CSCI361
Computer Security

Secret-key cryptography

Outline

- One-time-pad.
- Perfect secrecy.
- Unicity distance.
- Redundancy.
  - Rates.
- Measuring and increasing security.
- Unconditional versus practical security.
- Improving security.
- Block and stream ciphers (classical).

One-time-pad (Vernam cipher)

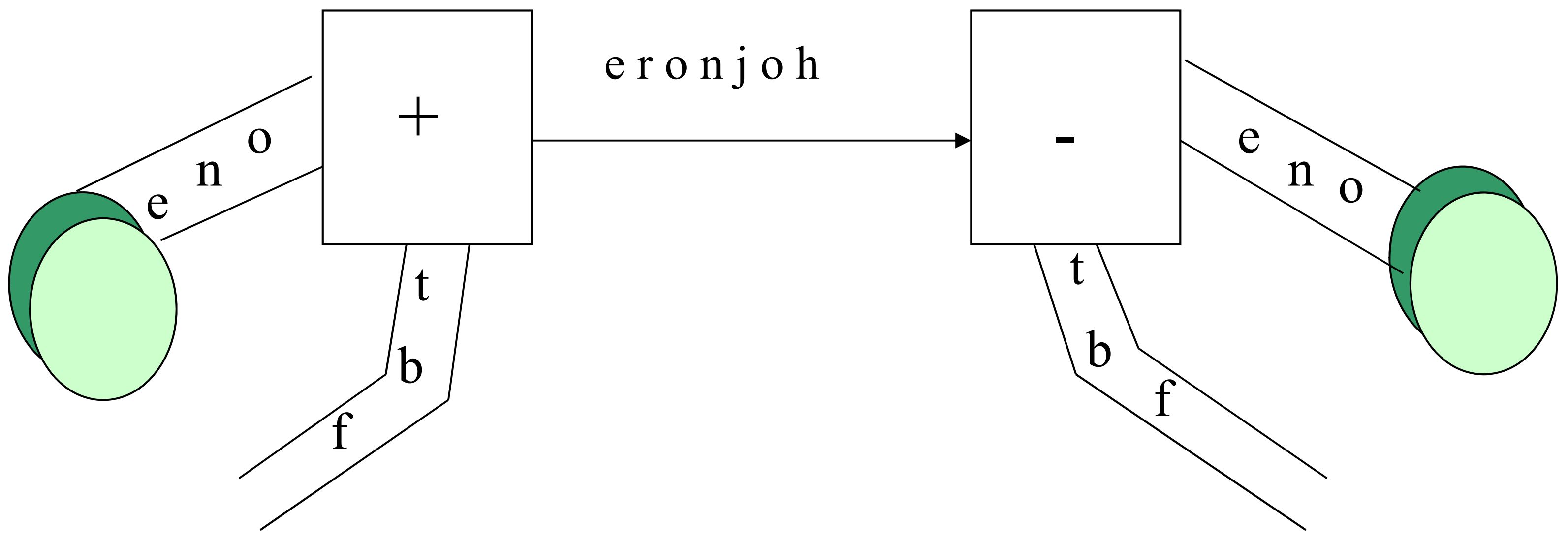
- Gilbert Vernam (1917/8) invented a cipher that was extended by Joseph Mauborgne to give a scheme which was later (Shannon 1949) proved to provide *perfect secrecy*.
  – We shall discuss later what *perfect secrecy* is.
- The cipher is called one-time-pad because the key is written on a long tape and is used only once.
- Encryption is by addition.
- To decrypt a cryptogram, the same key sequence must be used. Decryption is by subtracting the key sequence from the ciphertext.
- The key sequence is a truly random sequence, this was Mauborgne’s contribution.

- In a one-time pad the encryption of each element of plaintext is independent of the encryption on any other piece of plaintext.
- Encryption:

$Y_i = (X_i + K_i) \bmod 26$	Letter by letter.
$Y_i = X_i \oplus K_i$	Bit by bit.

- Decryption:

$X_i = (Y_i + K_i) \bmod 26$	Letter by letter.
$X_i = Y_i \oplus K_i$	Bit by Bit



```
Message   :   ONETIMEPAD
Key on pad:   TBFRGFARFM
Ciphertext:   HOJNOREGFP

  O + T  mod 26 = H
  N + B  mod 26 = O
  E + F  mod 26 = J ...
```

Perfect secrecy (Shannon 1949)

- Model: Transmitter, receiver, enemy.
- Attack: *Ciphertext only*.
- Assumption: *Enemy has unlimited power!*

- **Question 1**: Is it always possible to find the plaintext (or key)?
- **Question 2**: If it is possible, how can we measure the security?

- To find the plaintext in substitution ciphers we may use exhaustive key searches.
- For short cryptograms though, we may find multiple keys give *meaningful plaintexts*.

A	Z	N	P	T	F	Z	H	L	K	Z
m	y	s	t	e	r	y	p	l	a	y
r	e	d	b	l	u	e	c	a	k	e

- For shorter ciphertext more valid plaintext can be found.

H	J	M	T	T
y	a	h	o	o
a	t	r	e	e
m	e	t	o	o
i	w	i	l	l

- If the length of the ciphertext is 1, every decipherment is valid.
- For length around 50, a *unique* plaintext is likely to be found.

Perfect secrecy

- Using the knowledge of the plaintext languages a set of possible plaintexts are determined with certain probability.
- In a system with perfect secrecy, knowledge of the cryptogram does not help the enemy.

  $P(X=x) = P(X=x|Y=y), \qquad \forall\ x,y$

They are just as likely to guess the plaintext associated with a ciphertext **after** they see the ciphertext as they are **before** they see it.

Theorem (Shannon)

In a system with perfect secrecy the number of keys is at least equal to the number of messages.

- This tells us that to achieve perfect secrecy in practice, many key bits must be exchanged. This is not practical.
- How can we measure security then if we know it probably isn't perfect?
- Shannon proposed *unicity distance* as the measure of security.

Unicity distance

- N_0 is the least number of ciphertext characters needed to determine the key uniquely. If there are *E* keys and they are chosen with uniform probability, unicity distance is given by:

$$N_0 = \frac{\log_2 E}{d}$$

where *d* is the *redundancy* of the plaintext language.

Redundancy and rates

- Redundancy of a language is defined in terms of the *rates* of the language.

  d=R-r bits

- The **absolute rate R** of a language is the minimum number of bits to represent each character, assuming characters are equally likely and emitted independently. For an alphabet of size A, $R=\log_2 A$.
- The **true rate r** of a language is the average number of bits required to represent characters of that language. This uses the real probability distribution of characters.
- For English; $R \approx 4.7$ bits, $r \approx 1\text{-}1.5$ bits, $d \approx 3.2$ bits.
- True rate is always smaller than absolute rate, and the difference is the redundancy.

- All natural languages are redundant:

mst ids cn b xprsd n fwr ltrs,

bt th xprnc s mst nplsnt.

- This sentence is readable because we can fill all the missing characters: that is, all the missing characters are redundant.

- Redundancy is related to structure.
- A truly random source has no redundancy.

  mmhfsdacxnvfdvvdfpnfuipawedka
- Every character in this string is necessary: if one of them is omitted the information it carries is lost and cannot be recovered.

- Redundancy occurs because of the non-uniform letter frequencies, bigram (trigram …) frequencies and other (e.g. grammatical) structures of the language.

Measuring security

- We can use unicity distance as a measure to compare the security of various ciphers.
- Recall that we presented security as being about protecting *assets* against possible *threats*.
- Recall also the security principles we described, in particular the *Principle of Adequate Protection,* or the concept of *timeliness.*

- For monoalphabetic ciphers we have:

  $E=26!$ $P(K)=1/E$

  $\log_2 E \approx 88.4$ bits

  $N_0 \approx 88.4/3.7 \approx 23.9$ characters.

Therefore cryptograms (of English text plaintext) 24 or more letters can be deciphered (in general).

- For the one-time-pad:

  X = Message space is the set of English text of length k.

  Z = The key space which is the set of sequences of length k over the English alphabet.

