the following ciphertext:

**Example:**

Ciphertext obtained from an Affine Cipher

**FMXVEDKAPHFERBNDKRXRSREFMORUDSDKDVSHVUFEDKAPR KDLYEVLRHHRH**

The frequency analysis of this ciphertext is given in Table below.

There are only 57 characters of ciphertext, but this is sufficient to cryptanalyze an Affine Cipher. The most frequent ciphertext characters are:

| Character | No. of occurrences |
|---|---|
| R | (8occurrences), |
| D | (7occurrences), |
| E,H,K | (5occurrenceseach), |
| F,S,V | (4occurrenceseach). |

As a first guess, we might hypothesize that R is the encryption of e and D is the encryption of t, since e and t are (respectively) the two most common letters.

Expressed numerically, we have $e_K(4) = 17$ and $e_K(19) = 3$. Recall that $e_K(x) = ax + b$, where a and b are unknowns. So we get two linear equations in two unknowns:

$$4a + b = 17$$
$$19a + b = 3.$$

**Table** Frequency of Occurrence of the 26 Ciphertext Letters

| letter | frequency | letter | frequency |
|---|---|---|---|
| A | 2 | N | 1 |
| B | 1 | O | 1 |
| C | 0 | P | 2 |
| D | 7 | Q | 0 |
| E | 5 | R | 8 |
| F | 4 | S | 3 |
| G | 0 | T | 0 |
| H | 5 | U | 2 |
| I | 0 | V | 4 |
| J | 0 | W | 0 |
| K | 5 | X | 2 |
| L | 2 | Y | 1 |
| M | 2 | Z | 0 |

This system has the unique solution a = 6, b = 19 (in $\mathbb{Z}_{26}$). But this is an illegal key, since gcd(a, 26) = 2 > 1. So our hypothesis must be incorrect.

Our next guess might be that R is the encryption of e and E is the encryption of t. Proceeding as above, we obtain a = 13, which is again illegal.

So we try the next possibility, that R is the encryption of e and H is the encryption of t. This yields a = 8, again impossible. Continuing, we suppose that R is the encryption of e and K is the encryption of t. This produces a = 3, b = 5, which is at least a legal key. It remains to compute the decryption function corresponding to K = (3, 5), and then to decrypt the ciphertext to see if we get a meaningful string of English, or nonsense. This will confirm the validity of (3, 5).

If we perform these operations, we have $d_K(y) = 9y - 19$ and the given ciphertext decrypts to yield:

**Algorithms are quite general definition so far it**metic processes

We conclude that we have determined the correct key.

Prof Dr. Hala Bahjat Abdul Wahab

## ***Coincidence Tests***

Judging whether a given frequency distribution has the same degree of roughness as plaintext or random text is not easy to do by eye alone. To help you make this determination, a number of statistical tests have been developed for your use. The tests are based in probability theory, but you can use the tests whether or not you understand the underlying theories. The most common tests are called coincidence tests.

**c.** If you pick any two letters from a message, compare them together, and they happen to be the same letter, they are said to coincide. A comparison of the same letters, for example, two As is a coincidence. This comparison can be made of single letters or pairs of letters or longer strings of letters.

**d.** If you compare two single letters selected at random from the English alphabet, the probability of their being the same is 1 in 26. One divided by 26 is .0385. Expressed as a percentage, 1/26 is slightly less than 4 percent. You would expect to find a coincidence 3.85 times on the average in every 100 comparisons.

**e.** If you select two letters from English plaintext, however, the probability of their being the same is higher than 1 in 26. Frequency studies have shown that the probability of a coincidence in English plaintext is .0667. In other words, in every 100 comparisons, you would expect to find 6.67 coincidences in plaintext. Each language has its own probabilities, but similar traits occur in each alphabetic language.

**f.** Different coincidence tests use different methods of comparing letters with each other, but each rests on the probabilities of random and plaintext comparisons. The actual number of coincidences in a cryptogram can be compared with the random and plaintext probabilities to help make judgments about the cryptogram.

Prof.Dr.Hala Bahjat Abdul wahab

### Index of Coincidence

A common way of expressing the results of a coincidence test is the index of coincidence (XC). The index of coincidence is the ratio of observed coincidences to the number expected in a random distribution. For plaintext, the expected index of coincidence for single letters in English is the ratio of .0667 to .0385, which is 1.73.

# *Index of Coincidence Example*

While working at the Riverbank Laboratory, William F. Friedman (1891L 1969) developed the index of coincidence. For a given ciphertext, the index of coincidence I is defined to be the probability that two randomly selected letters in the ciphertext represent, t.he same plaintext symbol. For a given ciphertext, let no, 121,. . . ,*1225* be the respective letter counts of **A,** B, C, . . . , Z in the ciphertext, and set ***71*** = n o + **111** + . . . + r125. Then, the index of coincidence can be computed as

$$I = \frac{\binom{n_0}{2} + \binom{n_1}{2} + \cdots + \binom{n_{25}}{2}}{\binom{n}{2}} = \frac{1}{n(n-1)} \sum_{i=0}^{25} n_i(n_i - 1).$$

To see why the index of coincidence gives us useful information, first note that the empirical probability of randomly selecting two identical letters from a large English plaintext is

$$\sum_{i=0}^{25} p_i^2 \approx 0.065,$$

Prof. Hala Baliat Abdulwahab

where po is the probability of selecting an **A,** p l is the probability of selecting a B, and so on, and the values of p , are given in Table 1.3. This implies that an (English) ciphertext having an index of coincidence I x 0.065 is probably associated with a mono-alphabetic substitution cipher, since this statistic will not change if the letters are simply relabeled (which is the effect of encrypting with a simple substitution).

