COMPUTER SECURITY SEGMENT-2

INSTRUCTOR: SAZID ZAMAN KHAN
ASSISTANT PROFESSOR , CSE, IIUC

Symmetric Cipher Model

- A symmetric encryption scheme has five ingredients (Figure 3.1):
- ■ Plaintext: This is the original intelligible message or data that is fed into the algorithm as input.
- ■ Encryption algorithm: The encryption algorithm performs various substitutions and transformations on the plaintext.

- Secret key: The secret key is also input to the encryption algorithm. The key is a value independent of the plaintext and of the algorithm. The algorithm will produce a different output depending on the specific key being used at the time.
- The exact substitutions and transformations performed by the algorithm depend on the key.

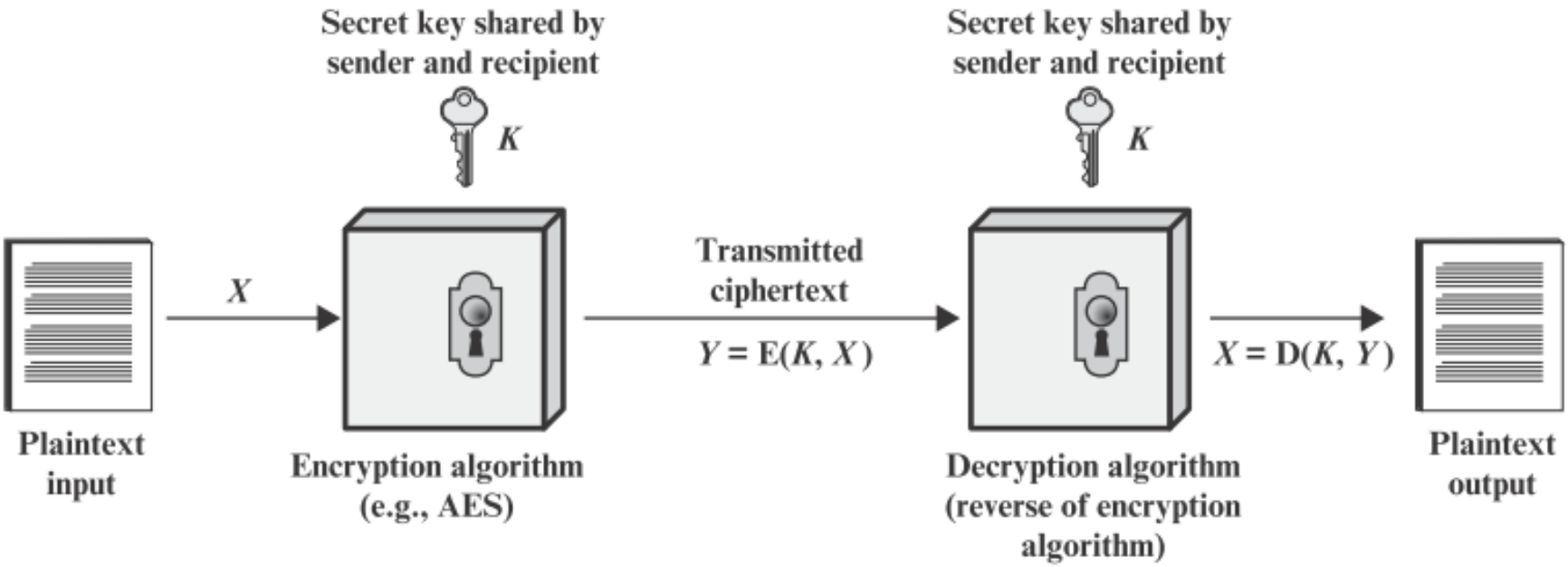


Figure 3.1 Simplified Model of Symmetric Encryption

- Ciphertext: This is the scrambled message produced as output. It depends on the plaintext and the secret key. For a given message, two different keys will produce two different ciphertexts.
- The ciphertext is an apparently random stream of data and, as it stands, is unintelligible.

■ Decryption algorithm: This is essentially the encryption algorithm run in reverse. It takes the ciphertext and the secret key and produces the original plaintext.

- There are two requirements for secure use of conventional encryption:
- The opponent should be unable to decrypt ciphertext or discover the key even if he or she is in possession of a number of ciphertexts together with the plaintext that produced each ciphertext.
- Sender and receiver must have obtained copies of the secret key in a secure fashion and must keep the key secure. If someone can discover the key and knows the algorithm, all communication using this key is readable.

- We do not need to keep the algorithm secret; we need to keep only the key secret.
- This feature of symmetric encryption is what makes it feasible for widespread use.
- The fact that the algorithm need not be kept secret means that manufacturers can and have developed low-cost chip implementations of data encryption algorithms.