Keys are chosen randomly with uniform distribution:

$E=26^k$, $\log_2 E \approx 4.7k$ so that

$N_0 \approx (4.7)/(3.7)k \approx 1.27k$

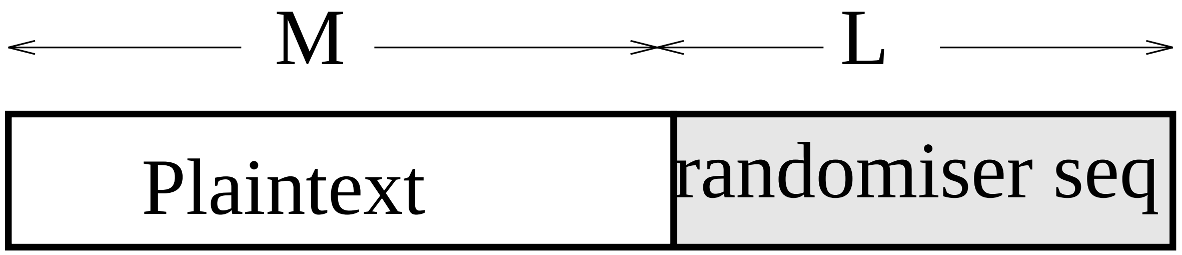
Therefore, a *unique* solution cannot be obtained even if all the characters of the ciphertext are intercepted.

Increasing security

- Studying perfect security suggests ways of increasing security of a cipher systems.
- We could *increase the size of the key space*.
  - The increase should be such that for a ciphertext Y, using a randomly chosen key Z for decryption, a random message X should be generated.

- We could *reduce the redundancy of the plaintext language*.
  - This can be achieved using data compression.
  - A perfect data compression algorithm will result in $d=0$ and hence $N_0 \rightarrow \infty$, i.e. makes the cipher unbreakable.

- We also could *use a randomiser to reduce the redundancy of the plaintext language*.



- The new redundancy is M/(L+M)d.

Unconditional versus practical security

- A system is *unconditionally secure* if it is secure against an enemy with unlimited resources (time, memory).

- In an unconditionally secure system, protection is provided in an *information theoretic* sense. That is, the reason that the enemy cannot break the cipher is that they do not have enough information.

- In practice however:
  - Enemies have limited resources.
  - Data has a limited life time anyway, *timeliness.*
  - The enemy may be able to access other types of attack:
    - Social attacks aimed at getting passwords.
    - Plaintext/ciphertext pair attacks.
- The one-time-pad is the simplest only encryption system with unconditional security, but it requires many key bits and security is under ciphertext only attack.

Practical or computational security

- Computational or practical security attempts to provide a more realistic model for a secure system than unconditional security.

- After the unicity distance is reached, there is a unique plaintext for a given ciphertext.

- In practical security the main objective of the cryptographer is to make sure the amount of work required to find that plaintext stays high even after the unicity distance.

  A cipher is computationally secure if the difficulty of the best attack exceeds the computational capability of the cryptanalyst (attacker!).

Principles suggested by Shannon’s work

- In the design of a good algorithm we must make methods of statistical analysis ineffective:
- **Diffusion**: Diffuse the statistical structure of the plaintext into long range statistics : statistics involved long combinations of letters.
- **Confusion**: The relationship between plaintext, ciphertext and key must be complex one so that knowledge of plaintext/ciphertext pairs cannot easily be used to find the key. In other words, Y=E(X,Z) should be a *nonlinear* and complex function.

Diffusion

- For example: for $X = X_1 X_2 \ldots$ we could use the relation

$$Y_n = E\left(\sum_{i=1}^{s} X_{n+i}\right)$$

  – This diffuses by averaging *s* consecutive letters.

  – For s=2, English alphabet, and additive cipher with shift 5 we have:

t	e	s	t	c	a	s	e
19	4	18	19	2	0	18	4
23	22	11	21	2	18	22	23
2	1	16	0	7	23	1	2
C	B	Q	A	H	X	B	C

The avalanche effect

- A desirable property for encryption functions is that small changes in either plaintext or key should result in significant changes in the ciphertext.
- The avalanche effect is in place if this is the case.
- For example, a cipher might be said to have the avalanche effect if a changing a single bit (in either the key or plaintext) results in half the output being different.

Confusion

- A simple additive cipher is linear.

$$Y = X \oplus Z$$

- We could use a more complex relationship. In modern ciphers the complex relationship is obtained by combining simple elements.
  - We will start to look at this soon.

Some numbers

- It is useful to get an idea about how complex the relationship needs to be. To do this we need to look at the amount of work that we must produce for a cryptographic algorithm to be secure. Consider this:
  - 10^{24} operations with $280.6*10^{12}$ operations per second (IBM Blue Gene/L October 2005) would take 113 years.
  - A system is probably safe enough to be considered secure if this amount of computation is the required "cost" of breaking the system.

- The number of operations or storage required depends on the attack. For a secure algorithm the number remains large even for the best (known) attack.
- Monoalphabetic ciphers are insecure because:
  - Although there are many keys, statistical methods allows many keys to be quickly eliminated.

Block and stream ciphers

- All the early ciphers we have studied so far are examples of *stream ciphers*.
- In stream ciphers characters are encrypted one at a time.

- A second class of ciphers are *block ciphers.* In block ciphers plaintext characters are grouped in blocks and then each block is encrypted.
- This provides **diffusion** across the block.

Playfair cipher (1850)

- The key is specified by a 5 by 5 matrix of 25 letters, with J omitted.

H	A	R	P	S
I	C	O	D	B
E	F	G	K	L
M	N	Q	T	U
V	W	X	Y	Z

- A pair of plaintext X_1X_2 is encrypted to Y_1Y_2 according to the following rules:
  - X_1X_2 in the same row: Y_1 and Y_2 are the two characters to the right of X_1 and X_2 (cyclic).
  - X_1X_2 in the same column: Y_1 and Y_2 are the two characters below X_1 and X_2 (cyclic).
  - $X_1=X_2$: A separating character (anything, X, Z …) is inserted in the plaintext between the two. This is really a pre-encryption phase.
  - X_1X_2 different rows & columns: Y_1 and Y_2 are the other two corners of the rectangle with X_1 and X_2 as corners. The direction we shall take for the example is anticlockwise from the first element to the second but this is arbitrary although we must be consistent.

H	A	R	P	S
I	C	O	D	B
E	F	G	K	L
M	N	Q	T	U
V	W	X	Y	Z

TH	IS	IS	AT	RI	AL
PM	BH	BH	NP	HO	FS

Transposition: Columnar

- Plaintext is written into a matrix by rows. The ciphertext is obtained by reading off the columns.

c	o	m	p	u
t	e	r	s	e
c	u	r	i	t
y	x	x	x	x

- plaintext : Computer Security
- ciphertext : CTCYOEUXMRRXPSIXUETX

Transposition: Periodic

- Break plaintext into blocks of size *d* and permutes characters in a block.

i	1	2	3	4	5
F(i)	5	1	4	3	2

compu	terse	curit	y
UCPMO	ETSRE	TCIRU	Y

- The relative frequency of ciphertext characters is the same as the plaintext.
- The cipher is breakable by *anagramming* - the process of restoring disarranged set of letters using the frequency of bigrams and trigrams.

CSCI361
Computer Security

Modern Secret-key cryptography I

Outline

- Block ciphers.
  - Design principles.
  - Iterated ciphers.
- Data Encryption Standard (DES) development.
- DES.
  - Structure.
  - Algorithms.
  - S-Boxes.
- Feistel ciphers/networks/structure.

Modern block ciphers

- In a block cipher algorithm, plaintext bits are grouped into blocks and then processed.
- We have already seen a classical block ciphers (the Playfair cipher), as well as the transposition building block.
- We are going to begin looking at a more formal presentation and analysis of block ciphers.
- Two important parameters of such algorithms are *block length* and *key size*.

