Cryptography (and Information Security)
6CCS3CIS / 7CCSMCIS

Prof. Luca Viganò

Department of Informatics
King's College London, UK

First term 2019/20

Lecture 2

Motivation then and now

Three can keep a secret, if two of them are dead.

— Benjamin Franklin

We interact and transact by directing flocks of digital packets towards each other through cyberspace, carrying love notes, digital cash, and secret corporate documents. Our personal and economic lives rely on our ability to let such ethereal carrier pigeons mediate at a distance what we used to do with face-to-face meetings, paper documents, and a firm handshake. How do we converse privately when every syllable is bounced off a satellite and smeared over an entire continent? How should a bank know that it really is Bill Gates requesting from his laptop in Fiji a transfer of $10,000,000,000 to another bank? Fortunately, the mathematics of cryptography can help.

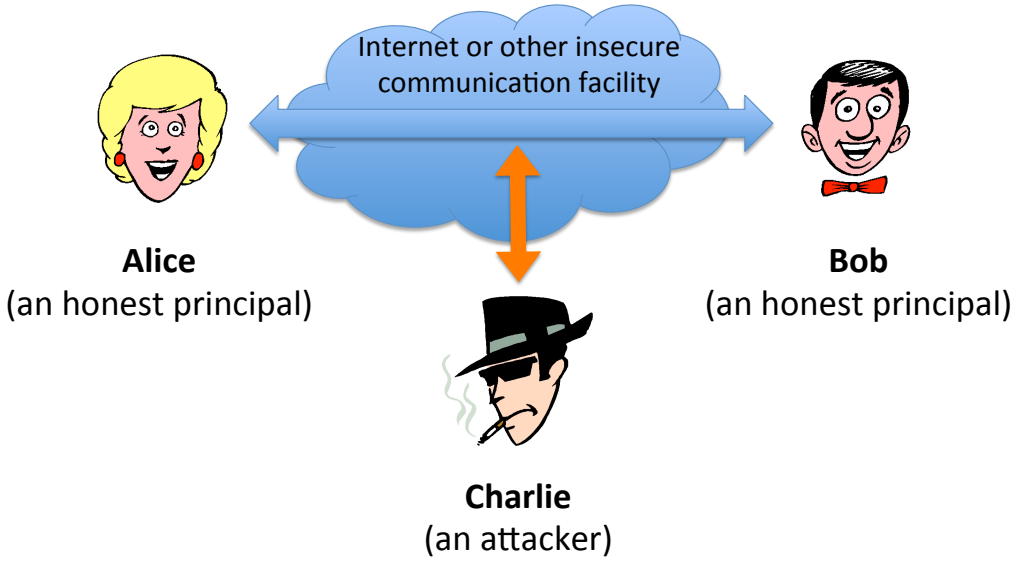
— Ron Rivest

Outline

Table of contents I

Table of contents II

What's it all about?

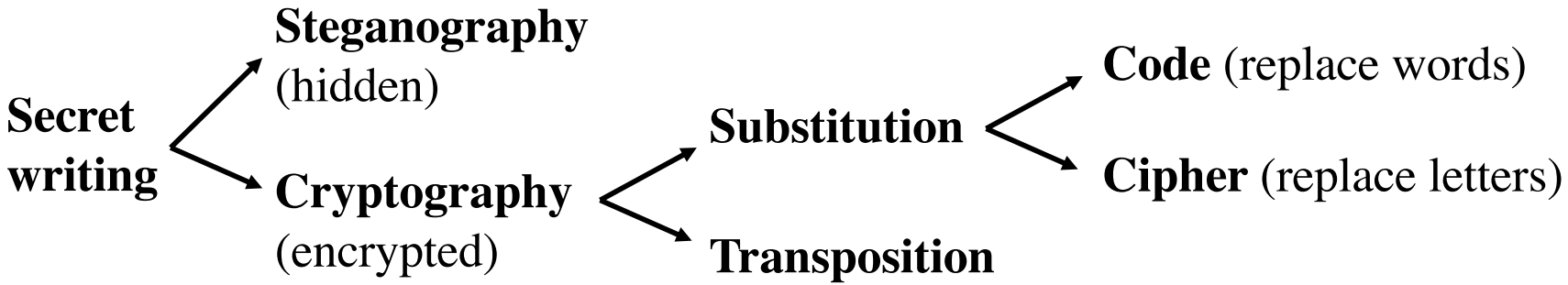


How do we turn an insecure communication facility (like the Internet) into a secure one?

Where security means that one of more security properties (e.g., confidentiality, integrity, authentication, non-repudiation, anonymity, unobservability, timeliness, availability, etc.) are guaranteed.

Cryptography is the enabling technology.

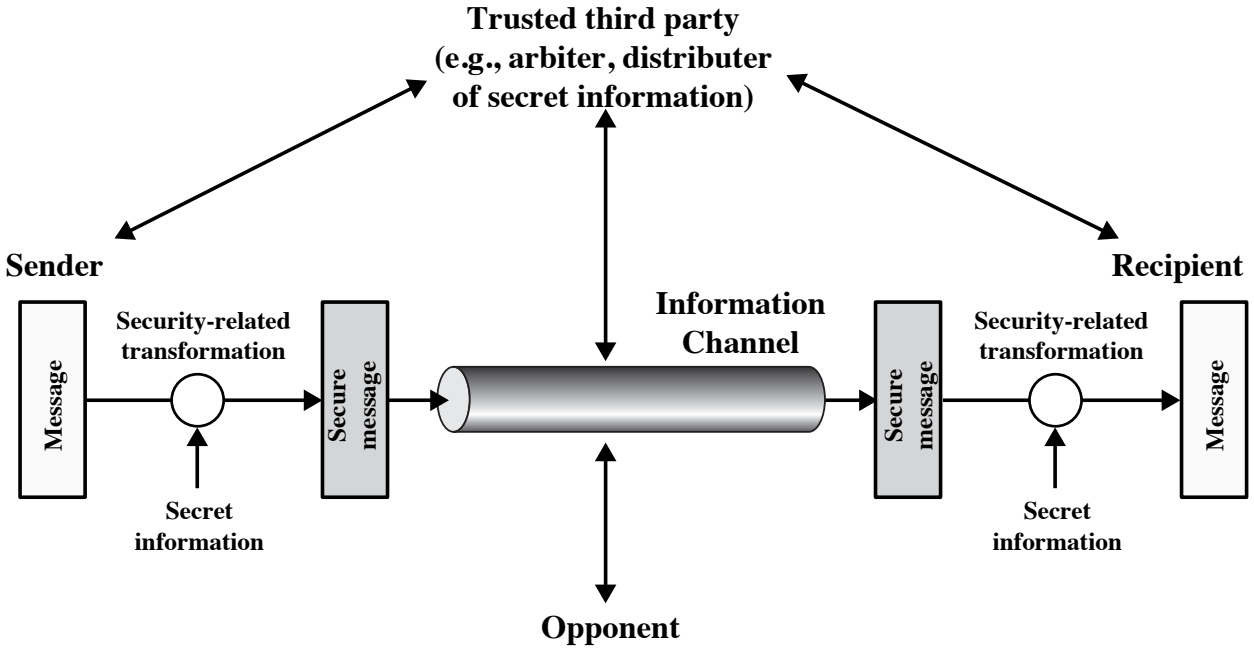
What is cryptography?