- Let us take a closer look at the essential elements of a symmetric encryption scheme, using Figure 3.2. A source produces a message in plaintext, X = [X1, X2, XM]. The M elements of X are letters in some finite alphabet. Traditionally, the alphabet usually consisted of the 26 capital letters. Nowadays,

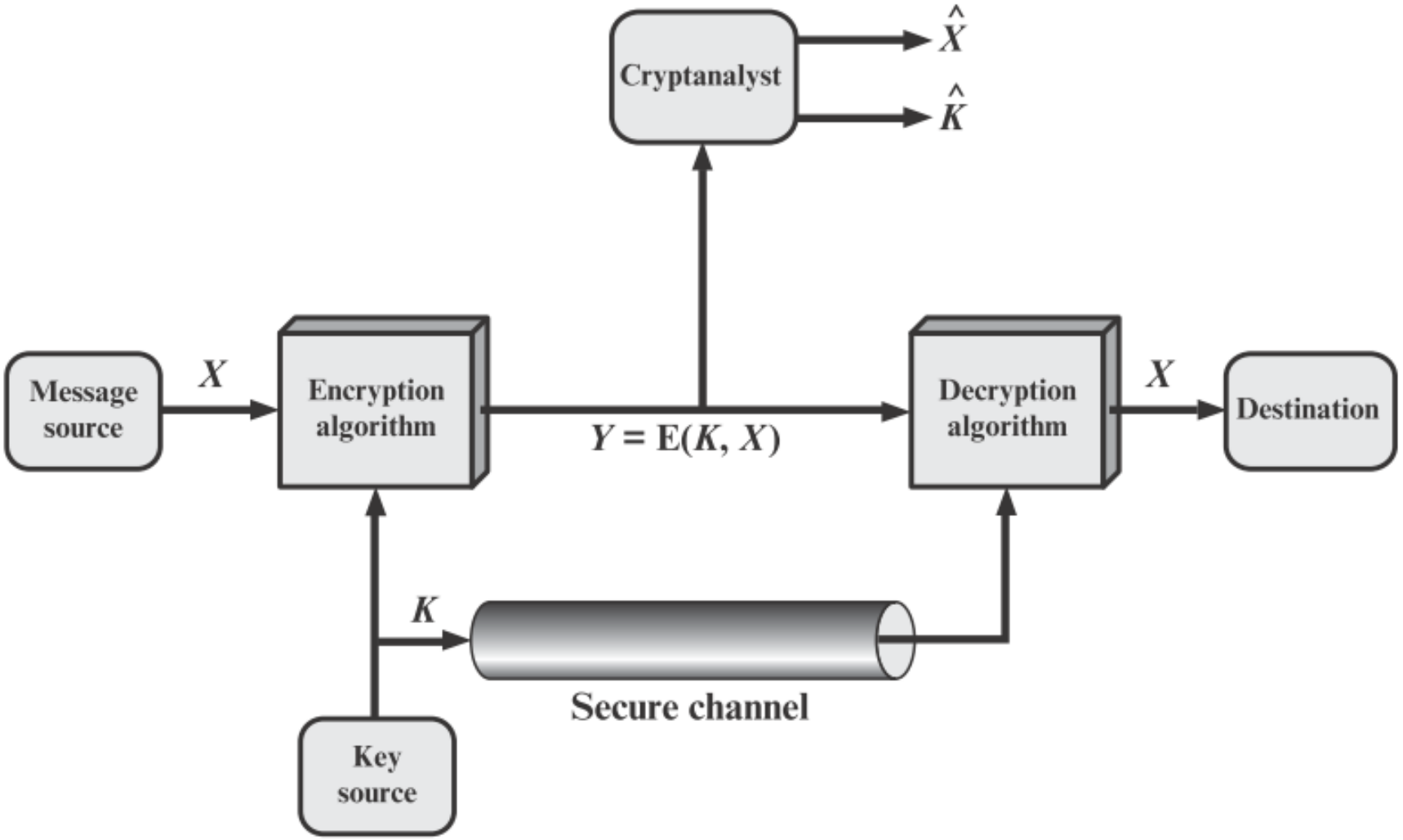
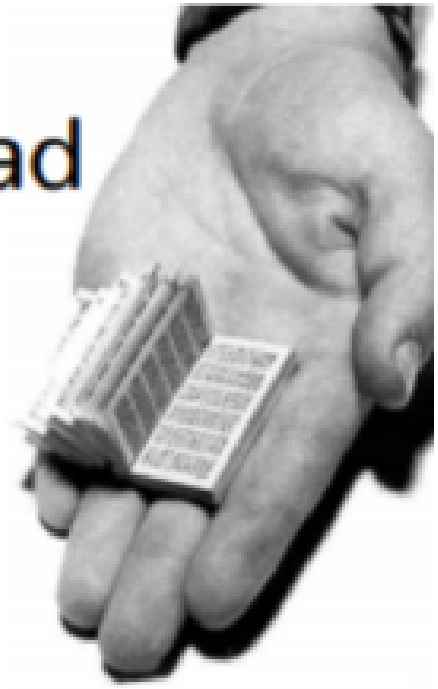


Figure 3.2 Model of Symmetric Cryptosystem

- the binary alphabet {0, 1} is typically used. For encryption, a key of the form K = [K1, K2 , KJ] is generated. If the key is generated at the message source, then it must also be provided to the destination by means of some secure channel.
- Alternatively, a third party could generate the key and securely deliver it to both source and destination. With the message X and the encryption key K as input, the encryption algorithm forms the ciphertext Y = [Y1, Y2 , YN]. We can write this as:
- Y = E(K, X)

- This notation indicates that Y is produced by using encryption algorithm E as a function of the plaintext X, with the specific function determined by the value of the key K.
- The intended receiver, in possession of the key, is able to invert the transformation:
- $X = D(K, Y)$
- An opponent, observing Y but not having access to K or X, may attempt to recover X or K or both X and K. It is assumed that the opponent knows the encryption (E) and decryption (D) algorithms.