Key	000	001	010	011	100	101	110	111
0	001	111	110	000	100	010	101	011
1	001	110	111	100	011	010	000	101
2	001	000	100	101	110	111	010	011
3	100	101	110	111	000	001	010	011
4	101	110	100	010	011	001	011	111

- **The block length is 3.** The cipher takes an arbitrary binary string of length 3, and uses the specified keys to produce 3 bits of output.
- We need **3 bits** to specify the key (1 of 5).
- The number of possible substitutions on an alphabet of size 8 is 8! (40320, about 16 bits). By using a subset of all possible substitutions we have reduced the size of the key.

- One important way of thinking about, or representing, block ciphers is the following:

A block cipher algorithm, with block size *b* bits, describes a mapping to an indexed subset of the set of all possible substitutions on an alphabet of size 2^b.

- The idea is to have an efficient way of generating a “nice” subset.

- We already know that for a secure cipher the indexed set should look random.
- Looking at the example we had before we see that if the enemy receives 001, they will know that the plaintext is either 000 or 101. This isn’t very good.

Key	000	001	010	011	100	101	110	111
0	001	111	110	000	100	010	101	011
1	001	110	111	100	011	010	000	101
2	001	000	100	101	110	111	010	011
3	100	101	110	111	000	001	010	011
4	101	110	100	010	011	001	011	111

A secure block cipher…

- … needs a **block length** large enough to deter statistical attacks.
  - Length 64 is usually considered large enough to make frequency analysis infeasible.
  - Generally, the larger the block length the longer the processing time. Furthermore the longer you have to wait before you can begin processing.
- … needs a **key space** large enough to make exhaustive key searches infeasible.
  - On the other hand, keys should be short enough so that key generation, distribution and storage can be efficiently managed.

Design principles for block ciphers

1. Security principles.
   - Confusion.
   - Diffusion.
2. Implementation principles.

   Two implementation methods are available.
   - Software – flexible, low cost.
   - Hardware (VLSI) – high speed.

     (Very Large Scale Integration)

Design principles for software

- Cipher operations should be on a sub-block size of length *natural* to the software, generally 16 or 32 bits.
- Operations should be simple and easy to implement. They should use basic instructions of the standard processors, that is addition, multiplication, shifting …
  - Bitwise permutations, for example, are hard to implement.

Design principles for hardware

- Encryption and decryption should be similar.
  - Ideally they should differ only in the way of using the secret key.
- The cipher should have a regular modular structure to facilitate VLSI implementation.

Iterated ciphers

- A block cipher is called an *iterated cipher* if it is based on a simple function f, repeated (or iterated) several times.
- Each iteration is called a *round*. The output of each round is a function of the output of the previous round and a sub-key derived from the full secret key by a key-scheduling algorithm.
- Decryption is performed by using the inverse of the round functions and the sub-keys in reverse order.
- A particularly interesting class of functions in this context are the involution functions. These functions are self-invertible. Employing such functions allows encryption and decryption modules to be the same, but keyed in the reverse order.

Involution example

- Consider that a function f acts on binary blocks of length 3.

$$f = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 1 & 3 \end{bmatrix}$$

- f(101)=011, f(f(101))=101
- f^{-1}=f, f(f(x))=x
- Another example is f(x) = x $\oplus$ 11, for x a binary block of length 2.

Towards DES: Preliminaries

- It became clear in the 1970's that there was a need for a standard encryption algorithm. There were several reasons for this:
  - Advances in information technology and the need for security to support this. Government and Commercial parties were beginning to use computers extensively.
  - Required assurance could not be provided by ad-hoc algorithms.
  - Different devices had to be able to exchange encrypted information.

- The standard needed to have the following properties:
  - It needed a high level of security.
  - It needed to be completely specified. In accordance with Kirchoff’s Law the security shouldn’t rely on a hidden part of the algorithm.
  - It needed to be economical to implement.
  - It needed to be adaptable to diverse application.

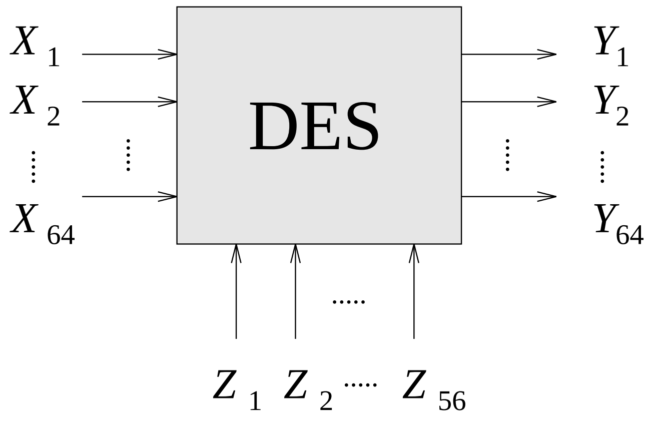
- In 1973 National Bureau of Standards published a solicitation for cryptosystems in the Federal Register. This lead ultimately to the development of the ***Data Encryption Standard***, or ***DES***.

- DES was developed at IBM based on an earlier cipher system known as Lucifer.

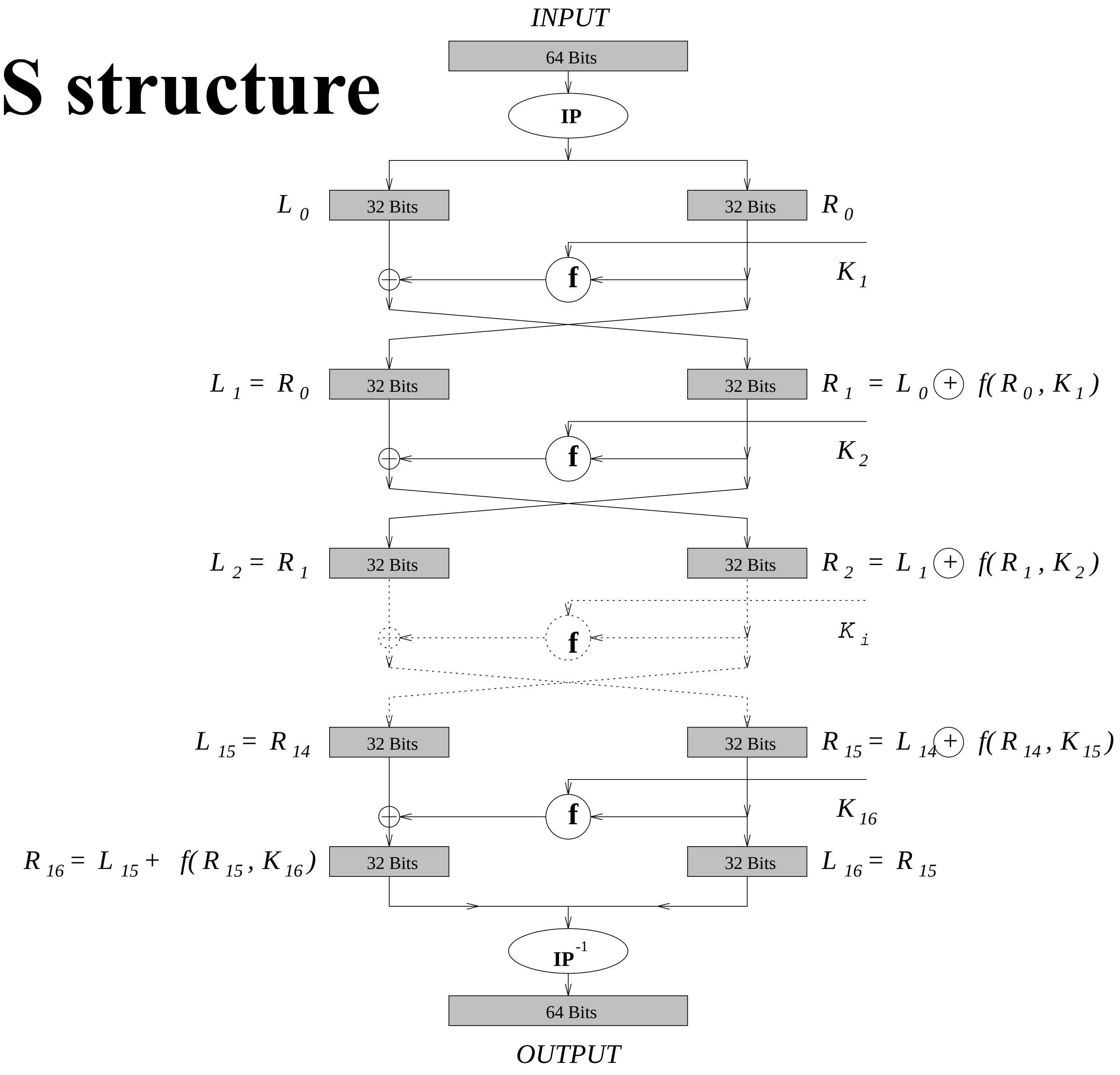
Data Encryption Standard (DES)

- DES (1977) is one of the most widely used, if not the most widely, and perhaps the most controversial cipher.
- It is a block cipher.
- It takes an input block of 64 bits and maps it into an output block of 64 bits using a key of 56 bits.