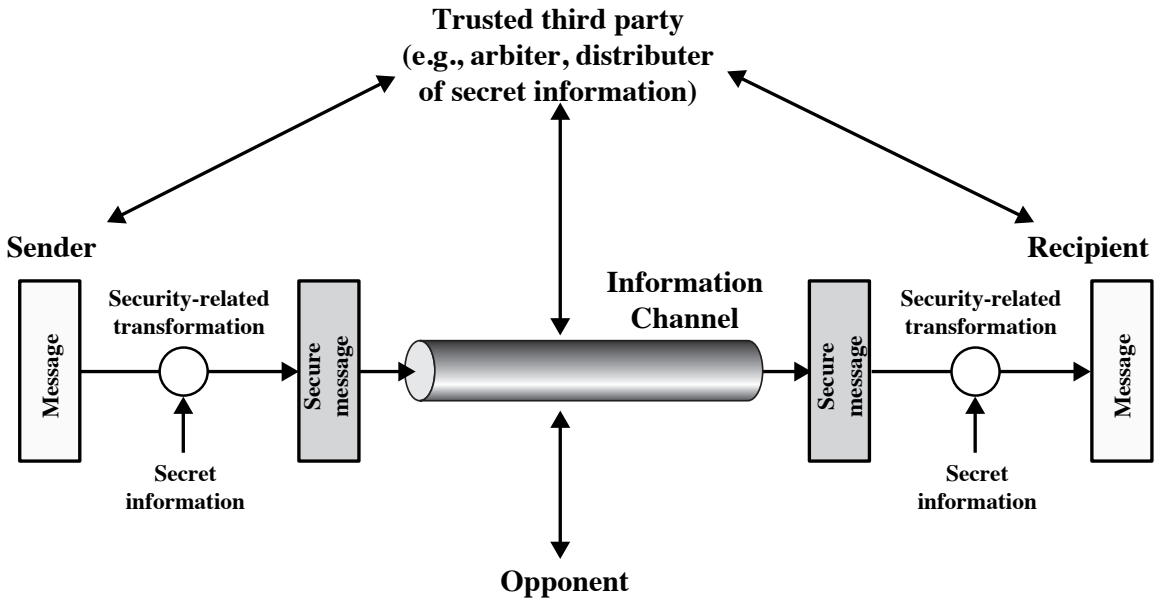
- Cryptology: the study of secret writing.
- Steganography: the science of hiding messages in other messages.
- Cryptography: the science of secret writing.
  Note: terms like encrypt, encode, and encipher are often (loosely and wrongly) used interchangeably.
- Cryptanalysis: science of recovering the plaintext from ciphertext without the key.

We will discuss each of these (in some detail).