Example: One Time Pad



Vernam (1917)

Key:	0	1	0	1	1	1	0	0	1	0
Plaintext:	1	1	0	0	0	1	1	0	0	0
Ciphertext:	1	0	0	1	1	0	1	0	1	0

- **Encryption:** $c = E_k(m) = m \oplus k$
- **Decryption:** $D_k(c) = c \oplus k = (m \oplus k) \oplus k = m$

Letter	ASCII Code	Binary	Letter	ASCII Code	Binary
a	097	01100001	A	065	01000001
b	098	01100010	B	066	01000010
c	099	01100011	C	067	01000011
d	100	01100100	D	068	01000100
e	101	01100101	E	069	01000101
f	102	01100110	F	070	01000110
g	103	01100111	G	071	01000111
h	104	01101000	H	072	01001000
i	105	01101001	I	073	01001001
j	106	01101010	J	074	01001010
k	107	01101011	K	075	01001011
l	108	01101100	L	076	01001100
m	109	01101101	M	077	01001101
n	110	01101110	N	078	01001110
o	111	01101111	O	079	01001111
p	112	01110000	P	080	01010000
q	113	01110001	Q	081	01010001
r	114	01110010	R	082	01010010
s	115	01110011	S	083	01010011
t	116	01110100	T	084	01010100
u	117	01110101	U	085	01010101
v	118	01110110	V	086	01010110
w	119	01110111	W	087	01010111

- A pseudorandom generator inputs a seed and outputs an expanded stream of bits. The output should be pseudorandom in the sense that it is not possible to distinguish the output from a truly randomly generated sequence of bits in a computationally feasible amount of time. The pseudorandom generator can be used to implement a stream cipher.

- The basic idea of a stream cipher is that we apply the pseudorandom generator to the short key, to expand the key into a pseudorandom stream as long as the message. Then we XOR the expanded key with the message to get the ciphertext as the result, just like a one-time pad, except now with a pseudorandomly generated pad.

- We still need to be careful about security with this construction. What if we use the same key twice? Imagine that we use a key k twice to encrypt two different messages, M1 and M2, with the same key stream to obtain two ciphertexts C1 = M1 ⊕ P RG(k) and C2 = M2 ⊕ P RG(k). Then C1 ⊕ C2 = M1 ⊕ M2, that is, the key streams are canceled out.

- From M1 ⊕ M2, what can we learn? The distribution of, say, English text or ASCII characters is very non-random, and an adversary can use frequency analysis or similar to derive likely sequences of plaintext that would result in the observed xored messages.

- If the opponent is interested in only this particular message, then the focus of the effort is to recover X by generating a plaintext estimate X_e.
- Often, however, the opponent is interested in being able to read future messages as well, in which case an attempt is made to recover K by generating an estimate K_e.

Cryptanalysis and Brute-Force Attack

- ■ 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–ciphertext 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.
- ■ 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.

- ■ 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.
- Therefore, all that the users 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 either of the foregoing two criteria are met.

Substitution Cipher- Case 1. Caesar (Shift Cipher) Cipher

- A substitution technique is one in which the letters of plaintext are replaced by other letters or by numbers or symbols.

Caesar Cipher

The earliest known, and the simplest, use of a substitution cipher was by Julius Caesar. The Caesar cipher involves replacing each letter of the alphabet with the letter standing three places further down the alphabet. For example,

```
plain:  meet me after the toga party
cipher: PHHW PH DIWHU WKH WRJD SDUWB
```

Note that the alphabet is wrapped around, so that the letter following Z is A. We can define the transformation by listing all possibilities, as follows:

```
plain:  a b c d e f g h i j k l m n o p q r s t u v w x y z
cipher: D E F G H I J K L M N O P Q R S T U V W X Y Z A B C
```

Let us assign a numerical equivalent to each letter:

a	b	c	d	e	f	g	h	i	j	k	l	m
0	1	2	3	4	5	6	7	8	9	10	11	12

n	o	p	q	r	s	t	u	v	w	x	y	z
13	14	15	16	17	18	19	20	21	22	23	24	25

Then the algorithm can be expressed as follows. For each plaintext letter p, substitute the ciphertext letter C:[2]

$$C = \mathrm{E}(3, p) = (p + 3) \bmod 26$$

A shift may be of any amount, so that the general Caesar algorithm is

$$C = \mathrm{E}(k, p) = (p + k) \bmod 26 \qquad \textbf{(3.1)}$$

---

where k takes on a value in the range 1 to 25. The decryption algorithm is simply

$$p = \mathrm{D}(k, C) = (C - k) \bmod 26 \qquad \textbf{(3.2)}$$

- If it is known that a given ciphertext is a Caesar cipher, then a brute-force cryptanalysis is easily performed: simply try all the 25 possible keys. Figure 3.3 shows the results of applying this strategy to the example ciphertext. In this case, the plaintext leaps out as occupying the third line.Three important characteristics of this problem enabled us to use a brute-force cryptanalysis:
- 1. The encryption and decryption algorithms are known.
- 2. There are only 25 keys to try.
- 3. The language of the plaintext is known and easily recognizable.

- In most networking situations, we can assume that the algorithms are known. What generally makes brute-force cryptanalysis impractical is the use of an algorithm that employs a large number of keys.