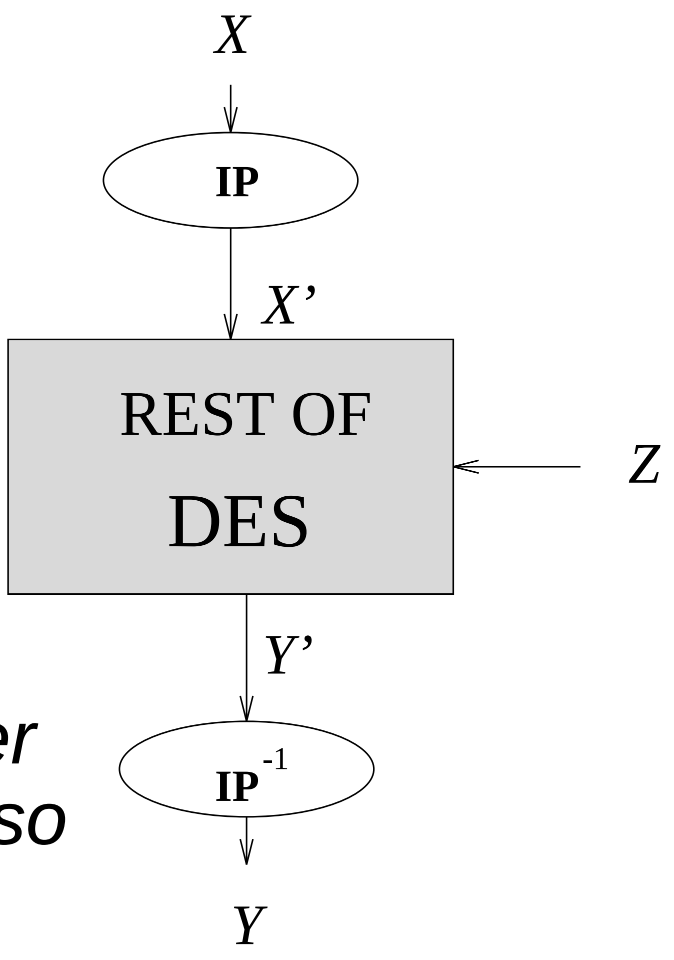
  $|X|=|Y|=2^{64}, \quad |Z|=2^{56}.$



DES structure



- The function *f* if a non-linear transformation, and is the source of the cryptographic strength of DES.
- IP is the initial permutation, and has no cryptographic significance.
  - *This is because if X and X' uniquely determine one another in a way known to the enemy, so do Y and Y'.*
- IP is used to facilitate getting bits onto the chip in VLSI DES.



Specification of IP

- L0 and R0 are the initial left half and right half of the post-IP data.

$$L_0\left\{\begin{array}{cccccccc}58 & 50 & 42 & 34 & 26 & 18 & 10 & 2\\ 60 & 52 & 44 & 36 & 28 & 20 & 12 & 4\\ 62 & 54 & 46 & 38 & 30 & 22 & 14 & 6\\ 64 & 56 & 48 & 40 & 32 & 24 & 16 & 8\end{array}\right.$$

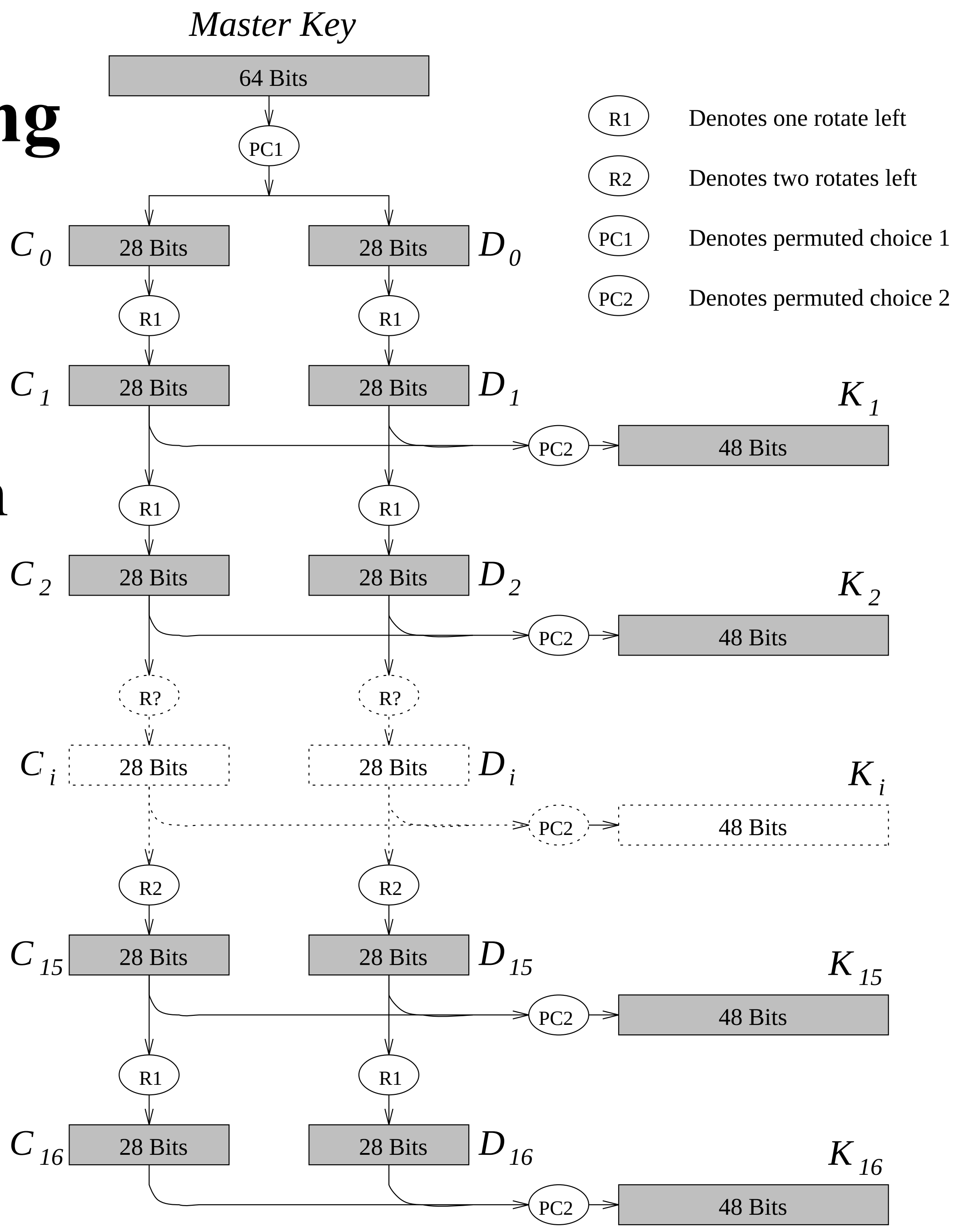
$$R_0\left\{\begin{array}{cccccccc}57 & 49 & 41 & 33 & 25 & 17 & 9 & 1\\ 59 & 51 & 43 & 35 & 27 & 19 & 11 & 3\\ 61 & 53 & 45 & 37 & 29 & 21 & 13 & 5\\ 63 & 55 & 47 & 39 & 31 & 23 & 15 & 7\end{array}\right.$$

This means: Bit 58 in the original input X becomes bit 1 of IP(X)

…

Key scheduling

This produces 16 subkeys from 56 bits of key.