The longer and more random a Vigenkre cipher keyword is, the more evenly the letters are distributed throughout the ciphertext. With a very long and very random keyword, we would expect to find

$$I \approx 26\left(\frac{1}{26}\right)^2 = \frac{1}{26} \approx 0.03846.$$

Therefore, a ciphertext having I **E** 0.03846 could be associated with a polyalphabetic cipher using a large keyword. Note that for any English ciphertext, the index of coincidence I must satisfy 0.03846 5 I 5 0.065. The question remains as to how to determine the length of the keyword of a Vigenkre cipher using the index of coincidence. The main weakness of the Vigenkre (or any similar periodic cipher) is that two identical characters occurring a distance apart that is a multiple of the key length will be encrypted identically. In such cryptosystems, the key length k can be approximated by a function involving the index of coincidence I and the length of the ciphertext N

**Example:- calculate Ic for this cipher text, and find the algorithm that is used**

**EEAHR RFOWW TGDTE SCHES ROEST EMCNEAOOTL AKNEE TSSEO AVXNC STPOO OEOEATASBI OAEER AXHEE RADNF PSINO ISEAURPNED XEPSE PFCDL LZTER JAETY RETHE**

Start by guessing the letter by the most frequent & compensated by the priority mentioned in the table.

1. When guessing 2 letters & the key , decode the cipher by additive algorithm.
2. We can guess which algorithm is used, by using this equation

Prof.Dr.Hala Bahjat Abdul wahab

Ic = Σ Fi ( Fi – 1 ) /n(n-1)

Ic > 0.065 => mono algorithm
Ic > 0.065 => poly algorithm
Ic = 0.065 => transposition algorithm

SOL/

| Letter | Frequency | Letter | Frequency | Letter | Frequency |
|---|---|---|---|---|---|
| A | 11 | J | 1 | S | 10 |
| B | 1 | K | 1 | T | 10 |
| C | 4 | L | 3 | U | 1 |
| D | 4 | M | 1 | V | 1 |
| E | 24 | N | 6 | W | 2 |
| F | 3 | O | 11 | X | 3 |
| G | 1 | P | 5 | Y | 1 |
| H | 4 | Q | 0 | Z | 1 |
| I | 3 | R | 8 | | |

Ic = Σ Fi ( Fi – 1 ) /
n(n-1)
Ic = 11(11-1) + 1(1-0) + 4(4-1) + 4(4-1) + 24(24-1) + 3(3-1) + 1(1-0) + 4(4-1) + 3(3-1) + 1(1 – 0 ) + 1(1-0) + 3(3-1) + 1(1-0) + 6(6-1) + 11(11-1) + 5(5-1) + 8(8-1) + 10(10-1) + 10(10-1) + 1(1-0) + 1(1-0) + 2(2-1) + 3(3-1) + 1(1-0) + 1(1-0) / 120 ( 120 – 1 )
Ic = 1072/14280 = 0.071 => mono algorithm

# LFSR Cryptanalysis

# Berlekamp-Massey Algorithm Attack

Given a binary sequence, the Berlekamp-Massey Algorithm provides an efficient method to determine the smallest LFSR that can generate the sequence. Here, “size” refers to the number of stages in the LFSR. The size of the minimal LFSR is known as the linear complexity (or linear span) of the sequence. Due to the threat of known plaintext attacks, a keystream must have a large period. Furthermore, due to the Berlekamp-Massey Algorithm, there must not exist any small LFSR that can generate a given keystream sequence. We expand on this

Prof.Dr.Hala Bahjat Abdul-Wahab

point below, after we have discussed the Berlekamp- Massey Algorithm and some of its implications. The Berlekamp- Massey Algorithm appears in Table1**,**

$$s = (s_0, s_1, \ldots, s_{n-1})$$

where denotes the binary sequence under consideration, L is the linear complexity and *C(x)* is the connection polynomial of the minimal LFSR. Note that the coefficients of all polynomials are to be taken modulo ***2.*** Also, *d* is known as the discrepancy, and the connection polynomial is of the form The

$$C(x) = c_0 + c_1 x + c_2 x^2 + \cdots + c_L x^L.$$

Berlekamp-Massey Algorithm processes the sequence s sequentially and at step k, the polynomial *C(x)* is the connection polynomial for the first ***k*** + 1 bits of s and L is the corresponding linear complexity. At step k, if the discrepancy is d = 0, then the connection polynomial *C(x)* computed at step k - **1** is also the connection polynomial for so, s1,. . . , sk and no change to *C(x)* or L is required. If, on the other hand, the discrepancy is d = 1, then C(x) must be modified, and the linear complexity L increases if the current value of L lies below the n/2 line. Massey Algorithm, the minimal LFSR will have been obtained. Below, we see that this property has implications for stream cipher design. It is not too difficult to show that the Berlekamp- Massey Algorithm requires on the order of n2 operations [62], where n is the number of bits processed and the operations are XOR. This is the most efficient known general algorithm for solving the shift register synthesis problem. However, there are more efficient algorithms for certain special cases;

Prof.Dr.Hala Bahjat Al wahab

Table1: Berlekamp~-Massey Algorithm

```
// Given binary sequence s = (s_0, s_1, s_2, ..., s_{n-1},
// Find linear complexity L and connection polyı
BM(s)
    C(x) = B(x) = 1
    L = N = 0
    m = -1
    while N < n // n is length of input sequence
        d = s_N ⊕ c_1 s_{N-1} ⊕ c_2 s_{N-2} ⊕ ··· ⊕ c_L s_{N-L}
        if d == 1 then
            T(x) = C(x)
            C(x) = C(x) + B(x) x^{N-m}
            if L ≤ N/2 then
                L = N + 1 - L
                m = N
                B(x) = T(x)
            end if
        end if
        N = N + 1
    end while
    return(L)
end BM
```

wahab

Prof.Dr.