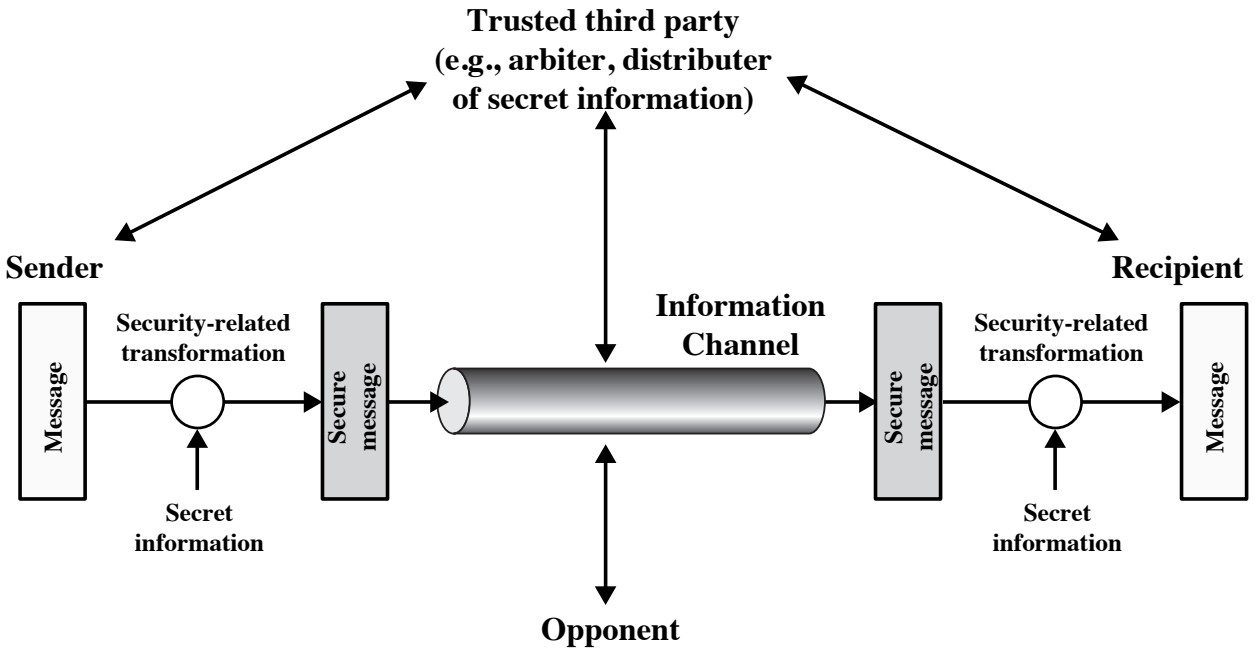
A general model for (network) security



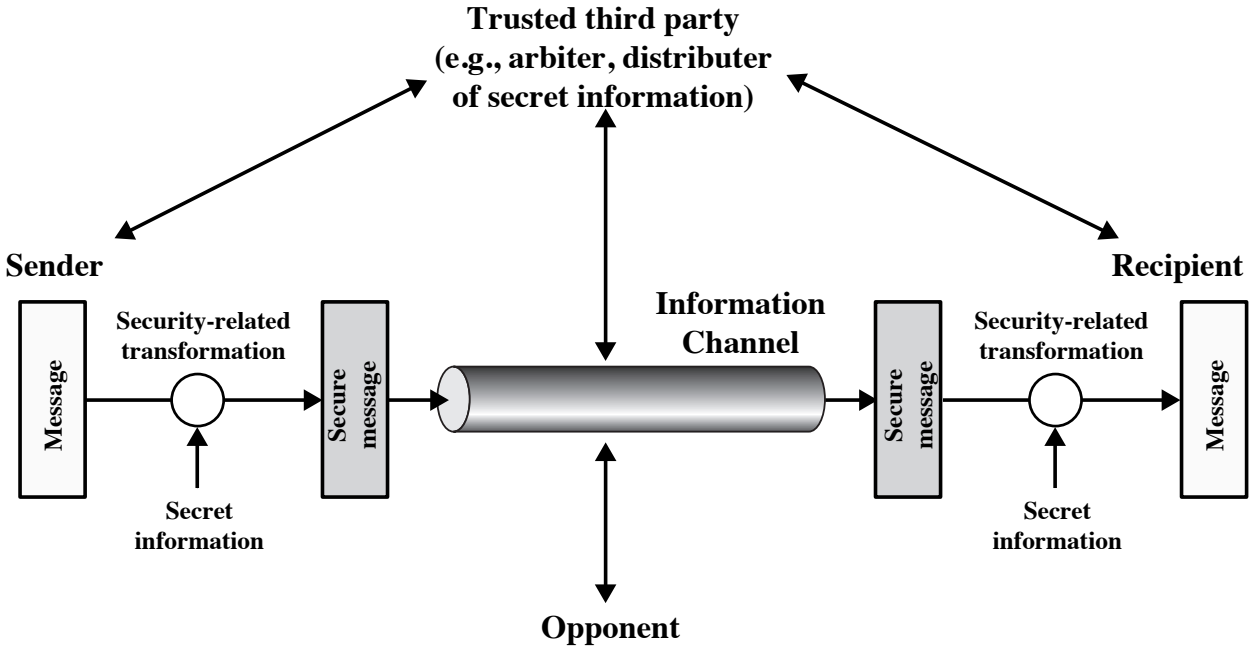
- A **message** is to be transferred **from one principal to another** across some sort of Internet service.
- The two principals must cooperate for the exchange to take place.
- A logical **information channel** is established by defining a route through Internet from source to destination and by principals' cooperative use of communication protocols (e.g., TCP/IP).



- All the techniques for providing security have two components:
  1. A **security-related transformation** on information to be sent, e.g.
     - **encryption** of the message, which scrambles the message so that it is unreadable by the opponent, and/or
     - addition of a **code** based on the contents of the message, which can be used to verify the identity of the sender (e.g., MAC or MDC).
  2. Some **secret information** shared by the two principals and, it is hoped, unknown to the opponent, e.g.
     - **encryption key** used in conjunction with transformation to scramble message before transmission and unscramble it on reception.

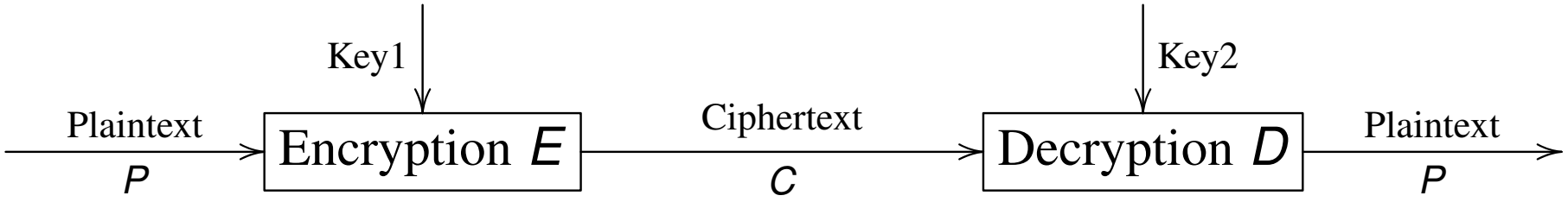


- A **trusted third party** may be needed to achieve secure transmission, e.g.
  - responsible for distributing the secret information to the two principals while keeping it from any opponent, or
  - needed to arbitrate disputes between the two principals concerning the authenticity of a message transmission.



- This general model shows that there are **4 basic tasks**:
  1. Design an algorithm for performing the security-related transformation. The algorithm should be such that an opponent cannot defeat its purpose.
  2. Generate the secret information to be used with the algorithm.
  3. Develop methods for the distribution and sharing of the secret information.
  4. Specify a protocol to be used by the two principals that makes use of the security algorithm and the secret information to achieve a particular security service.

General cryptographic schema



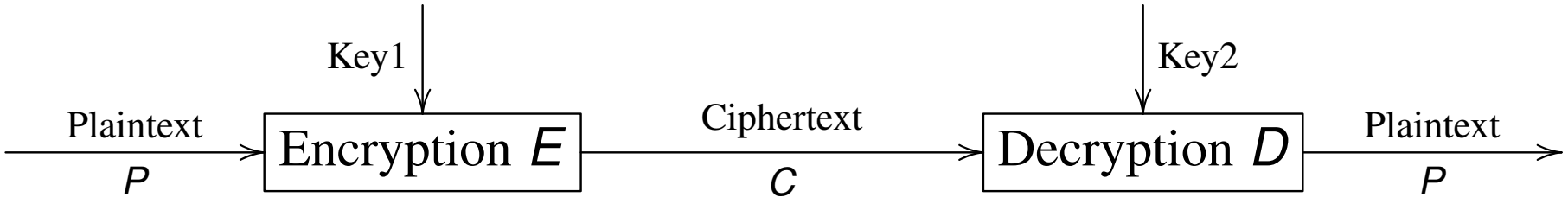
where $E(\text{Key1}, P) = C$ and $D(\text{Key2}, C) = P$.

Terminology

- **Plaintext** (**plain text**, **clear text**, ...): text that can be read and "understood" (e.g., by a human being).
- **Encryption**: transformation (function, process, procedure, ...) E that takes in input a plaintext and a key and generates a ciphertext.
- **Ciphertext** (**cipher text**, **encrypted text**, ...): transformed (scrambled, ...) text that needs to be "processed" to be "understood" (e.g., by a human being).
- **Decryption**: transformation (function, process, procedure, ...) D that takes in input a ciphertext and a key and generates a plaintext.

Cipher: a function (algorithm, ...) for performing encryption/decryption.

General cryptographic schema



where $E(\text{Key1}, P) = C$ and $D(\text{Key2}, C) = P$.

- **Symmetric algorithms**:
  - Key1 = Key2, or are easily derived from each other.
- **Asymmetric (or public key) algorithms**:
  - Different keys, which cannot be derived from each other.
  - Public key can be published without compromising private key.
- Encryption and decryption should be easy, if keys are known.
- **Security depends only on secrecy of the key, not on the algorithm.**

Kerckhoffs' "La Cryptographie Militaire"

Security depends only on secrecy of the key, not on the algorithm.

J.-G.-H.-V.-F.-A.-A. Kerckhoffs von Nieuwenhof, "La Cryptographie Militaire" in Journal des sciences militaires, vol. IX, 1883 (!)