56 bits key?

- You might have noticed the key size at the top is 64 bits. What is going on?
- DES expects 64 bits of key, but only uses 56. The remainder are generally used for parity checking.
- 56 bits isn’t actually very big as keys go, actually it is too small now and was even a concern at the time.
- The size was chosen with chip implementation in mind.

Permuted choice one

- PC-1 extracts 28 bits for C_0 and 28 for D_0.

	57	49	41	33	25	17	9
C_0	1	58	50	40	34	26	18
	10	2	59	51	43	35	27
	19	11	3	60	52	44	36

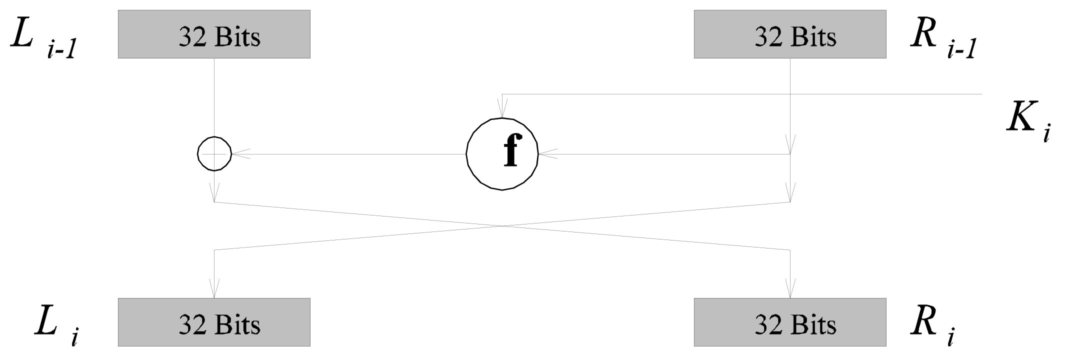
	63	55	47	39	31	33	15
D_0	7	62	54	46	38	30	22
	14	6	61	53	45	37	29
	21	13	5	28	20	12	4

Permuted choice two

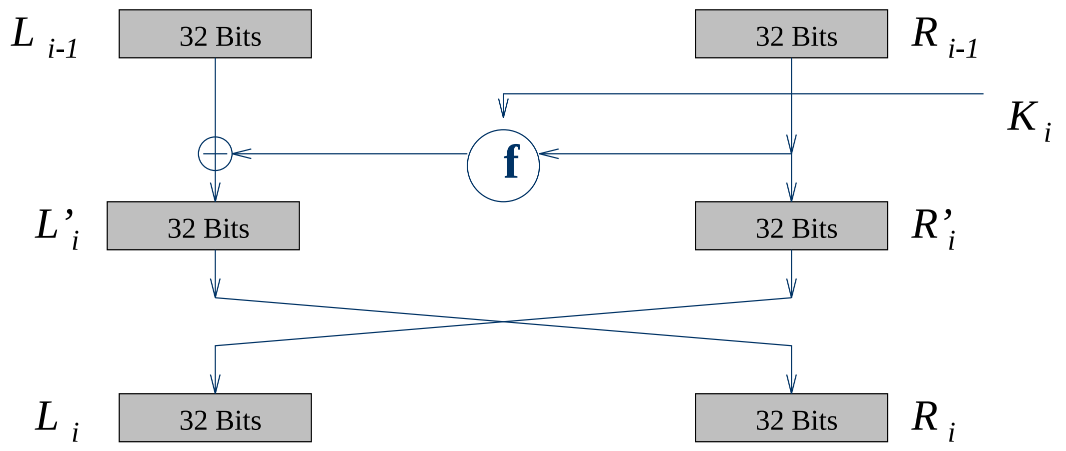
- PC-2 extracts 48 bits for a subkey.

14	17	11	24	1	5	3	28
15	6	21	10	23	19	12	4
26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40
51	45	33	48	44	49	39	56
34	53	46	42	50	36	29	32

The round structure of DES



Why? This can be redrawn as …



… the Feistel structure!

The Feistel structure

- DES has a Feistel structure. This means…
  - Multiple rounds with ordered sub-keys.
  - The input is split into two parts.
  - The right hand side is transformed by a *round function f* in each round, before being combined with the left hand side using an XOR operation.
  - The left and right hand sides are switched after each round to obtain the input to the next round.

- Feistel (1973) proposed this structure as a means of combining substitution and permutation elements to provide confusion and diffusion (as described by Shannon).

Parameters in Feistel ciphers

- A Feistel cipher or network, that is a cipher with the Feistel structure, has various parameters which can characterise them:
  - Block size.
  - Key size.
  - Number of rounds. A single round of a Feistel structure is not expected to provide enough security, however the cipher should be such that multiple rounds of it provides more security.
  - Subkey generation algorithm.
  - Round function *f*. It doesn’t have to be invertible, since it is only used in one direction.

- Each round of DES is a product cipher whose first component cipher is a substitution cipher **F(.)**, and whose second component cipher is the transposition **T(.)**.
- In the last round **T** is omitted.
- The substitution cipher is an *involution*, this is characteristic of Feistel networks.

$$
\begin{aligned}
\mathbf{F_i(L_{i-1},R_{i-1})} &= \mathbf{(L_{i-1} \oplus f(k_i,R_{i-1}),R_{i-1})} \\
\mathbf{F_iF_i(L_{i-1},R_{i-1})} &= \mathbf{F_i(L_{i-1} \oplus f(k_i,R_{i-1}),R_{i-1})} \\
&= \mathbf{(L_{i-1} \oplus f(k_i,R_{i-1}) \oplus f(k_i,R_{i-1}),R_{i-1})} \\
&= \mathbf{(L_{i-1},R_{i-1})}
\end{aligned}
$$

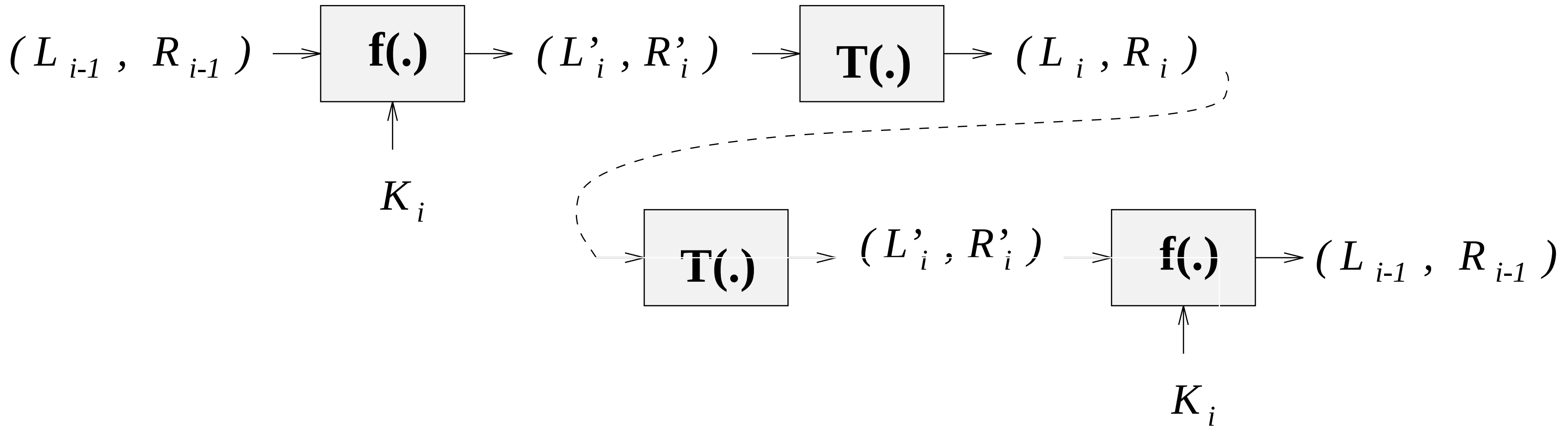
DES Decryption

- The transposition cipher T(.) is an involution too.

$$T(L_i^{'}, R_i^{'}) = (R_i^{'}, L_i^{'})$$

$$T(T(L_i^{'}, R_i^{'})) = T(R_i^{'}, L_i^{'}) = (L_i^{'}, R_i^{'})$$

- To decrypt one round.