KEY	
	PHHW PH DIWHU WKH WRJD SDUWB
1	oggv og chvgt vjg vqic rctva
2	nffu nf bgufs uif uphb qbsuz
3	meet me after the toga party
4	ldds ld zesdq sgd snfz ozqsx
5	kccr kc ydrcp rfc rmey nyprw
6	jbbq jb xcqbo qeb qldx mxoqv
7	iaap ia wbpan pda pkcw lwnpu
8	hzzo hz vaozm ocz ojbv kvmot
9	gyyn gy uznyl nby niau julns
10	fxxm fx tymxk max mhzt itkmr
11	ewwl ew sxlwj lzw lgys hsjlq
12	dvvk dv rwkvi kyv kfxr grikp
13	cuuj cu qvjuh jxu jewq fqhjo
14	btti bt puitg iwt idvp epgin
15	assh as othsf hvs hcuo dofhm
16	zrrg zr nsgre gur gbtn cnegl
17	yqqf yq mrfqd ftq fasm bmdfk
18	xppe xp lqepc esp ezrl alcej
19	wood wo kpdob dro dyqk zkbdi
20	vnnc vn jocna cqn cxpj yjach
21	ummb um inbmz bpm bwoi xizbg
22	tlla tl hmaly aol avnh whyaf
23	skkz sk glzkx znk zumg vgxze
24	rjjy rj fkyjw ymj ytlf ufwyd
25	qiix qi ejxiv xli xske tevxc

Figure 3.3 Brute-Force Cryptanalysis of Caesar Cipher

Another Substitution Cipher

- The goal of the substitution cipher is the encryption of text (as opposed to bits in modern digital systems). The idea is very simple: We substitute each letter of the
- alphabet with another one.
- Example 1.1.
- A → K

Another Substitution Cipher

- Let's determine the key space of the substitution cipher: When choosing the replacement for the first letter A, we randomly choose one letter from the 26 letters of the alphabet (in the example above we chose k).
- The replacement for the next alphabet letter B was randomly chosen from the remaining 25 letters, etc. Thus there exist the following number of different substitution tables:
- key space of the substitution cipher = $26 \cdot 25 \cdots 3 \cdot 2 \cdot 1 = 26! \approx 2^{88}$

Another Substitution Cipher

- Even with hundreds of thousands of high-end PCs such a search would take several decades! Thus, we are tempted to conclude that the substitution cipher is secure. But this is incorrect because there is another, more powerful attack.

Example 1.2. Let's look at another ciphertext:

```
iq ifcc vqqr fb rdq vfllcq na rdq cfjwhwz hr bnnb
            hcc hwwhbsqvqbre hwq vhlq
```

◇

This does not seem to make too much sense and looks like decent cryptography. *However, the substitution cipher is not secure at all!* Let's look at ways of breaking the cipher.

Cryptanalyzing this substitution cipher

- The statistical properties of the plaintext are preserved in the ciphertext. If we go back to the second example we observe that the letter q occurs most frequently in the text. From this we know that q must be the substitution for one of the frequent letters in the English language.

Cryptanalyzing this substitution cipher

- For practical attacks, the following properties of language can be exploited:
- 1. Determine the frequency of every ciphertext letter. The frequency distribution, often even of relatively short pieces of encrypted text, will be close to that of the given language in general.
- In particular, the most frequent letters can often easily be spotted in ciphertexts. For instance, in English E is the most frequent letter (about 13%), Tis the second most frequent letter (about 9%).

Cryptanalyzing this substitution cipher

- 2. The method above can be generalized by looking at pairs or triples, or quadruples, and so on of ciphertext symbols. For instance, in English (and some other European languages), the letter Q is almost always followed by a U. This behavior can be exploited to detect the substitution of the letter Q and the letter U.
- 3. If we assume that word separators (blanks) have been found (which is only sometimes the case), one can often detect frequent short words such as THE, AND, etc.

Cryptanalyzing this substitution cipher

Table 1.1 Relative letter frequencies of the English language

Letter	Frequency	Letter	Frequency
A	0.0817	N	0.0675
B	0.0150	O	0.0751
C	0.0278	P	0.0193
D	0.0425	Q	0.0010
E	0.1270	R	0.0599
F	0.0223	S	0.0633
G	0.0202	T	0.0906
H	0.0609	U	0.0276
I	0.0697	V	0.0098
J	0.0015	W	0.0236
K	0.0077	X	0.0015
L	0.0403	Y	0.0197
M	0.0241	Z	0.0007

Transposition Techniques

- All the techniques examined so far involve the substitution of a ciphertext symbol for a plaintext symbol. A very different kind of mapping is achieved by performing some sort of permutation on the plaintext letters. This technique is referred to as a transposition cipher. The simplest such cipher is the rail fence technique, in which the plaintext is written down as a sequence of diagonals and then read off as a sequence of rows. For example, to encipher the message “meet me after the toga party” with a rail fence of depth 2, we write the following:

```
m e m a t r h t g p r y
  e t e f e t e o a a t
```

The encrypted message is

```
MEMATRHTGPRYETEFETEOAAT
```

This sort of thing would be trivial to cryptanalyze. A more complex scheme is to write the message in a rectangle, row by row, and read the message off, column by column, but permute the order of the columns. The order of the columns then becomes the key to the algorithm. For example,

```
Key:            4 3 1 2 5 6 7
Plaintext:      a t t a c k p
                o s t p o n e
                d u n t i l t
                w o a m x y z
Ciphertext:     TTNAAPTMTSUOAODWCOIXKNLYPETZ
```

Thus, in this example, the key is 4312567. To encrypt, start with the column

- A pure transposition cipher is easily recognized because it has the same letter frequencies as the original plaintext. For the type of columnar transposition just shown, cryptanalysis is fairly straightforward and involves laying out the ciphertext in a matrix and playing around with column positions. Digram and trigram frequency tables can be useful [See the following figure from ref-2].
- The transposition cipher can be made significantly more secure by performing more than one stage of transposition.