Six fundamental principles for military ciphers:

1. Le système doit être matériellement, sinon mathématiquement, indéchiffrable.
2. Il faut qu'il n'exige pas le secret, et qu'il puisse sans inconvénient tomber entre les mains de l'ennemi.
3. La clef doit pouvoir en être communiquée et retenue sans le secours de notes écrites, et être changée ou modifiée au gré des correspondants.
4. Il faut qu'il soit applicable à la correspondance télégraphique.
5. Il faut qu'il soit portatif, et que son maniement ou son fonctionnement n'exige pas le concours de plusieurs personnes.
6. Enfin, il est nécessaire, vu les circonstances qui en commandent l'application, que le système soit d'un usage facile, ne demandant ni tension d'esprit, ni la connaissance d'une longue série de règles à observer.

Kerckhoffs' "La Cryptographie Militaire"

Security depends only on secrecy of the key, not on the algorithm.

J.-G.-H.-V.-F.-A.-A. Kerckhoffs von Nieuwenhof, "La Cryptographie Militaire" in Journal des sciences militaires, vol. IX, 1883 (!)

Six fundamental principles for military ciphers:

1. The system must be substantially, if not mathematically, undecipherable.
2. **The system must not be required to be secret and can be stolen by the enemy without causing trouble.**
3. It must be easy to communicate and retain the key without the aid of written notes, it must also be easy to change or modify the key at the discretion of the correspondents.
4. The system ought to be compatible with telegraph communication.
5. It must be portable, and its use must not require more than one person.
6. Finally, given the circumstances in which such system is applied, it must be easy to use and must neither stress the mind or require the knowledge of a long series of rules.

A simple example

- Map each letter to a number:

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

- Define:

$$C = E(K, P) = (P + K) \bmod 26$$
$$P = D(K, C) = (C - K) \bmod 26$$

- Pick, say, $K = 3$ so that $C = E(3, P) = (P + 3) \bmod 26$
- If $P =$ "H"
  then $C = E(3, 7) = (7 + 3) \bmod 26 = 10 \bmod 26 = 10 =$ "K"
- If $P =$ "Y" then $C = (24 + 3) \bmod 26 = 27 \bmod 26 = 1 =$ "B"
- Hence, if full plaintext is "HEYYOU", then ciphertext is "KHBBRX"
- How difficult is to encrypt?
- And to decrypt? What if ciphertext is "KHOOR ZRUOG"? (space is extra information)
- And what if ciphertext is "L WRSL QRQ DYHYDQR QLSRWL"?

A simple example... but still in use

- Might look very simple, but still used in some form today.
- For instance, Bernardo Provenzano (1933–2016), accused of being the “capo di tutti capi” of the Sicilian mafia, was captured after eluding the police for over forty years.
- Provenzano was caught after police intercepted messages between Provenzano and other members of his organization written in a cipher that they were able to break:

A	B	C	D	E	F	G	H	I	L	M	N	O	P	Q	R	S	T	U	V	Z
4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24

Table of contents I

Table of contents II

A mathematical formalization: En-/decryption

- $\mathcal{A}$, the alphabet, is a finite set.
- $\mathcal{M} \subseteq \mathcal{A}^*$ is the message space. $M \in \mathcal{M}$ is a plaintext (message).
- $\mathcal{C}$ is the ciphertext space, whose alphabet may differ from $\mathcal{M}$.
- $\mathcal{K}$ denotes the key space of keys.
- Each $e \in \mathcal{K}$ determines a bijective function from $\mathcal{M}$ to $\mathcal{C}$, denoted by E_e. E_e is the encryption function (or transformation).

  Note: we will write $E_e(P) = C$ or, equivalently, $E(e, P) = C$.
- For each $d \in \mathcal{K}$, D_d denotes a bijection from $\mathcal{C}$ to $\mathcal{M}$.
  D_d is the decryption function.
- Applying E_e (or D_d) is called encryption (or decryption).

A mathematical formalization: En-/decryption (cont.)

- An encryption scheme (or cipher) consists of a set $\{E_e \mid e \in \mathcal{K}\}$ and a corresponding set $\{D_d \mid d \in \mathcal{K}\}$ with the property that for each $e \in \mathcal{K}$ there is a unique $d \in \mathcal{K}$ such that $D_d = E_e^{-1}$; i.e.,

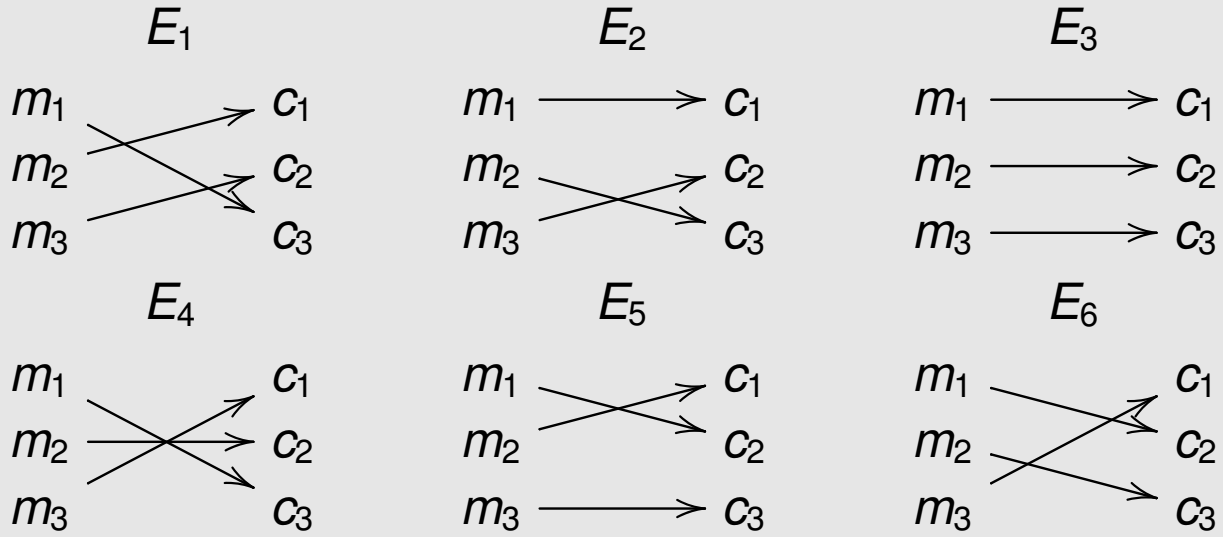
$$D_d(E_e(m)) = m \qquad \text{for all } m \in \mathcal{M}\,.$$

- The keys e and d above form a key pair, sometimes denoted by (e, d). They can be identical (i.e., the symmetric key).

- To construct an encryption scheme requires fixing a message space $\mathcal{M}$, a ciphertext space $\mathcal{C}$, and a key space $\mathcal{K}$, as well as encryption transformations $\{E_e \mid e \in \mathcal{K}\}$ and corresponding decryption transformations $\{D_d \mid d \in \mathcal{K}\}$.