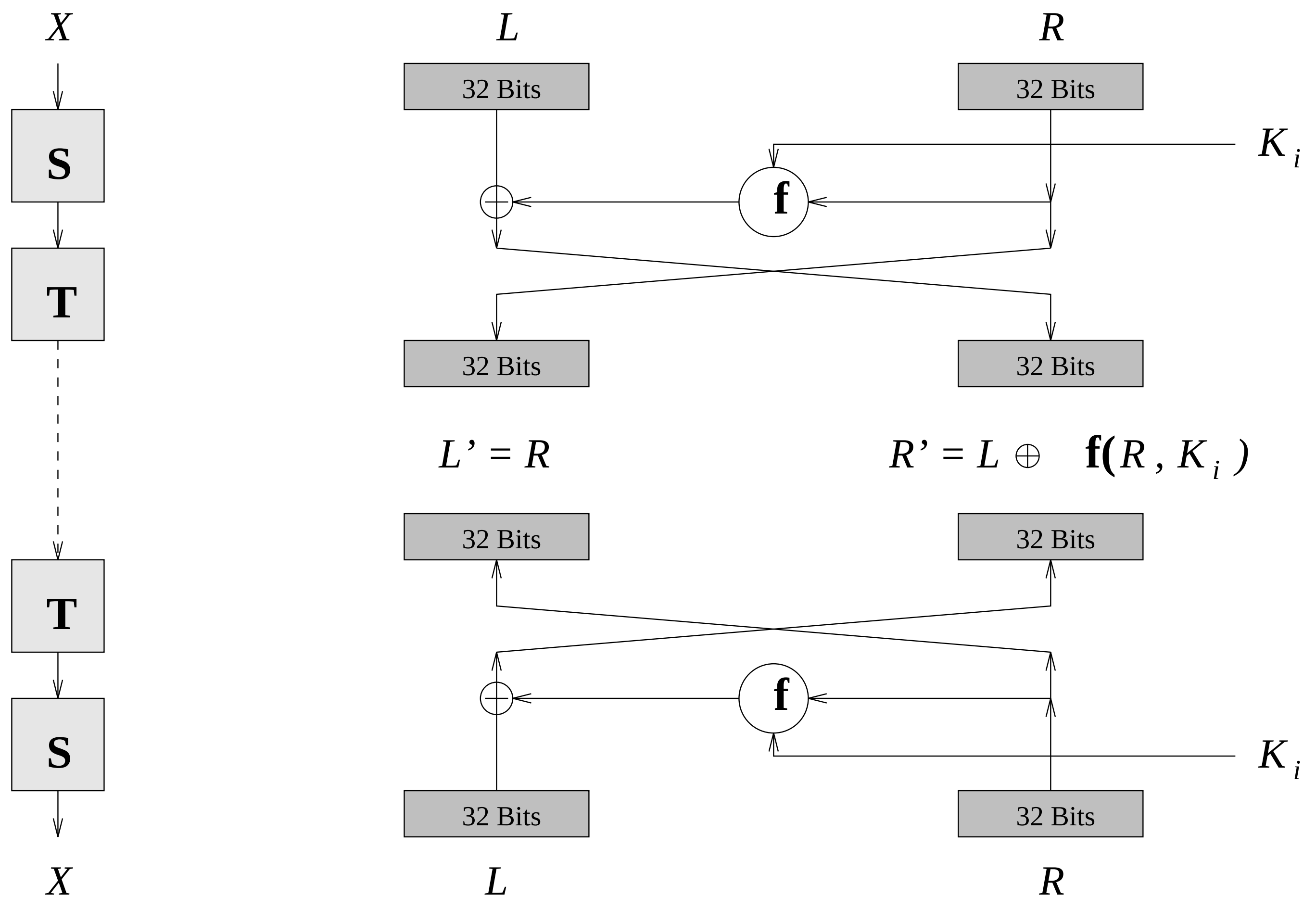
- The full 16 round DES algorithm is a product cipher composed of 16 rounds, each round a substitution and a transposition (except the last).

  $DES=(IP)^{-1}F_{16}TF_{15}T\ldots F_{2}TF_{1}(IP)$

- Decryption is similar:

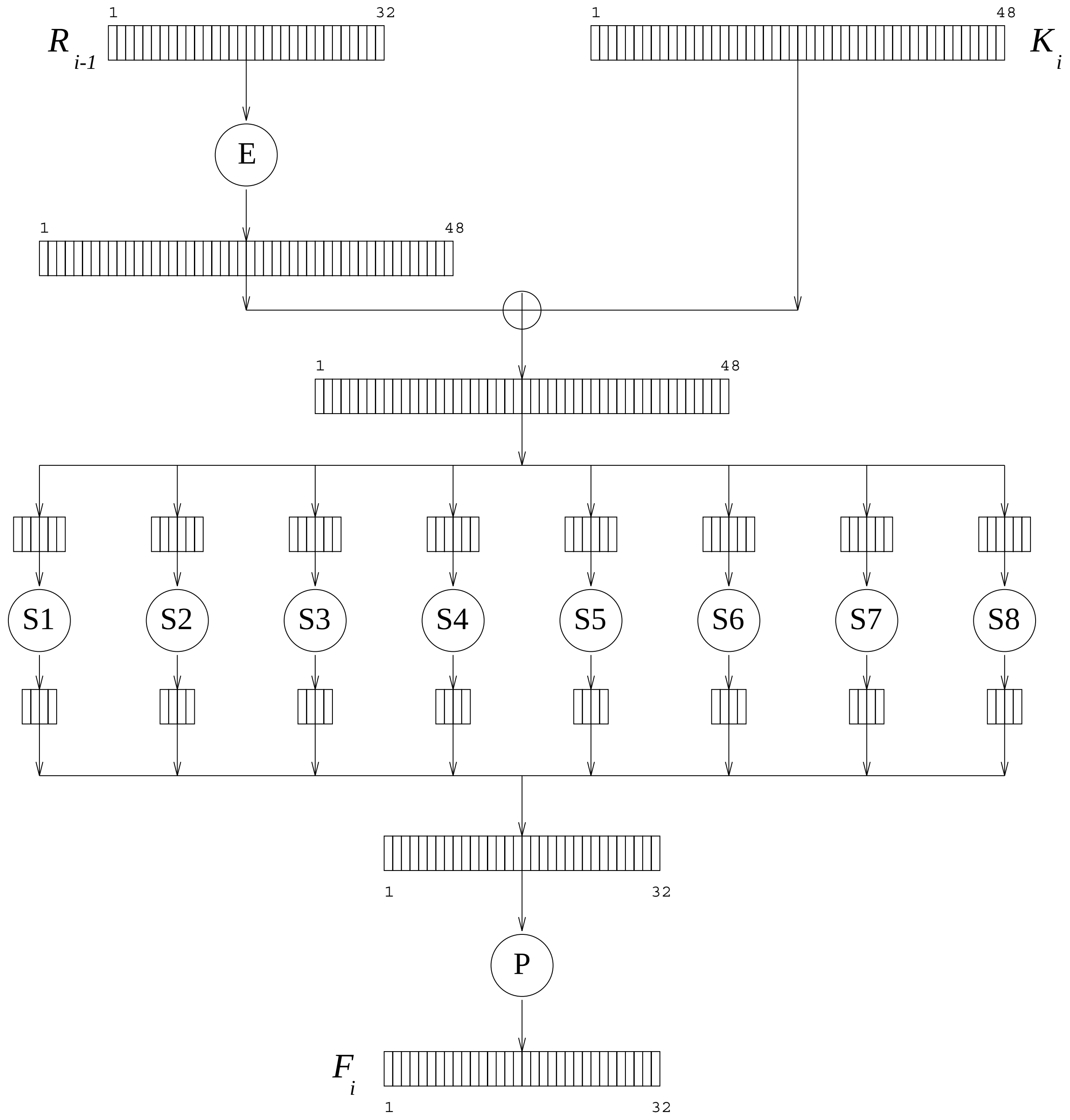
  $DES^{-1}=(IP)^{-1}F_{1}TF_{2}T\ldots F_{15}TF_{16}(IP)$

Decryption algorithm



- The decryptor for DES is identical to the encryptor, except that the 16 keys are used in reverse order.