Bigram Frequencies

A.k.a digraphs. We can't list all of the bigram frequencies here, the top 30 are the following (in percent %):

```
TH :  2.71      EN :  1.13      NG :  0.89
HE :  2.33      AT :  1.12      AL :  0.88
IN :  2.03      ED :  1.08      IT :  0.88
ER :  1.78      ND :  1.07      AS :  0.87
AN :  1.61      TO :  1.07      IS :  0.86
RE :  1.41      OR :  1.06      HA :  0.83
ES :  1.32      EA :  1.00      ET :  0.76
ON :  1.32      TI :  0.99      SE :  0.73
ST :  1.25      AR :  0.98      OU :  0.72
NT :  1.17      TE :  0.98      OF :  0.71
```

The `english_bigrams.txt` file provides the counts used to generate the frequencies above:

- english_bigrams.txt

Trigram Frequencies

A.k.a trigraphs. We can't list all of the trigram frequencies here, the top 30 are the following (in percent %):

```
THE :  1.81      ERE :  0.31      HES :  0.24
AND :  0.73      TIO :  0.31      VER :  0.24
ING :  0.72      TER :  0.30      HIS :  0.24
ENT :  0.42      EST :  0.28      OFT :  0.22
ION :  0.42      ERS :  0.28      ITH :  0.21
HER :  0.36      ATI :  0.26      FTH :  0.21
FOR :  0.34      HAT :  0.26      STH :  0.21
THA :  0.33      ATE :  0.25      OTH :  0.21
NTH :  0.33      ALL :  0.25      RES :  0.21
INT :  0.32      ETH :  0.24      ONT :  0.20
```

Fig-1: Bigram and Trigram frequencies [2]

```
Key:      4 3 1 2 5 6 7
Input:    t t n a a p t
          m t s u o a o
          d w c o i x k
          n l y p e t z
Output:   NSCYAUOPTTWLTMDNAOIEPAXTTOKZ
```

To visualize the result of this double transposition, designate the letters in the original plaintext message by the numbers designating their position. Thus, with 28 letters in the message, the original sequence of letters is

```
01 02 03 04 05 06 07 08 09 10 11 12 13 14
15 16 17 18 19 20 21 22 23 24 25 26 27 28
```

After the first transposition, we have

```
03 10 17 24 04 11 18 25 02 09 16 23 01 08
15 22 05 12 19 26 06 13 20 27 07 14 21 28
```

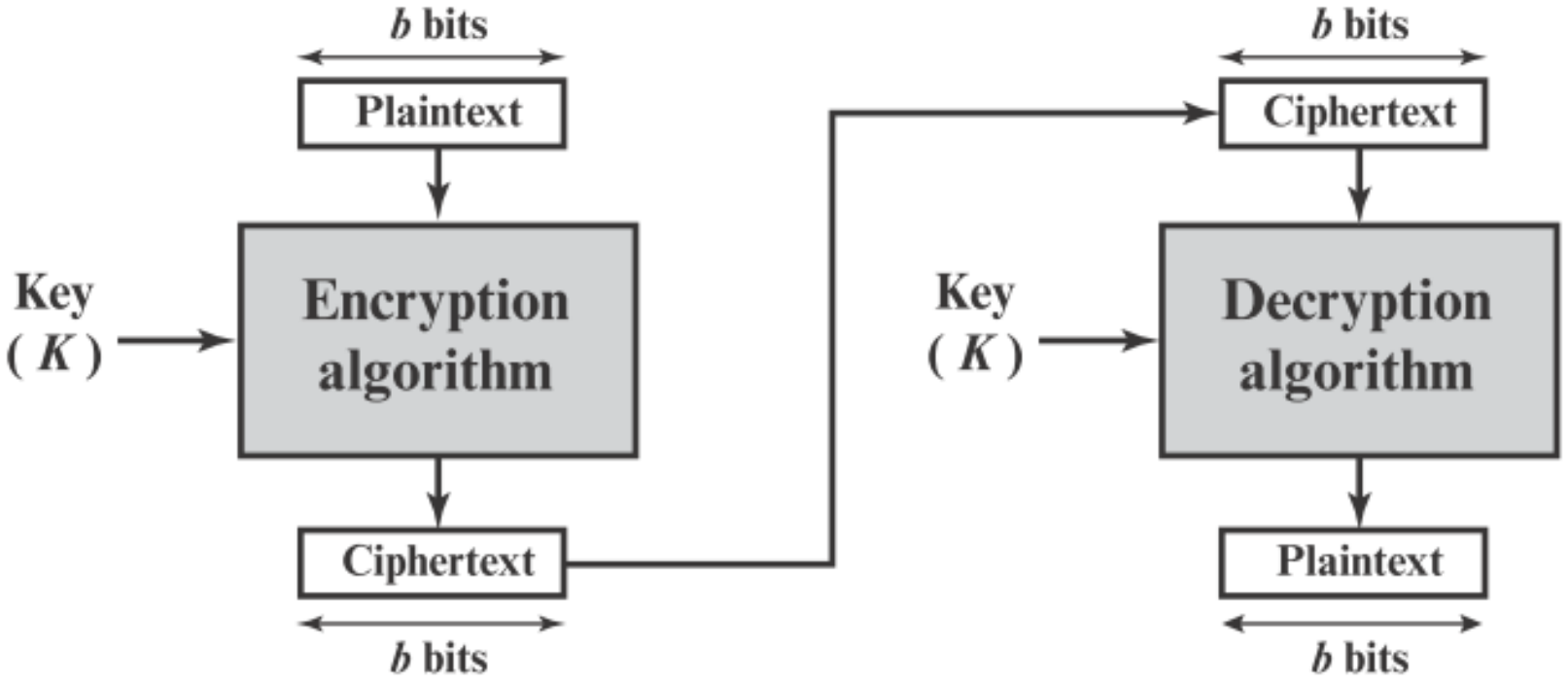
which has a somewhat regular structure. But after the second transposition, we have

```
17 09 05 27 24 16 12 07 10 02 22 20 03 25
15 13 04 23 19 14 11 01 26 21 18 08 06 28
```