An example

Let $\mathcal{M} = \{m_1, m_2, m_3\}$ and $\mathcal{C} = \{c_1, c_2, c_3\}$.
There are 3! = 6 bijections from $\mathcal{M}$ to $\mathcal{C}$.
The key space $\mathcal{K} = \{1, 2, 3, 4, 5, 6\}$ specifies these transformations.



Suppose Alice and Bob agree on the transformation E_1.
To encrypt m_1, Alice computes $E_1(m_1) = c_3$.
Bob decrypts c_3 by reversing the arrows on the diagram for E_1 and observing that c_3 points to m_1.

Table of contents I

Table of contents II

Three characteristics of cryptographic systems

Cryptographic systems characterized along 3 independent dimensions:

1. Type of operations used to transform plaintext into ciphertext.

- All encryption algorithms are based on two general principles:
  - **Substitution**: each element in plaintext (bit, letter, group of bits or letters) is mapped into another element.
  - **Transposition**: elements in plaintext are rearranged.

  All **operations must be reversible** (so that no information is lost).
- Most systems, referred to as **product systems**, involve multiple stages of substitutions and transpositions.

2. Number of keys used.

- **Symmetric, single-key, secret-key, or conventional encryption**: both sender and receiver use "same" key.
- **Asymmetric, two-key, or public-key encryption**: sender and receiver use different keys.

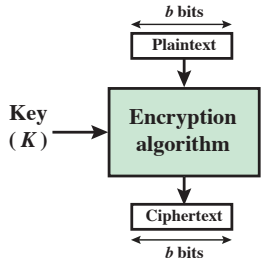
Three characteristics of cryptographic systems

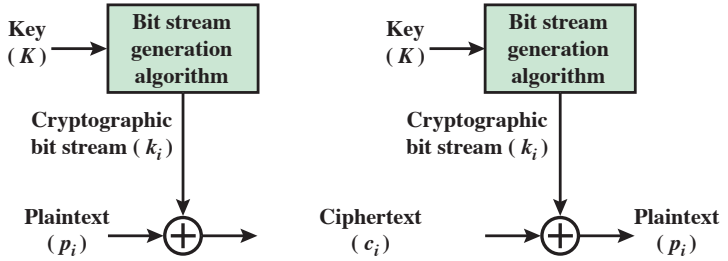
3. Way in which plaintext is processed.

- **Block cipher** processes input one block of elements at a time, producing an output block for each input block.
- **Stream cipher** processes input elements continuously, producing in output one element at a time, as it goes along.

In other words:

- A block cipher is an encryption scheme that breaks up the plaintext message into strings (blocks) of a fixed length t and encrypts one block at a time.
- A stream cipher is one where the block-length is 1.





- In contrast, codes work on words of varying length.

Codes

- Translation given by a code-book.

Word	Code
...	...
The	**1701**
secret	**5603**
mischiefs	**4008**
that	**3790**
I	**2879**
set	**0524**
...	...

```
2327 6605 1702 9853 0001 0970 3190 8817 1320 0000 = I do the wrong, and first begin to brawl.
1701 5603 4008 3790 2879 0524 7946               = The secret mischiefs that I set abroach
2879 2870 6699 1702 3982 5550 8102 7354 0000     = I lay unto the grievous charge of others.
```

(Richard III, Act I, Scene 3)

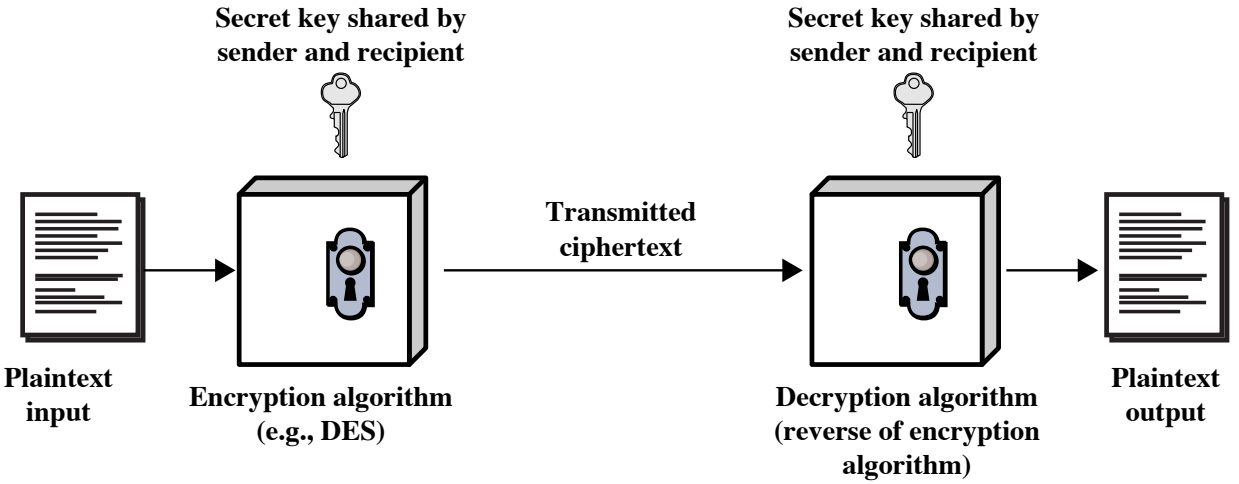
- In general: a string of symbols stands for a complete message.
  - Example: “OCELOT" is ciphertext for “TURN LEFT 90 DEGREES" and “LOLLIPOP" is ciphertext for “TURN RIGHT 90 DEGREES".
- **Problems**:
  - if there’s no entry for “FIREWALL", then you can’t say it!
  - Security is “pushed” to the code-book, which needs to be protected.

Table of contents I

Table of contents II

Symmetric key encryption (symmetric cipher model)

- An encryption scheme $\{E_e \mid e \in \mathcal{K}\}$ and $\{D_d \mid d \in \mathcal{K}\}$ is symmetric-key if for each associated pair (e, d) it is computationally "easy" to determine d knowing only e and to determine e from d. In practice $e = d$.
  - Also known as: secret-key, single-key, one-key, shared-key, conventional encryption.
  - Sender and recipient share a common key.
  - All classical encryption algorithms are symmetric-key (it was only type of encryption prior to invention of public-key crypto in 1970's).
  - Most widely used.