Next, we illustrate the Berlekamp-Massey Algorithm. Consider the periodic sequcnce s, with one period given by …….1

For this sequence, the first few steps of the Berlekamp-Massey Algorithm are illustrated in Table2**.** For the periodic sequence ***(1)*** **,** the linear complexity is L = 6 (Problem 1 asks for the connection polynomial). Therefore, if we let 10011 be the initial fill of the LFSR corresponding to the connection polynomial determined by the Berlekamp-Massey Algorithm, the LFSR generates the sequencc s in (1). Here, we do not attempt to prove the validity of the Berlekarnp-Massey Algorithm. but we note in passing that the algorithm is closely related to the extended Euclidean Algorithm and continued fraction

algorithms. We also note one important- but non-obvious--fact, namely, that any 2L bits through the Berlekamp.

# DES Cryptanalysis

# An Attack on a 3-round DES

## Differential Cryptanalysis

One very well-known attack on **DES** is the method of "differential cryptanalysis" introduced by Biham and Shamir. This is a chosen-plaintext attack. Although it does not provide a practical method of breaking the usual 16-round **DES**, it does succeed in breaking **DES** if the number of rounds of encryption is reduced. For instance, 8-round **DES** can be broken in only a couple of minutes on a small personal computer.

We will now describe the basic ideas used in this technique. For the purposes of this attack, we can ignore the initial permutation IP and its inverse (it has no effect on cryptanalysis). As mentioned above, we consider **DES** restricted to $n$ rounds, for various values of $n \leq 16$. So, in this setting, we will regard $L_0R_0$ as the plaintext, and $L_nR_n$ as the ciphertext, in an $n$-round **DES**. (Note also that we are not inverting $L_nR_n$.)

Differential cryptanalysis involves comparing the x-or (exclusive-or) of two plaintexts to the x-or of the corresponding two ciphertexts. In general, we will be looking at two plaintexts $L_0R_0$ and $L_n^*R_n^*$ with a specified x-or value $L_0'R_0' = L_0R_0 \oplus L_0^*R_0^*$. Throughout this discussion, we will use prime markings (′) to indicate the x-or of two bitstrings.

Prof.Dr. Hala Bahjat Abdul wahab

***DEFINITION :1*** *Let $S_j$ be a particular S-box* $(1 \le j \le 8)$*. Consider an (ordered) pair of bitstrings of length six, say* $(B_j, B_j^*)$*. We say that the input* x-or *(of $S_j$) is* $B_j \oplus B_j^*$ *and the output* x-or *(of $S_j$) is* $S_j(B_j^*) \oplus S_j(B_j^*)$.

Note that an input x-or is a bitstring of length six and an output x-or is a bitstring of length four.

***DEFINITION :2*** *For any* $B_j' \in (\mathbb{Z}_2)^6$*, define the set* $\Delta(B_j')$ *to consist of the ordered pairs* $(B_j, B_j^*)$ *having input x-or* $B_j'$.

It is easy to see that any set $\Delta(B_j')$ contains $2^6 = 64$ pairs, and that

$$\Delta(B_j') = \{(B_j, B_j \oplus B_j') : B_j \in (\mathbb{Z}_2)^6\}.$$

For each pair in $\Delta(B_j')$, we can compute the output x-or of $S_j$ and tabulate the resulting distribution. There are 64 output x-ors, which are distributed among $2^4 = 16$ possible values. The non-uniformity of these distributions will be the basis for the attack.

***Example 1***

Suppose we consider the first S-box, $S_1$, and the input x-or 110100. Then

$$\Delta(110100) = \{(000000, 110100), (000001, 110101), \ldots, (111111, 001011)\}.$$

For each ordered pair in the set $\Delta(110100)$, we compute output x-or of $S_1$. For example, $S_1(000000) = E_{16} = 1110$ and $S_1(110100) = 9_{16} = 1001$, so the output x-or for the pair (000000, 110100) is 0111.

If this is done for all 64 pairs in $\Delta(110100)$, then the following distribution of output x-ors is obtained:

Prof Dr.Hala Bahjat Abdul wahab

| 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 |
|---|---|---|---|---|---|---|---|
| 0 | 8 | 16 | 6 | 2 | 0 | 0 | 12 |

| 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |
|---|---|---|---|---|---|---|---|
| 6 | 0 | 0 | 0 | 0 | 8 | 0 | 6 |

In Example 3.1, only eight of the 16 possible output x-ors actually occur. This particular example has a very non-uniform distribution. In general, if we fix an S-box $S_j$ and an input x-or $B'_j$, then on average, it turns out that about 75 - 80% of the possible output x-ors actually occur.