$f(k_i, R_{i-1})$



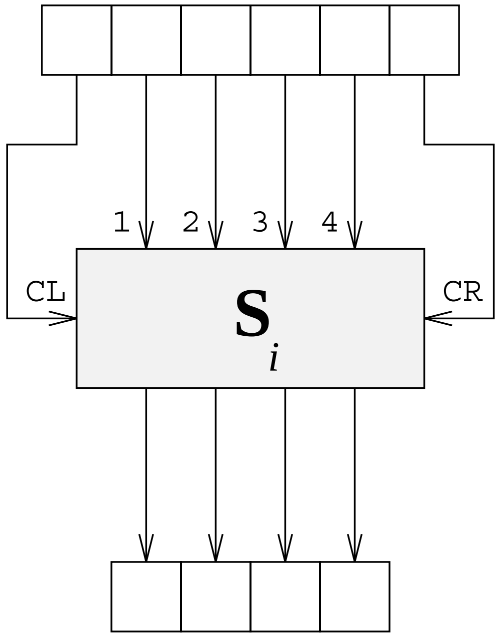
- P is the permutation

16	7	20	21	29	12	28	17
1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9
10	13	30	6	22	11	4	25

- E is the expansion function

32	1	2	3	4	5
8	9	10	11	12	13
16	17	18	19	20	21
24	25	26	27	28	29
4	5	6	7	8	9
12	13	14	15	16	17
20	21	22	23	24	25
28	29	30	31	32	1

Structure of S-Boxes



- CL = left control bit.
- CR= right control bit.
- The CL and CR select one row of the S-Box S_i to use.

- For each of the 4 choices of (CL,CR), S_i performs a different substitution on the 16 possible values of the 4 inner input bits.
- For example $S_1(1,0,1,1,1,0)=[1,0,1,1]$.
- In the S-box below we represent the output values by the integer value of the four binary digits.

00	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
01	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
10	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
11	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

CSCI361
Computer Security

Modern Secret-key cryptography II

Outline

- Properties of S-Boxes.
  - Boolean functions.
- Weaknesses in DES.
  - Complements.
  - Weak keys.
  - Brute force (in parallel).
- Multiple encryption.
  - Meet-in-the-middle.
  - Triple DES.
- Differential cryptanalysis.
- Linear cryptanalysis.

Properties of S-Boxes

- The design principles used for S-Boxes in DES were classified (US) information.
  - This was one of the controversial aspects of DES.
- One of the designers, Coppersmith revealed the three design properties of S-boxes in 1994. These properties ensure good confusion and diffusion for the algorithm. They allayed some concerns regarding DES.

1. For every choice of control bits, every output bit is a **nonlinear** function of input bits.
   - Each row of an S-box consists of 4 non-linear Boolean functions. This non-linearity is the basis of the required **confusion** for DES.

2. Changing a single input bit to an S-box changes at least two output bits of that S-box.
   - This means that the statistical properties of the input bits are spread over the output bits → **diffusion**.
3. When a single input bit is held constant, there is a good balance of 0's and 1's in the $2^5=32$ output half-bytes as the remaining 5 input bits are varied.
   - This ensures a uniform distribution of the algorithm output, and stops cryptanalysts from making statistical inferences.

- 1, 2 and 3 imply good confusion and diffusion.

Boolean functions

- A **Boolean function** is *linear* if it can be written as

$$f(x_1, x_2, \ldots, x_n) = \sum a_i x_i$$

where *ai* and *xi* take only value 0 or 1 and addition is modulo 2.

- The function $f(x_1,x_2,x_3)=x_1+x_2x_3 \pmod 2$ is *non-linear*, since it cannot be written in that form.
- Linear functions are cryptographically weak because the relationship between input and output is very simple.

Number of rounds

- After 8 rounds every bit of output is affected by every bit of input and every bit of key. That is, **the input information is diffused over the whole output block!**
- Moreover, the dependency between input, output and key is very complex.
- However attacks found on DES imply that 8 rounds does not provide enough complexity.
- 16 rounds are used.

Weaknesses in DES

1. **Complement property:**

Let $\bar{u}$ denote complement of u. That is,

$\bar{u}=u+1 \pmod 2$.

$$DES_Z(X) = DES_{\bar{Z}}(\bar{X})$$

So, if we know ciphertext Y for a plaintext X produced with a key Z, then we also know the cryptogram that key $\bar{Z}$ produces for $\bar{X}$.

This effectively halves the number of keys to be tested in an exhaustive key search.

2. Not every key is a good key:

Let *Z* denote the original 64 bit key.

Z is a **weak** or **self-dual** key if the key scheduling algorithm applied to the key Z produces identical sub-keys $Z_1=Z_2=Z_3=\ldots Z_{16}$.

In this case encryption and decryption are the same operation.

$$DES_Z = DES_Z^{-1}$$

- There are four weak keys. (With each 8-bit word having odd-parity).

i. [00000001 00000001 00000001 00000001 00000001 00000001 00000001 00000001]

ii. [11111110 11111110 11111110 11111110 11111110 11111110 11111110 11111110]

iii. [00011111 00011111 00011111 00011111 00001110 00001110 00001110 00001110]

iv. [11100000 11100000 11100000 11100000 11110001 11110001 11110001 11110001]

v. These are the keys which make C_0 all zero or all one, **and** also make D_0 all zero or all one.

(0101010101010101), (FEFEFEFEFEFEFEFE), (IFIFIFIF0E0E0E0E), and (E0E0E0E0FIFIFIFI)

- Z is a **semi-weak** key, or **key with a dual**, if there is another key Z’ such that

$$DES_Z^{-1} = DES_{Z'} \quad \text{or} \quad DES_{Z'}^{-1} = DES_Z$$

There are 12 semi-weak keys (with odd parity for each 8-bit word). 6 pairs.

E001E00lFl0lFl0l	01E001E00lFl0lFl
FElFFElFFEOEFEOE	1FFElFFEOEFEOEFE
E01FE01FF10EF10E	1FE01FEOOEF10EF1
01FE01FE01FE01FE	FE01FE01FE01FE01
011F011F0l0E010E	1F011F0l0E0l0E01
E0FEE0FEFlFEFlFE	FEE0FEE0FEFlFEF1

- Weak and semi-weak keys can have disastrous effects if multiple encryption strategies are used.
- However, they are too few to be of general cryptanalytic use.

3. **Exhaustive key search:** (brute force attack)

DES has $2^{56} \approx 10^{17}$ keys. Given a plaintext block and the corresponding ciphertext block, we can try each key to decrypt the ciphertext and compare the result with the known plaintext.

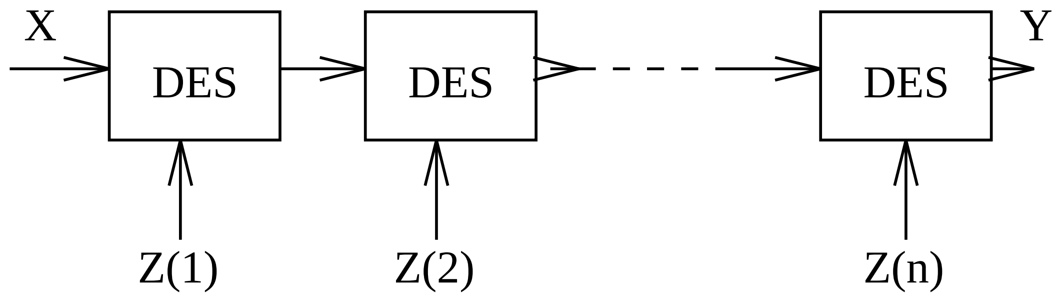
If we tried one key every 10^{-6} seconds, this would take about 10^{11} seconds, or about 3170 years!