This is a much less structured permutation and is much more difficult to cryptanalyze.

Stream and Block Ciphers

- A stream cipher is one that encrypts a digital data stream one bit or one byte at a time.
- A block cipher is one in which a block of plaintext is treated as a whole and used to produce a ciphertext block of equal length. Typically, a block size of 64 or 128 bits is used. As with a stream cipher, the two users share a symmetric encryption key.
- In general, they seem applicable to a broader range of applications than stream ciphers.



(b) Block cipher

Figure 4.1 Stream Cipher and Block Cipher

Block Cipher

- A block cipher operates on a plaintext block of n bits to produce a ciphertext block of n bits. There are 2^n possible different plaintext blocks and, for the encryption to be reversible (i.e., for decryption to be possible), each must produce a unique ciphertext block. Such a transformation is called reversible, or nonsingular.
- The following examples illustrate nonsingular and singular transformations for n = 2:

Reversible Mapping		Irreversible Mapping	
Plaintext	**Ciphertext**	**Plaintext**	**Ciphertext**
00	11	00	11
01	10	01	10
10	00	10	01
11	01	11	01

In the latter case, a ciphertext of 01 could have been produced by one of two plaintext blocks. So if we limit ourselves to reversible mappings, the number of different transformations is $2^n!$.[2]

The reasoning is as follows: For the first plaintext, we can choose any of 2^n ciphertext blocks. For the second plaintext, we choose from among $2^n - 1$ remaining ciphertext blocks, and so on.

Block Cipher

- Figure 4.2 illustrates the logic of a general substitution cipher for n = 4.
- A 4-bit input produces one of 16 possible input states, which is mapped by the substitution cipher into a unique one of 16 possible output states, each of which is represented by 4 ciphertext bits. The encryption and decryption mappings can be defined by a tabulation, as shown in Table 4.1.

Block Cipher

- This is the most general form of block cipher and can be used to define any reversible mapping between plaintext and ciphertext. Feistel refers to this as the ideal block cipher, because it allows for the maximum number of possible encryption mappings from the plaintext block.