It will be convenient to have some notation to describe these distributions and how they arise, so we make the following definitions.

***DEFINITION 3.3*** *For* $1 \leq j \leq 8$, *and for bitstrings* $B'_j$ *of length six and* $C'_j$ *of length four, define*

$$IN_j(B'_j, C'_j) = \{B_j \in (\mathbb{Z}_2)^6 : S_j(B_j) \oplus S_j(B_j \oplus B'_j) = C'_j\}$$

*and*

$$N_j(B'_j, C'_j) = |IN_j(B'_j, C'_j)|.$$

Prof.Dr.Nala Bahjat Abdul wahab

| output x-or | possible inputs |
|---|---|
| 0000 | |
| 0001 | 000011, 001111, 011110, 011111<br>101010, 101011, 110111, 111011 |
| 0010 | 000100, 000101, 001110, 010001<br>010010, 010100, 011010, 011011<br>100000, 100101, 010110, 101110<br>101111, 110000, 110001, 111010 |
| 0011 | 000001, 000010, 010101, 100001<br>110101, 110110 |
| 0100 | 010011, 100111 |
| 0101 | |
| 0110 | |
| 0111 | 000000, 001000, 001101, 010111<br>011000, 011101, 100011, 101001<br>101100, 110100, 111001, 111100 |
| 1000 | 001001, 001100, 011001, 101101<br>111000, 111101 |
| 1001 | |
| 1010 | |
| 1011 | |
| 1100 | |
| 1101 | 000110, 010000, 010110, 011100<br>100010, 100100, 101000, 110010 |
| 1110 | |
| 1111 | 000111, 001010, 001011, 110011<br>111110, 111111 |

figure(1):Possible inputs with input x-or 110100

$N_j(B'_j, C'_j)$ counts the number of pairs with input x-or equal to $B'_j$ which have output x-or equal to $C'_j$ for the S-box $S_j$. The actual pairs having the specified input x-ors and giving rise to the specified output x-ors can be obtained from the set $IN_j(B'_j, C'_j)$. Observe that this set can be partitioned into $N_j(B'_j, C'_j)/2$ pairs, each of which has (input) x-or equal to $B'_j$.

Observe that the distribution tabulated in Example 1 consists of the values $N_1(110100, C'_1)$, $C'_1 \in (\mathbb{Z}_2)^4$. The sets $IN_1(110100, C'_1)$ are listed in Figure 3.8.

For each of the eight S-boxes, there are 64 possible input x-ors. Thus, there are 512 distributions which can be computed. These could easily be tabulated by computer.

Prof.Dr.Hala Bahjat Abdul wahab

Recall that the input to the S-boxes in round $i$ is formed as $B = E \oplus J$, where $E = E(R_{i-1})$ is the expansion of $R_{i-1}$ and $J = K_i$ consists of the key bits for round $i$. Now, the input x-or (for all eight S-boxes) can be computed as follows:

$$\begin{aligned} B \oplus B^* &= (E \oplus J) \oplus (E^* \oplus J) \\ &= E \oplus E^*. \end{aligned}$$

It is very important to observe that the input x-or does not depend on the key bits $J$. (However, the output x-or certainly does depend on these key bits.)

We will write each of $B$, $E$ and $J$ as the concatenation of eight 6-bit strings:

$$\begin{aligned} B &= B_1 B_2 B_3 B_4 B_5 B_6 B_7 B_8 \\ E &= E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 \\ J &= J_1 J_2 J_3 J_4 J_5 J_6 J_7 J_8, \end{aligned}$$

and we write $B^*$ and $E^*$ in a similar way. Let us suppose for the moment that we know the values $E_j$ and $E_j^*$ for some $j$, $1 \le j \le 8$, and the value of the output x-or for $S_j, C_j' = S_j(B_j) \oplus S_j(B_j^*)$. Then it must be the case that

$$E_j \oplus J_j \in IN_j(E_j', C_j'),$$

where $E_j' = E_j \oplus E_j^*$.

Suppose we define a set $test_j$ as follows:

***DEFINITION :4*** *Suppose $E_j$ and $E_j^*$ are bitstrings of length six, and $C_j'$ is a bitstring of length four. Define*

$$test_j(E_j, E_j^*, C_j') = \{B_j \oplus E_j : B_j \in IN_j(E_j', C_j')\},$$

Prof. Dr. Hala Bahjat Abdul wahab

*where* $E'_j = E_j \oplus E^*_j$.

That is, we take the x-or of $E_j$ with every element of the set $IN_j(E'_j, C'_j)$.

The following result is an immediate consequence of the discussion above.

***THEOREM:1***

*Suppose* $E_j$ *and* $E^*_j$ *are two inputs to the S-box* $S_j$*, and the output x-or for* $S_j$ *is* $C'_j$. *Denote* $E'_j = E_j \oplus E^*_j$. *Then the key bits* $J_j$ *occur in the set* $test_j(E_j, E^*_j, C'_j)$.

Observe that there will be exactly $N_j(E'_j, C'_j)$ bitstrings of length six in the set $test_j(E_j, E^*_j, C'_j)$; the correct value of $J_j$ must be one of these possibilities.