- But we can **test keys in parallel**.
- Building a device with 10^7 DES chips would allow exhaustive cryptanalysis with only 10^{10} decryptions per chip.
- Even a few years ago chips existed with more than 10^5 encryptions/decryptions per second, proposals for 10^6 were around as long ago as 1998.
- January 1999 a DES key was broken in 22 hours and 15 minutes.
- Diffie & Hellman (1977) estimated ½ day and $5000 per solution on a special purpose computer costing about $20,000,000.
- They extrapolated their result to conclude that in 1987 the cost of such a machine would be $200,000.

- Michael Wiener (1993) gave a detailed design for a key search machine.
  - The machine was based on a key search chip which is pipelined, so that 16 encryptions take place simultaneously.
  - This chip can test $5*10^7$ keys per second, and could be built using existing technology (Field Programmable Gate Arrays) for $10.50 per chip, at the time.
  - A frame consisting of 5760 chips could be built for $100,000. This would allow a DES key to be found in about 1.5 days on average.
  - A machine using 10 frames would cost $1,000,000, but would reduce the average search time to about 3.5 hours.

Increasing the key length

- There is one method of increase the key size which can be used for DES or any other cipher.
  - Multiple encryption.



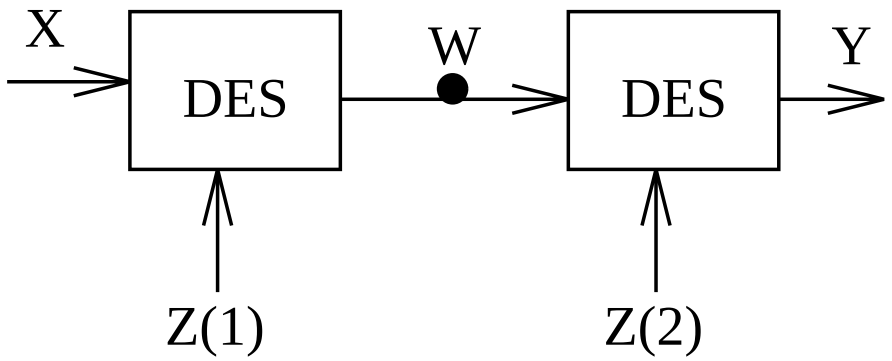
- Multiple encryption does not necessarily produce a stronger cipher. It depends on the algebraic structure of the cipher.
  - There were some concerns that DES might have a group structure, this was proven not to be the case.

- **Example**: Using double encryption with a monoalphabetic cipher does not produce a stronger cipher.

- It is important to consider what the effective key size is for double encryption.
- Naively we might consider that double encryption increases the size of the key by a factor of 2, that is the enemy must find 112 bits of key.
- Unfortunately this is not true, because the enemy can mount a **meet-in-the-middle** attack.

Meet-in-the-middle attack

- We can show that double (DES) encryption produces a cipher with effectively 57 bits of key, not 112.



- The enemy knows a pair of plaintext & ciphertext (X,Y), and a few further cryptograms.

Attack description…

1. The enemy tries all 2^{56} values of Z(2) to decrypt Y. Then he produces a list (w_i, z_i) where

$$w_i = DES_{z_i}^{-1}(Y)$$

This list is sorted on w_i to facilitate later retrievals.

W	Z(2)
w_1	z_1
…	…
w_n	z_n

$n=2^{56}$

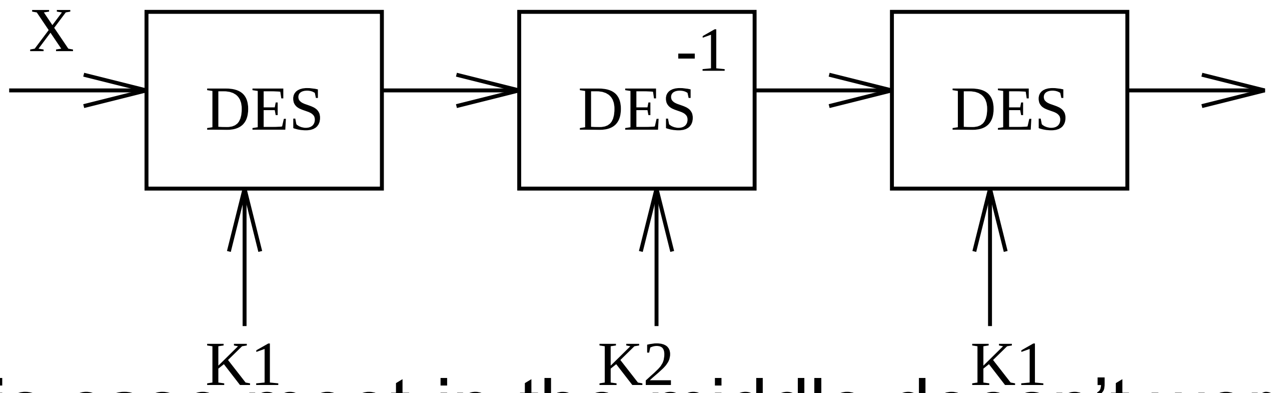
2. The enemy tries all 2^{56} values of Z(1) to encrypt X.
   - After each encryption the enemy checks to see if the result is in the sorted list of w_i.
   - If $DES_{z'}(X)=w_p$, the attacker will use Z(1)=z’ and Z(2)=z_p as the guessed keys.
   - These keys will be checked with other ciphertext, or other plaintext/ciphertext pairs if the attacker has them.
   - On the basis of this checking the guess is either confirmed, (z’,z_p) is the secret key, or the attacker continues their search.

Number of operations

- The number of operations is equal to the number of encryption and decryption operations required.
- All 2^{56} decryptions for Z(2) are used to build the table.
- At most 2^{56} encryptions for Z(1) are required.
- Thus at most $2^{56}+2^{56}=2^{57}$ operations are required.
- So the effective key space is 57 bits.

Triple DES (3DES-EDE2)

- This was proposed by Tuchman (1978).
- There are various different "modes".



- In this case meet-in-the-middle doesn't work.
- Software implementations of this cipher are relatively slow.

Differential cryptanalysis

- Biham & Shamir 1991
  - This is a **chosen plaintext attack** on iterated ciphers.
  - In differential cryptanalysis we analyse the effect of the difference in a plaintext pair on the differences of the resultant ciphertext pair. These differences are used to assign probabilities to possible keys and result in a **good estimate** of possible keys.
  - The essence of the attack is an observation made on a single round.

- Complexity of the attack is measured in terms of the number of encryptions, or the number of plaintext/ciphertext required for the analysis, and depends on the number of rounds.
- For small numbers of rounds the attack is more efficient than exhaustive key search but with increased number of rounds it becomes less efficient. The results to the right were announced in 1991:

Rounds	Complexity
4	2^4
6	2^8
8	2^{16}
10	2^{35}
12	2^{43}
14	2^{51}
15	2^{52}
16	2^{58}

- These results suggest the method does not work against 16 round DES.
  - Was choosing 16 rounds a coincidence?
  - No. DES designers at IBM knew by 1974 about differential cryptanalysis, long before it was used against FEAL by Murphy (1990) or applied against DES by Biham & Shamir (1991).
  - The S-Boxes were chosen to resists differential cryptanalysis.

- Another way of describing the effectiveness of the attack is to describe how much information is needed. A differential cryptanalysis attack against DES requires 2^{47} chosen plaintexts.

- An interesting result of this attack is that it shows **no key scheduling** can make DES more secure against this attack.

Linear cryptanalysis

- Matsui 1993. First applied to another cipher (FEAL, Matsui & Yamagishi, 1992).
- This method finds linear approximations for DS which hold with a probability p≠½.

$$x_{i_1} \oplus x_{i_2} \ldots y_{j_1} \oplus y_{j_2} \ldots = z_{k_1} \oplus z_{k_2} \ldots$$

- These kind of equations can be used to find estimates for key bits: The left hand side of the equation is calculated for many pairs of plaintext/ciphertext. The majority outcome gives an estimate for right hand side of the equation.
- It exploits the fact that knowledge *of pairs of* plaintext blocks, and the corresponding ciphertext blocks, usually allows the round key to be determined.
- This requires approximately 2^{43} known plaintexts.