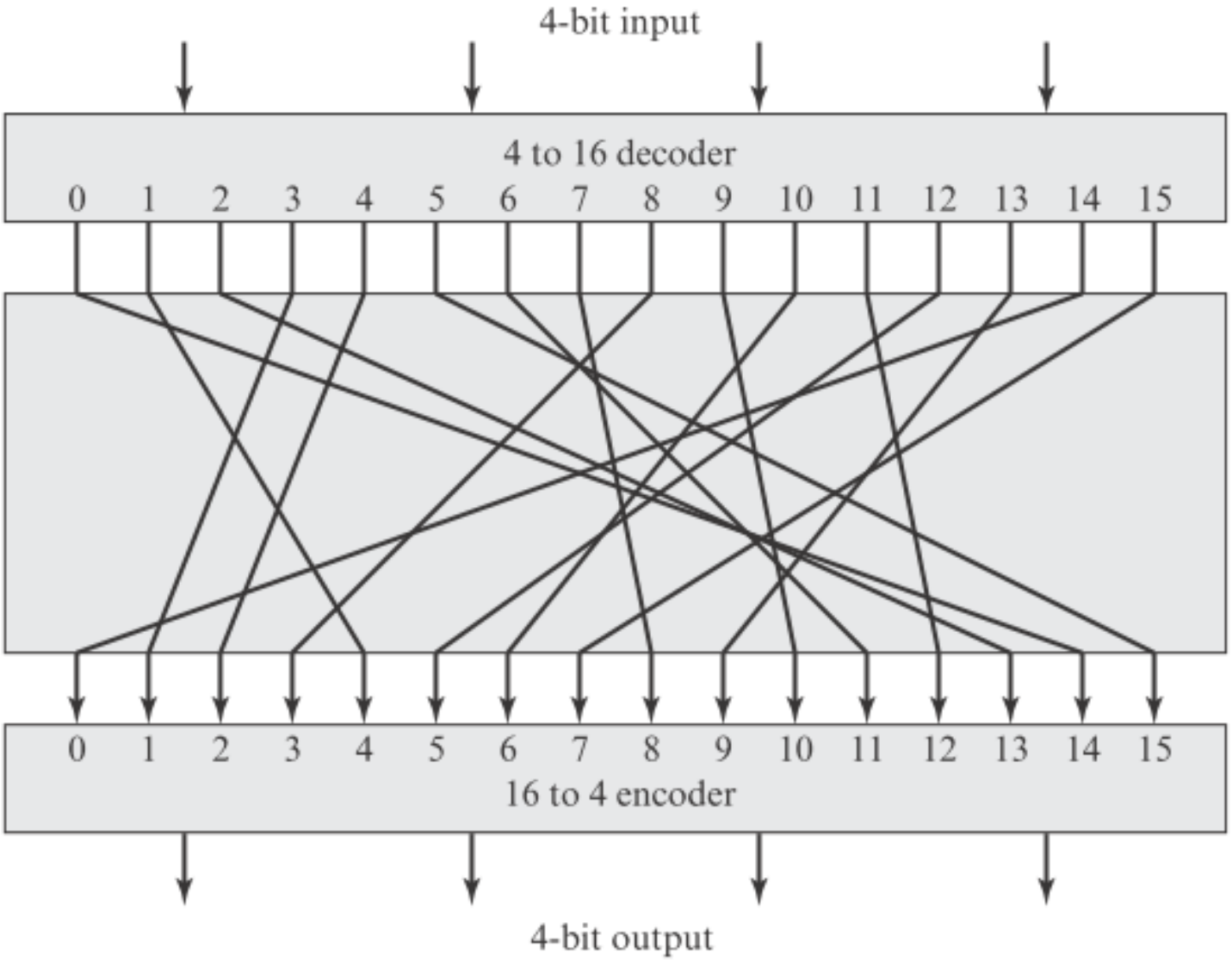


Figure 4.2 General n-bit-n-bit Block Substitution (shown with $n = 4$)

Table 4.1 Encryption and Decryption Tables for Substitution Cipher of Figure 4.2

Plaintext	Ciphertext
0000	1110
0001	0100
0010	1101
0011	0001
0100	0010
0101	1111
0110	1011
0111	1000
1000	0011
1001	1010
1010	0110
1011	1100
1100	0101
1101	1001
1110	0000
1111	0111

Ciphertext	Plaintext
0000	1110
0001	0011
0010	0100
0011	1000
0100	0001
0101	1100
0110	1010
0111	1111
1000	0111
1001	1101
1010	1001
1011	0110
1100	1011
1101	0010
1110	0000
1111	0101

- But there is a practical problem with the ideal block cipher. If a small block size, such as n = 4, is used, then the system is equivalent to a classical substitution cipher. Such systems, as we have seen, are vulnerable to a statistical analysis of the plaintext.
- If n is sufficiently large and an arbitrary reversible substitution between plaintext and ciphertext is allowed, then the statistical characteristics of the source plaintext are masked to such an extent that this type of cryptanalysis is infeasible.

- An arbitrary reversible substitution cipher (the ideal block cipher) for a large block size is not practical, however, from an implementation and performance point of view. For such a transformation, the mapping itself constitutes the key. Consider again Table 4.1, which defines one particular reversible mapping from plaintext to ciphertext for n = 4. The mapping can be defined by the entries in the second column, which show the value of the ciphertext for each plaintext block.

- This, in essence, is the key that determines the specific mapping from among all possible mappings. In this case, using this straightforward method of defining the key, the required key length is (4 bits) * (16 rows) = 64 bits. In general, for an n-bit ideal block cipher, the length of the key defined in this fashion is n * 2^n bits. For a 64-bit block, which is a desirable length to thwart statistical attacks, the required key length is 64 * 2^64 = 2^70 ≈ 10^21 !!!!!(WOW).