*Example 3.2*

Suppose $E_1 = 000001, E^*_1 = 110101$ and $C'_1 = 1101$. Since $N_1$ (110100, 1101) = 8, there will be exactly eight bitstrings in the set $test_1$ (000001, 110101, 1101). From Figure 3.8, we see that

$$IN_1(110100, 1101) = \{000110, 010000, 010110, 011100, 100010, 100100, 101000, 110010\}.$$

Hence,

$$test_1(000001, 110101, 1101) = \{000111, 010001, 010111, 011101, 100011, 100101, 101001, 110011\}.$$

If we have a second such triple $E_1, E^*_1, C'_1$, then we can obtain a second set $test_1$ of possible values for the keybits in $J_1$. The true value of $J_1$ must be in the intersection of both sets. If we have several such triples, then we can quickly

Prof. Hala Bahjat Abdul-Wahab

determine the key bits in $J_1$. One straightforward way to do this is to maintain an array of 64 counters, representing the 64 possibilities for the six key bits in $J_1$. A counter is incremented every time the corresponding key bits occur in a set $test_1$ for a particular triple. Given $t$ triples, we hope to find a unique counter which has the value $t$; this will correspond to the true value of the keybits in $J_1$.

Let's now see how the ideas of the previous section can be applied in a chosen plaintext attack of a 3-round **DES**. We will begin with a pair of plaintexts and corresponding ciphertexts: $L_0R_0, L_0^*R_0^*, L_3R_3$ and $L_3^*R_3^*$. We can express $R_3$ as follows:

$$\begin{aligned} R_3 &= L_2 \oplus f(R_2, K_3) \\ &= R_1 \oplus f(R_2, K_3) \\ &= L_0 \oplus f(R_0, K_1) \oplus f(R_2, K_3). \end{aligned}$$

$R_3^*$ can be expressed in a similar way, and hence

$$R_3' = L_0' \oplus f(R_0, K_1) \oplus f(R_0^*, K_1) \oplus f(R_2, K_3) \oplus f(R_2^*, K_3).$$

Now, suppose we have chosen the plaintexts so that $R_0 = R_0^*$, i.e., so that

$$R_3' = L_0' \oplus f(R_0, K_1) \oplus f(R_0^*, K_1) \oplus f(R_2, K_3) \oplus f(R_2^*, K_3).$$

Now, suppose we have chosen the plaintexts so that $R_0 = R_0^*$, i.e., so that

$$R_0' = 00\ldots0.$$

Prof.Dr.Rola Ba... Abdul wahab

Input: $L_0R_0, L_0^*R_0^*, L_3R_3$ and $L_3^*R_3^*$, where $R_0 = R_0^*$
1. compute $C' = \mathrm{P}^{-1}(R_3' \oplus L_0')$
2. compute $E = \mathrm{E}(L_3)$ and $E^* = \mathrm{E}(L_3^*)$
3. **for** $j = 1$ **to** 8 **do**
   compute $test_j(E_j, E_j^*, C_j')$

**Figure** 2 Differential attack on 3-round DES

Then $f(R_0, K_1) = f(R_0^*, K_1)$ and so

$$R_3' = L_0' \oplus f(R_2, K_3) \oplus f(R_2^*, K_3).$$

At this point, $R_3'$ is known since it can be computed from the two ciphertexts, and $L_0'$ is known since it can be computed from the two plaintexts. This means that we can compute $f(R_2, K_3) \oplus f(R_2^*, K_3)$ from the equation

$$f(R_2, K_3) \oplus f(R_2^*, K_3) = R_3' \oplus L_0'.$$

Now, $f(R_2, K_3) = \mathrm{P}(C)$ and $f(R_2^*, K_3) = \mathrm{P}(C^*)$, where $C$ and $C^*$, respectively, denote the two outputs of the eight S-boxes (recall that P is a fixed, publicly known permutation). Hence,

$$\mathrm{P}(C) \oplus \mathrm{P}(C^*) = R_3' \oplus L_0',$$

and consequently

$$C' = C \oplus C^* = \mathrm{P}^{-1}(R_3' \oplus L_0').$$

This is the output x-or for the eight S-boxes in round three.

Now, $R_2 = L_3$ and $R_2^* = L_3^*$ are also known (they are part of the ciphertexts). Hence, we can compute

Prof.Dr.Hala Bahjat Abdul wahab

$$E = \mathrm{E}(L_3)$$

and

$$E^* = \mathrm{E}(L_3^*)$$

using the publicly known expansion function E. These are the inputs to the S-boxes for round three. So, we now know $E$, $E^*$, and $C'$ for the third round, and we can proceed, as in the previous section, to construct the sets $test_1, \ldots, test_8$ of possible values for the key bits in $J_1, \ldots, J_8$.

A pseudo-code description of this algorithm is given in Figure 3.9. The attack will use several such triples $E$, $E^*$, $C'$. We set up eight arrays of counters, and thereby determine the 48 bits in $K_3$, the key for the third round. The 56 bits in the key can then be computed by an exhaustive search of the $2^8 = 256$ possibilities for the remaining eight key bits.

Let's look at an example to illustrate.