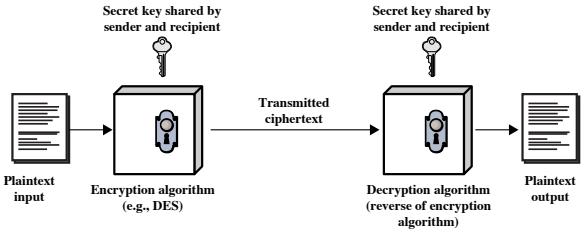
2 requirements for secure use of symmetric encryption

1. A strong encryption algorithm.

- At a minimum: attacker who knows algorithm and has access to one or more ciphertexts should be unable to decipher ciphertext or figure out key.
- Stronger: attacker should be unable to decrypt ciphertext or discover key even if he/she is in possession of a number of ciphertexts together with plaintext that produced each ciphertext.

2. Sender and receiver must obtain copies of secret key in a secure fashion (e.g., a secure channel) and must keep key secure.

- If someone can discover the key and knows the algorithm, all communication using this key is readable.

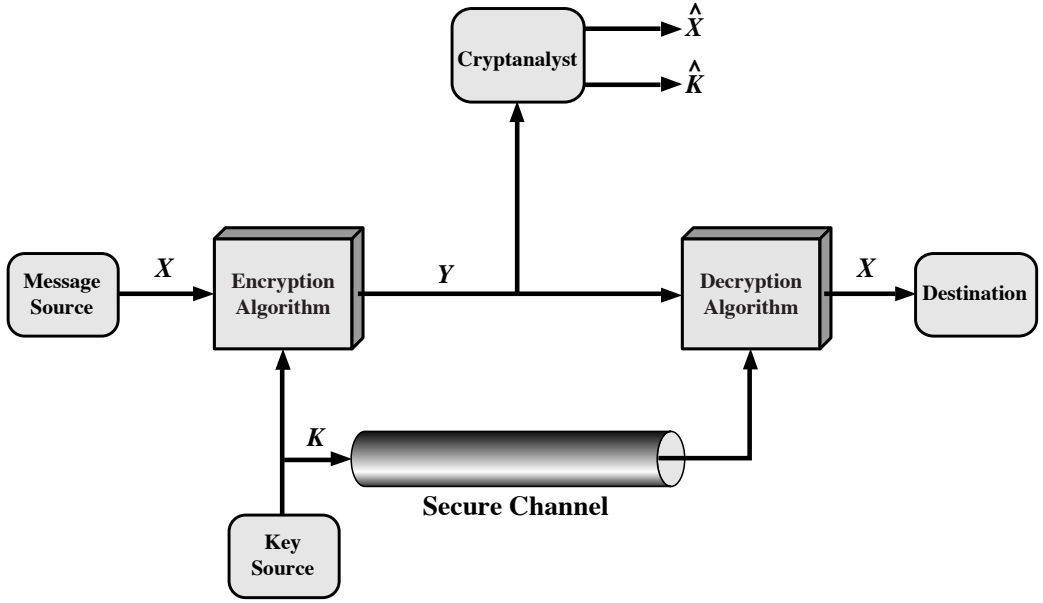


Keep only the key secret

We do not need to keep the algorithm secret;
we need to keep only the key secret.

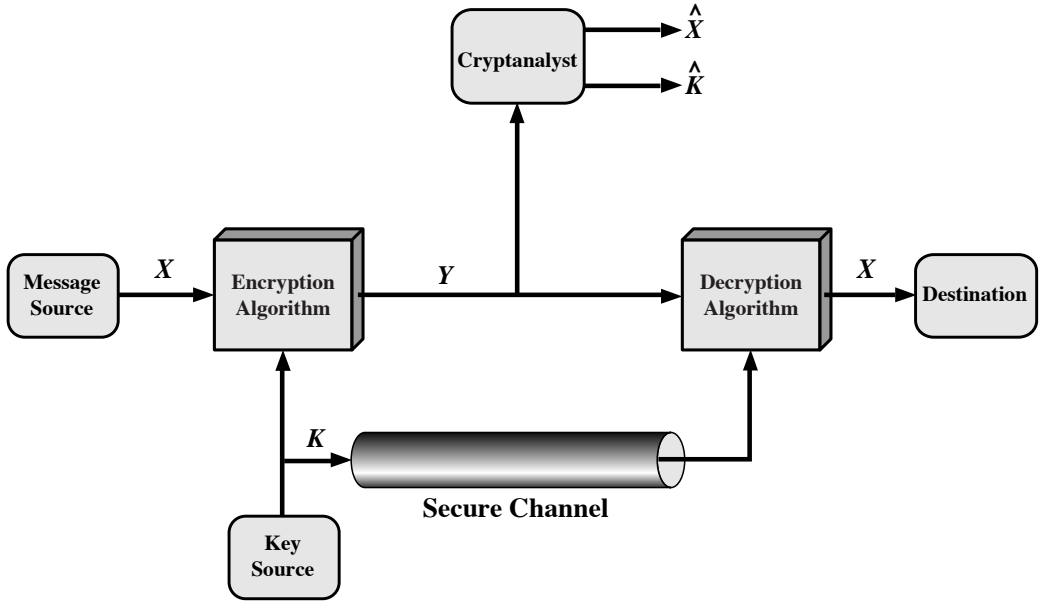
- We assume that is impractical to decrypt a message on basis of ciphertext *plus* knowledge of encryption/decryption algorithm.
- This makes symmetric encryption feasible for widespread use:
  - Manufacturers can and have developed low-cost chip implementations of data encryption algorithms.
  - Chips widely available and incorporated into a number of products.

Detailed model of symmetric cryptosystem



- A source produces a message in plaintext: $X = [X_1, X_2, \ldots, X_i]$. The i elements of X are letters in some finite alphabet.
  - Traditionally: alphabet consisted of the 26 capital letters.
  - Nowadays: binary alphabet $\{0, 1\}$ typically used.
- An encryption key of the form $K = [K_1, K_2, \ldots, K_j]$ 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.

Detailed model of symmetric cryptosystem



- Encryption algo forms ciphertext $Y = E(K, X) = [Y_1, Y_2, \ldots, Y_n]$.
- The intended receiver, in possession of the key K, is able to invert the transformation: $X = D(K, Y)$.
- Attacker
  - knows the encryption (E) and decryption (D) algorithms,
  - observing Y but not having access to K or X, may attempt to recover X or K or both X and K, by generating $\hat{X}$ and/or $\hat{K}$.

Table of contents I

Table of contents II

Cryptanalysis and brute-force attacks

- Typical objective of attacking an encryption system
  - is not simply to recover the plaintext of a single ciphertext
  - but to recover the key in use (so that all future and past messages encrypted with that key are compromised).
- 2 general attack approaches:

Cryptanalysis

- Attacks rely on nature of the algorithm plus perhaps some knowledge of general characteristics of the plaintext or even some sample plaintext-ciphertext pairs.
- Exploits the characteristics of the algorithm to attempt to deduce a specific plaintext or to deduce the key being used.

Brute-force attack

- 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.