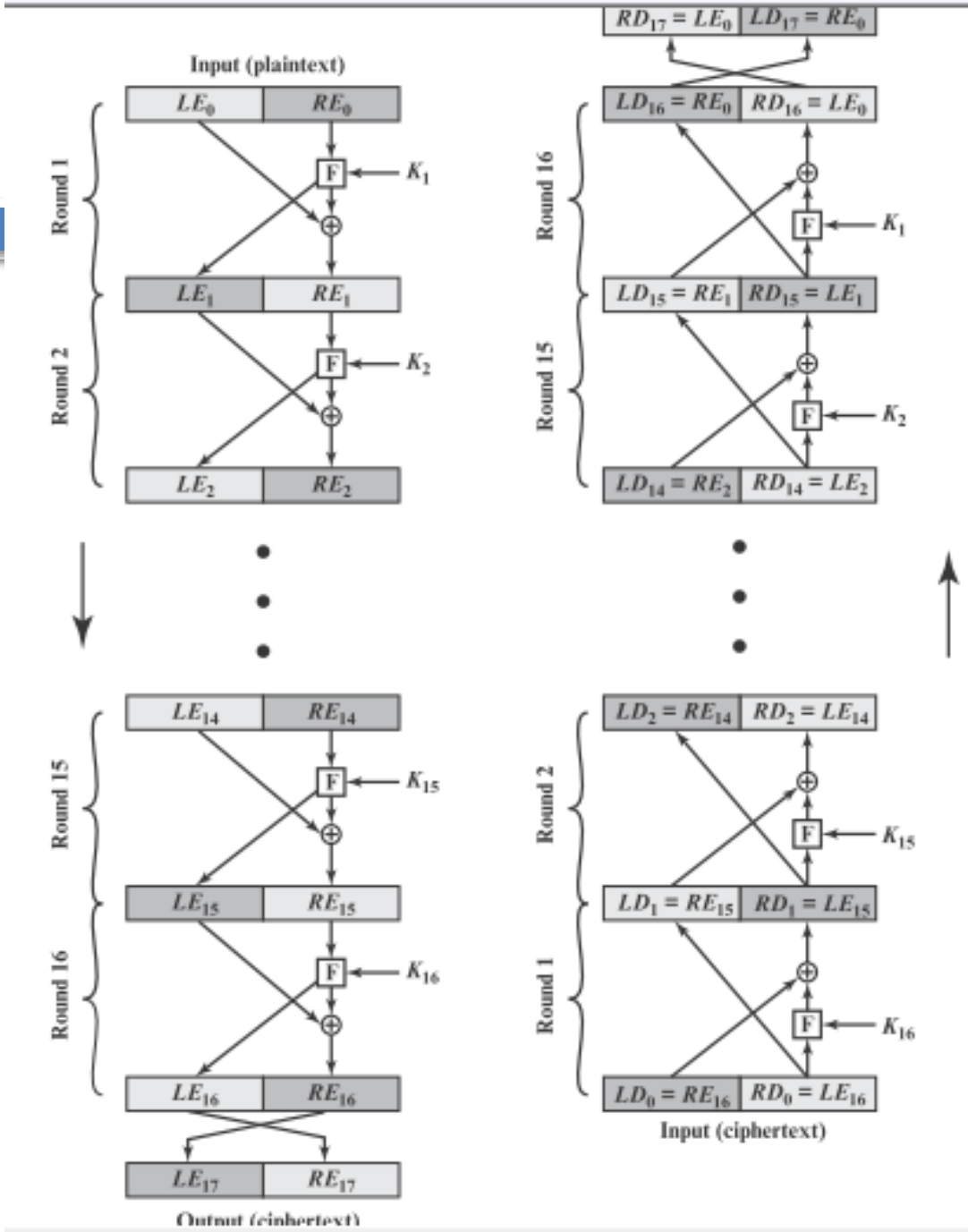
- See Sheet Ideal Block Cipher Example-1.

- In considering these difficulties, Feistel points out that what is needed is an approximation to the ideal block cipher system for large n.

Feistel Cipher Structure

Feistel Cipher Structure The left-hand side of Figure 4.3 depicts the encryption structure proposed by Feistel. The inputs to the encryption algorithm are a plaintext block of length $2w$ bits and a key K. The plaintext block is divided into two halves, LE_0 and RE_0. The two halves of the data pass through n rounds of processing and then combine to produce the ciphertext block. Each round i has as inputs LE_{i-1} and RE_{i-1} derived from the previous round, as well as a subkey K_i derived from the overall K. In general, the subkeys K_i are different from K and from each other. In Figure 4.3, 16 rounds are used, although any number of rounds could be implemented.

All rounds have the same structure. A **substitution** is performed on the left half of the data. This is done by applying a *round function* F to the right half of the data and then taking the exclusive-OR of the output of that function and the left half of the data. The round function has the same general structure for each round but is parameterized by the round subkey K_i. Another way to express this is to say that F is a function of right-half block of w bits and a subkey of y bits, which produces an output value of length w bits: $F(RE_i, K_{i+1})$. Following this substitution, a **permutation** is performed that consists of the interchange of the two halves of the data.[6] This structure is a particular form of the substitution-permutation network (SPN) proposed by Shannon.


Input (plaintext)
LE_0
RE_0
Round 1
Round 2
F
K_1
K_2
LE_1
RE_1
LE_2
RE_2
LE_{14}
RE_{14}
Round 15
Round 16
K_{15}
K_{16}
LE_{15}
RE_{15}
LE_{16}
RE_{16}
LE_{17}
RE_{17}
Output (ciphertext)
$RD_{17} = LE_0$
$LD_{17} = RE_0$
$LD_{16} = RE_0$
$RD_{16} = LE_0$
$LD_{15} = RE_1$
$RD_{15} = LE_1$
$LD_{14} = RE_2$
$RD_{14} = LE_2$
$LD_2 = RE_{14}$
$RD_2 = LE_{14}$
$LD_1 = RE_{15}$
$RD_1 = LE_{15}$
$LD_0 = RE_{16}$
$RD_0 = LE_{16}$
Input (ciphertext)

- Some parameters of Feistel network are Block size, number of rounds, subkey generation algorithm, round function F.

References and disclaimer

- All resources used here are properties of the respective owners. Those are used here for educational purpose.
- **References:**
- 1. Cryptography and Network Security, Principles and Practice (7th Edition)
- -William Stallings.
- 2. https://onlinetoolz.net/bitshift
- 3.https://mathworld.wolfram.com/InversePermutation.html#:~:text=An%20inverse%20permutation%20is%20a,and%2010%20in%20are%20likewise
- 4. Symmetriccrypto-notes.pdf (ucsd.edu)