***Example 2:***

Suppose we have the following three pairs of plaintexts and ciphertexts, where the plaintexts have the specified x-ors, that are encrypted using the same key. We use a hexadecimal representation, for brevity:

| **plaintext** | **ciphertxt** |
|---|---|
| 748502CD38451097 | 03C70306D8A09F10 |
| 3874756438451097 | 78560A0960E6D4CB |

Prof.Dr.Hala Bahjat Abdul wahab

| | |
|---|---|
| 486911026ACDFF31 | 45FA285BE5ADC730 |
| 375BD31F6ACDFF31 | 134F7915AC253457 |

| | |
|---|---|
| 357418DA013FEC86 | D8A31B2F28BBC5CF |
| 12549847013FEC86 | 0F317AC2B23CB944 |

From the first pair, we compute the S-box inputs (for round 3) from Equations (3.2) and (3.3). They are:

$$E = 000000000111111000001110100000000110100000001100$$
$$E^* = 101111110000001010101100000001010100000001010010.$$

The S-box output x-or is calculated using Equation (3.1) to be:

$$C' = 100101100101110101011011011001 11.$$

From the second pair, we compute the S-box inputs to be

$$E = 101000001011111111111010000010101000000101111 0110$$
$$E^* = 100010100110101001011110101111110010100010101010$$

and the S-box output x-or is

$$C' = 100111001001110000011111010101 10.$$

From the third pair, the S-box inputs are

$$E = 111011110001010100000110100011111011010010101111 11$$
$$E^* = 000001011110100110100010101111110101011000000100$$

and the S-box output x-or is

Prof.Dr. Ala Bedaiat Abdul wahab

$$C' = 110101010111010111011011100101011.$$

Next, we tabulate the values in the eight counter arrays for each of the three pairs. We illustrate the procedure with the counter array for $J_1$ from the first pair. In this pair, we have $E'_1 = 101111$ and $C'_1 = 1001$. The set

$$IN_1(101111, 1001) = \{000000, 000111, 101000, 101111\}.$$

Since $E_1 = 000000$, we have that

$$J_1 \in test_1(000000, 101111, 1001) = \{000000, 000111, 101000, 101111\}.$$

Hence, we increment the values 0, 7, 40, and 47 in the counter array for $J_1$.The final tabulations are now presented. If we think of a bit-string of length six as being the binary representation of an integer between 0 and 63, then the 64 values correspond to the counts of 0, 1, . . . , 63. The counter arrays are as

| $J_1$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 3 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

| $J_2$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 3 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 2 | 0 | 0 | 0 |

| $J_3$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

Prof. Bahjat Abdul wahab

| $J_4$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 3 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 2 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 1 |

| $J_5$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 2 | 0 |

| $J_6$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 1 | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| $J_7$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 2 | 1 | 0 | 1 | 0 | 3 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 2 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 1 | 2 | 1 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |

| $J_8$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| 0 | 3 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Prof.Dr.Hala B l wahab

In each of the eight counter arrays, there is a unique counter having the value 3. The positions of these counters determine the key bits in $J_1, \ldots, J_8$. These positions are (respectively): 47, 5, 19, 0, 24, 7, 7, 49. Converting these integers to binary, we obtain $J_1, \ldots, J_8$:

$J_1 = 101111$

$J_2 = 000101$

$J_3 = 010011$

$J_4 = 000000$

$J_5 = 011000$

$J_6 = 000111$

$$J_7 = 000111$$

$$J_8 = 110001.$$

We can now construct 48 bits of the key, by looking at the key schedule for round 3. It follows that $K$ has the form

| | | | |
|---|---|---|---|
| 0001101 | 0110001 | 01?01?0 | 1?00100 |
| 0101001 | 0000??0 | 111?11? | ?100011 |

where parity bits are omitted and "?" denotes an unknown key bit. The complete key (in hexadecimal, including parity bits), is:

1A624C89520DEC46.

Prof.Dr.Hala Bahjat Abdul Wahab

## ***Example***

Differential cryptanalysis it can be used in an attempt to crypt analyze 3-round DES. Using the following pairs of PT/CT to find the equations that help you to deduce the information about the key:

PT1=0001 0100 , PT2=0011 0100, CT1=0111 1111 ,CT2=0001 1100 **.**

**Sol:** let PTR be the right half of PT, etc. Note PTR1=PTR2 but

PTL2≠ PTL2

CTL1=0111= L2+F(R2,KEY3)=R1+F(R2,KEY3)=L0+F(R0,KEY1)+F(R2,KEY3)

= PTL1+F(PTR1,KEY1)+F(CTR1,KEY3) =

0001+F(PTR1,KEY1)+F(1111,KEY3).

Similarly CTL2=0001=0011+f(PTR2,KEY1)+F(1100,KEY3)

So we have CTL1+CTL2=0111+0001=0110 but also

CTL1+CTL2=0001+0011+F(1111,KEY3)+F(1100,KEY3)=0100

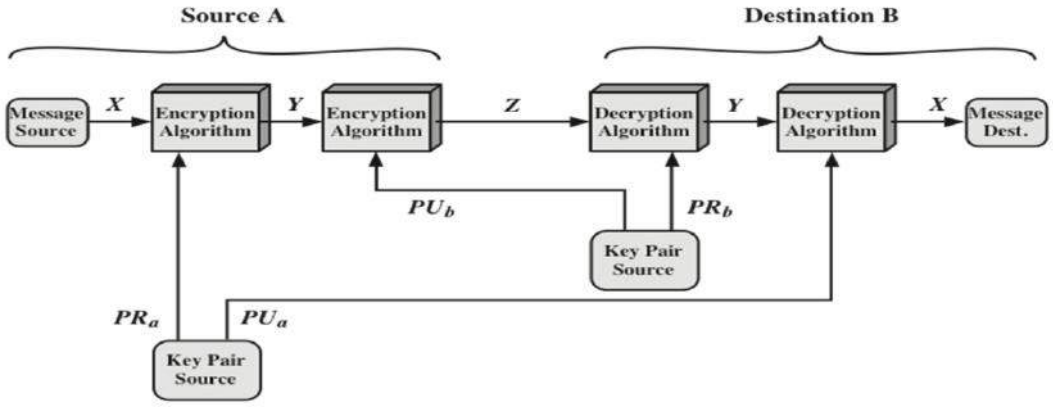
In general **F(CTR1,KEY3)+F(CTR2,KEY3)=PTL1+PTL2+CTL1+CTL2** and the right half of that equation is know.