Brute-force attack

- It is always possible: simply try every key until an intelligible translation of the ciphertext into plaintext is obtained.
  It thus assumes that plaintext is known or recognizable.
- Its cost (heavily) depends on key size and on average, half of all possible keys must be tried to achieve success.
- Average time required for exhaustive key search:

Key Size (bits)	Number of Alternative Keys	Time Required at 1 Decryption/μs	Time Required at 10^6 Decryptions/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	$2^{128} = 3.4 \times 10^{38}$	$2^{127}\ \mu$s = 5.4×10^{24} years	5.4×10^{18} years
168	$2^{168} = 3.7 \times 10^{50}$	$2^{167}\ \mu$s = 5.9×10^{36} years	5.9×10^{30} years
26 characters (permutation)	$26! = 4 \times 10^{26}$	$2 \times 10^{26}\ \mu$s = 6.4×10^{12} years	6.4×10^6 years

  - Key size: 56 used for DES, 168 for triple DES, 128 (minimum size) for AES.
  - Also: substitution codes that use a 26-character key (discussed later), in which all possible permutations of the 26 characters serve as keys.
  - 1 decryption/μs is perfectly reasonable.
  - 10^6 decryption/μs (in the future?): DES no longer computationally secure!

Some comparisons for key size

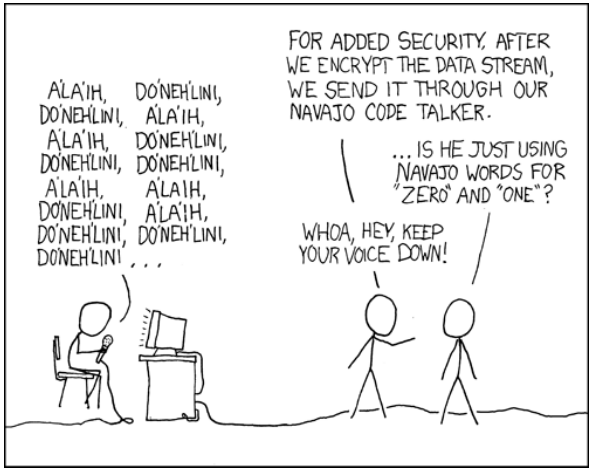
Key Size (bits)	Number of Alternative Keys	Time Required at 1 Decryption/μs	Time Required at 10^6 Decryptions/μs
32	$2^{32} = 4.3 \times 10^9$	2^{31} μs = 35.8 minutes	2.15 milliseconds
56	$2^{56} = 7.2 \times 10^{16}$	2^{55} μs = 1142 years	10.01 hours
128	$2^{128} = 3.4 \times 10^{38}$	2^{127} μs = 5.4×10^{24} years	5.4×10^{18} years
168	$2^{168} = 3.7 \times 10^{50}$	2^{167} μs = 5.9×10^{36} years	5.9×10^{30} years
26 characters (permutation)	$26! = 4 \times 10^{26}$	2×10^{26} μs = 6.4×10^{12} years	6.4×10^6 years

- 2^{92} atoms in the average human body
- 2^{128} possible keys for a 128-bit key (= total number of IP addresses available under IPv6)
- 2^{170} atoms in the planet
- 2^{233} atoms in the galaxy
- 2^{256} possible keys for a 256-bit key ($\approx 10^{77}$)
  10^{78} to 10^{82} atoms in the universe
- 2^{333} smallest power of 2 that is greater than googol (10^{100})

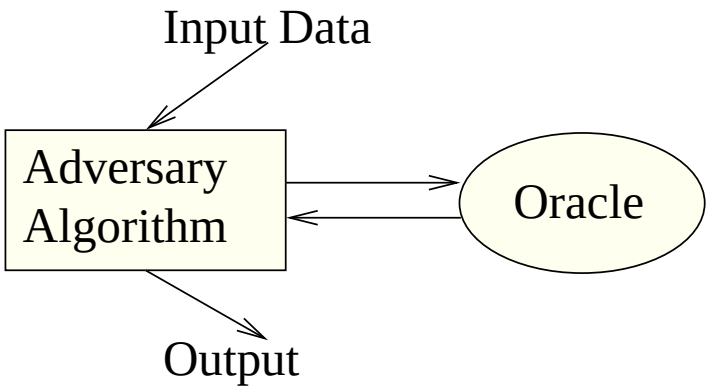
Cryptanalitic attacks

- Always assume attackers know the algorithms used!
  - Worst-case analysis and realistic in open systems.
  - Algorithms should be published to facilitate the evaluation of their security.
- Contrast with security by obscurity.

  Analogy: hide a letter under your mattress versus lock it in a safe, whose design has been published and whose locking mechanism has withstood attacks from the world's best safecrackers.
- But security by obscurity has proven extremely dangerous!



Model of Attack



We can think of the adversary as playing a game:

Input: Whatever adversary necessarily knows from the beginning, e.g., public key, distribution of plain texts, etc.

Oracle: Models information adversary can obtain during an attack. Different kinds of information characterize different types of attacks.

Output: Whatever the adversary wants to compute, e.g., secret key, partial information on plain text, etc. He wins if he succeeds.

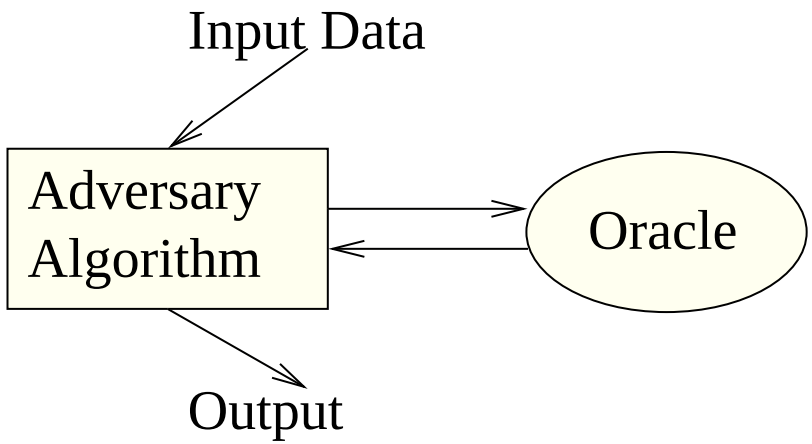
Types of attack

Type of Attack	Known to Cryptanalyst
Ciphertext Only	• Encryption algorithm • Ciphertext
Known Plaintext	• Encryption algorithm • Ciphertext • One or more plaintext-ciphertext pairs formed with the secret key
Chosen Plaintext	• Encryption algorithm • Ciphertext • Plaintext message chosen by cryptanalyst, together with its corresponding ciphertext generated with the secret key
Chosen Ciphertext	• Encryption algorithm • Ciphertext • Ciphertext chosen by cryptanalyst, together with its corresponding decrypted plaintext generated with the secret key
Chosen Text	• Encryption algorithm • Ciphertext • Plaintext message chosen by cryptanalyst, together with its corresponding ciphertext generated with the secret key • Ciphertext chosen by cryptanalyst, together with its corresponding decrypted plaintext generated with the secret key