Prof Dr.Hala Bahjat Abdul wahab

# Public key cryptanalysis

***PUBLIC KEY CRYPTOGRAPHY*** A form of cryptography which the key used to encrypt a message differs from the key used to decrypt it. In public key cryptography a, user has a pair of cryptographic keys a public key and a private key .The private key is kept secret, while the public key may be widely distributed. The two main branches public key cryptography are: 1. Public key encryption 2. Digital signatures **Why Public-Key Cryptography?** . public-key/two-key/ asymmetric cryptography involves the use of two keys: **a public-key**, which may be known by anybody, and can be used to encrypt messages, and verify signatures **a private-key**, known only to the recipient, used to decrypt messages, and sign (create) signatures.

***Applications for Public-Key Cryptosystems*** * encryption/ decryption (provide security) *a digital signatures (provide authentication) *key exchange (of session keys).




***Cryptanalysis of Asymmetric Cryptography*** Asymmetric cryptography (or public key cryptography) is cryptography that relies on using two keys; one private, and one public. Such ciphers invariably rely on "hard" mathematical problems as the basis of their security, so **an obvious point of attack is to develop methods for solving the problem**. Asymmetric schemes are designed around the (conjectured) difficulty of solving various mathematical problems. If an improved algorithm can be found to solve the problem, then the system is weakened. For example, the security of the Diffie-Hellman key exchange scheme depends on the difficulty of calculating the **discrete logarithm**. In 1983, Don Coppersmith found a faster way to find discrete logarithms (in certain groups), and thereby requiring cryptographers to use larger groups (or different types of groups). RSA's security depends (in part) upon the difficulty of integer factorization — a breakthrough in factoring would impact the security of RSA.

Prof.Dr. Hala Bahjat Abd

## *ADVANTAGES AND DISADVANTAGES OF ASYMMETRIC CRYPTOSYSTEM*

**ADVANTAGES** • In asymmetric or public key, cryptography there is no need for exchanging keys, thus eliminating the key distribution problem. • The primary advantage of public-key cryptography is increased security: the private keys do not ever need to be transmitted or revealed to anyone. • Can provide digital signatures that can be repudiated

**DISADVANTAGES** • A disadvantage of using public-key cryptography for encryption is speed: there are popular secret-key encryption methods which are significantly faster than any currently available public-key encryption method. There are many secret-key encryption methods that are significantly faster than any currently available public-key encryption method. Nevertheless, public-key cryptography can be used with secret-key cryptography to get the best of both worlds. For encryption, the best solution is to combine public- and secret-key systems in order to get both the security advantages of public-key systems and the speed advantages of secret-key systems. Such a protocol is called a digital envelope.

***Weaknesses*** Keys in public-key cryptography, due to their unique nature, are more computationally costly than their counterparts in secret-key cryptography. Asymmetric keys must be many times longer than keys in secret-cryptography in order to boast equivalent security .Keys in asymmetric cryptography are also more vulnerable to brute force attacks than in secret-key cryptography. There exist algorithms for public-key cryptography that allow attackers to crack private keys faster than a brute force method would require. The widely used and pioneering RSA algorithm has such an algorithm that leaves it susceptible to attacks in less than brute force time . While generating longer keys in other algorithms will usually prevent a brute force attack from succeeding in any meaningful length of time, these computations become more computationally intensive. These longer keys can still vary in effectiveness depending on the computing power available to an attacker. - Public-key cryptography also has vulnerabilities to attacks such as the man in the middle attack .In this situation, a

Prof.Dr.Hala Bahjat Abdul wahab

malicious third party intercepts a public key on its way to one of the parties involved. The third party can then instead pass along his or her own public key with a message claiming to be from the original sender. An attacker can use this process at every step of an exchange in order to successfully impersonate each member of the conversation without any other parties having knowledge of this deception.

***RSA Algorithm.***

| **Key Generation** | |
|---|---|
| Select $p, q$ | $p$ and $q$ both prime, $p \neq q$ |
| Calculate $n = p \times q$ | |
| Calculate $\phi(n) = (p-1)(q-1)$ | |
| Select integer $e$ | $\gcd(\phi(n), e) = 1; 1 < e < \phi(n)$ |
| Calculate $d$ | $d \equiv e^{-1} \pmod{\phi(n)}$ |
| Public key | $PU = \{e, n\}$ |
| Private key | $PR = \{d, n\}$ |

| **Encryption** | |
|---|---|
| Plaintext: | $M < n$ |
| Ciphertext: | $C = M^e \bmod n$ |

| **Decryption** | |
|---|---|
| Ciphertext: | $C$ |
| Plaintext: | $M = C^d \bmod n$ |

Prof.Dr.Hala

# *Attacks On RSA*

*1.Mathematical Attacks on RSA* Mathematical attacks focus on attacking the underlying structure of RSA function. The first intuitive attack is the attempt to factor the modulus N. Because knowing the factorization of N, one may easily obtain M(N), from which d can be determined by d = 1/e mod M(N). However, at present, the fastest factoring algorithm runs in exponential time. Our objective is to survey RSA attacks that decrypts message without directly factoring N.

***2.Elementary attacks***

**2.1 Small Private Key attacks** : To improve the RSA decryption performance in the matter of running-time, Alice might tend to use a small value of da, rather

than a large random number. A small private key indeed will improve performance dramatically, but unfortunately, a attack posed by M.Wiener [shows that a small d leads to a total collapse of RSA cryptosystem. This break of RSA is base on Wiener's Theorem, which in general provides a lower constraint for d. Wiener has proved that Marvin may efficiently find d when d < 1/3 ∗N^(1/4).

2.**2Small Public Key Attacks** : Similar to the private key preferences, to reduce encryption time, it is customary to use a small public key (e), but unlike the previous situation, attacks on small e turn out to be much less effective. The most powerful attacks on small e are based on **Coppersmith's Theorem**. This theorem provides an algorithm for efficiently finding all roots of N that are less than x = N^(1/d). 2.3 Timing Attacks In a timing attack, the attacker eavesdrops during the victim's session and uses statistical analysis of the user's typing patterns and inter-keystroke timings to discern sensitive session information.While timing analysis may not directly result in the decryption of sensitive data, it can be used to gain information about the encryption key and perhaps the cryptosystem in use. Once the attacker has successfully broken an encryption, he or she may launch a replay attack, which is an attempt to resubmit a recording of the deciphered authentication to gain entry into a secure source .

2.4 **Man-in-the-Middle Attack** :A man-in-the-middle attack is designed to intercept the transmission of a public key or even to insert a known key structure in place of the requested public key. From the perspective of the victims of such attacks, their encrypted communication appears to be occurring normally, but in fact the attacker is receiving each encrypted message and decoding it and then encrypting and sending it to the originally intended recipient. Establishment of public keys with digital signatures can prevent the traditional man-in-the-middle attack, as the attacker cannot duplicate the signatures. 2.5 Brute Force: Factoring N, computing *e*th root modulo N Given <C, *e*, N>, Marvin can do a brute force

Prof.Dr.Hala Bahjat Abdul wahab

search to find M. But factoring N involves a much smaller search space and thus, more efficient to do. Factoring is a well researched problem. Pomerance has an interesting introduction into what clever methods can be done. For example, how to factor 8051? Naïve approach is trying all primes up to square root of 8051, or almost 90. Clever trick is to find a square of primes that subtract to the number, example 90^2 – 7^2 = 8051. Thus (90 – 7) * (90 + 7) = 8051; 83 & 97 are the factors

**2.6Fault Analysis Attack** As the name implies, a fault analysis attack depends on the induced or implementation error on a key dependent cryptographic operation. Fault analysis attacks can be mounted against both the secret key and public key cryptographic devices. This attack exploits the likely errors on the RSA decryption or signing operations in cryptographic devices ***Common Modulus Attack*** The idea of the common modulus is that in a session of RSA with several users there is a trusted entity which denes a modulus N and provides for each user a pair of public and private valid RSA keys dened modulo (n), but not the factorization of N. That is, each user Ui gets the public key < ei;N > and the private key < di;N >. Simmons [42] showed that, without needing to factor the modulus, if the same plain text is encrypted and sent to two

Prof.Dr.Hala Bahjat Abdul wahab

users with co-prime public exponents, any other user can decrypt the corresponding cypher text. Theorem 19. Let N = pq be a RSA modulus and let < e1;N >;< e2;N > be two public keys such that (e1; e2) = 1. Suppose a plain text m is encrypted with both public keys. Knowing c1 = me1 (mod N), c2 = me2 (mod N) and the public keys, we can compute m in time polynomial in log(N). Theorem . Let N = pq be a RSA modulus and let < e1;N >;< e2;N > be two public keys such that (e1; e2) = 1. Suppose a plain text m is encrypted with both public keys. Knowing c1 = me1 (mod N), c2 = me2 (mod N) and the public keys, we can compute m in time polynomial in log(N).

Proof. Knowing e1 and e2, we compute integers a1; a2 such that a1e1+a2e2 = 1 using the Extended Euclidean Algorithm.

Now we compute c1^a1. c2^a2= ma1e1.ma2e2 = ma1e1+a2e2 = m (mod N)

***how to use common modulus attack?*** Let Alice, Bob, Chris and Eve communicate over a public network. They encrypt all messages they send using RSA system. Bob and Chris have the RSA modulus nB and nC respectively with nB = nC But different public encryption exponents: eB≠eC. Suppose gcd(eB,eC)=1gcd(eB,eC)=1, and that Alice sends the same secret message to Bob and Chris. Think about this: what does it mean that gcd(eB,eC)=1. Formally that means there exist some s1,s2 such that eBs1+eCs2=1. Say you have two cipher texts (the following math is all done modulo the shared modulus), CB=MeB and CC=MeC. You can do the following:

: Cs1B∗Cs2C=(MeB)s1∗(MeC)s2 =MeBs1∗MeCs2 =MeBs1+eCs2 =M1 =M

Prof.Dr.Hala Bahjat Abdul wahab