Types of attack

- **Ciphertext only:**
  - **Given:** $C_1 = E_K(M_1), \ldots, C_n = E_K(M_n)$
  - **Deduce:** $M_1, \ldots, M_n$ or algorithm to compute M_{n+1} from $C_{n+1} = E_K(M_{n+1})$
- **Known plaintext:**
  - **Given:** $M_1, C_1 = E_K(M_1), \ldots, M_n, C_n = E_K(M_n)$
  - **Deduce:** Inverse key or algorithm to compute M_{n+1} from $C_{n+1} = E_K(M_{n+1})$
- **Chosen plaintext:** Same as above but cryptanalyst may choose $M_1, \ldots, M_n$.
- **Adaptive chosen plaintext:** Cryptanalyst can not only choose plaintext, but he can modify the plaintext based on encryption results.
- **Chosen ciphertext:** Cryptanalyst can chose different ciphertexts to be decrypted and gets access to the decrypted plaintext.

How to build a definition of security



1. Specify an oracle (a type of attack).
2. Define what the adversary needs to do to win the game, i.e., a condition on his output.
3. The system is secure under the definition, if any efficient adversary wins the game with only negligible probability.

A standard definition (conventional encryption)

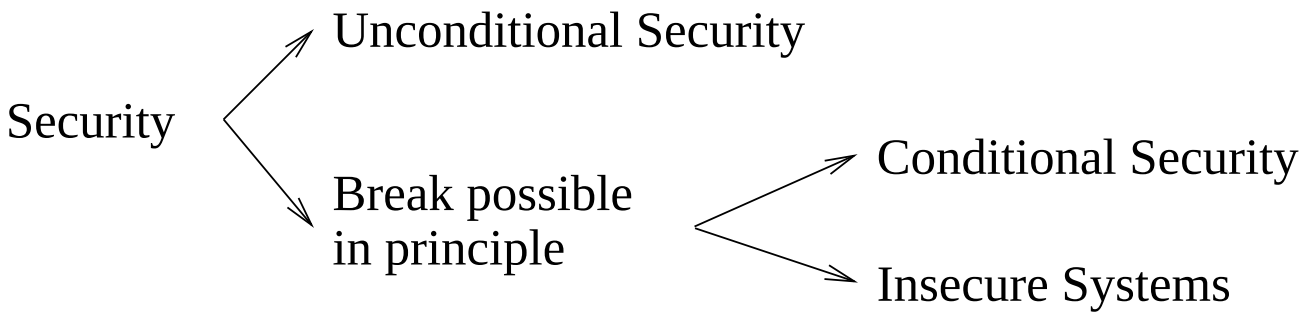
- No input data for adversary.
- Choose plaintext attack of following kind:

  Case 0: when asked to encrypt message m, oracle returns encryption of m under a fixed key that is randomly chosen initially; or

  Case 1: oracle returns encryption of a randomly chosen message, totally independent of m.

Idea: In case 1, adversary gets completely useless data. If he cannot tell this apart from correct encryptions, he cannot do any damage in the real world (case 0) either.

Classification of security



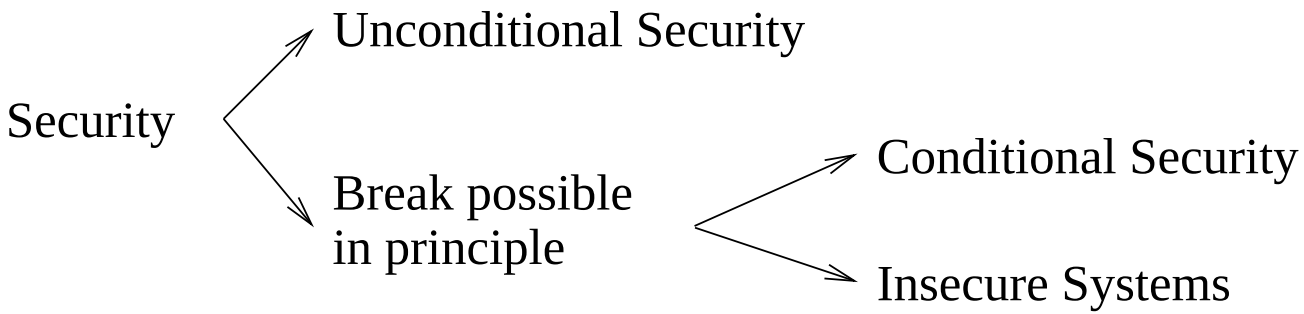
Unconditional Security

System (algorithm) is secure even if attacker has unbounded computing power since the ciphertext provides insufficient information to uniquely determine the corresponding plaintext.

- Security measured using information theory.
- With exception of one-time pad, there's no unconditionally secure encryption algorithm.
- Hence, strive for algorithm that meets one or both of:
  - Cost of breaking cipher exceeds value of encrypted information.
  - Time required to break cipher exceeds useful lifetime of information.

Algo is **computationally secure** if either of these two criteria met.

Classification of security



Conditional Security

System can be broken in principle, but this requires more computing power than a realistic attacker would have.

- Security measured using complexity theory.

Table of contents I

Table of contents II

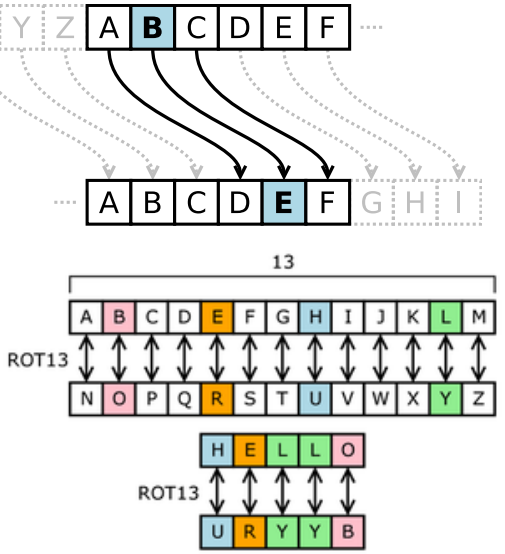
Table of contents III

Substitution ciphers

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

If the plaintext is viewed as a sequence of bits, then substitution involves replacing plaintext bit patterns with ciphertext bit patterns.

- Some simple substitution ciphers:
  - **KHOOR ZRUOG** = **HELLO WORLD**
    Caesar cipher: each plaintext character is replaced by character 3 to the right modulo 26.
  - **Jnf vg n pne be n png V fnj ?** = **Was it a car or a cat I saw ?**
    ROT13: shift each letter by 13 places.
    Under Unix-like systems:
    tr a-zA-Z n-za-mN-ZA-M
  - **1-24-4 1-24-4** = **BYE BYE**
    Alphanumeric: substitute numbers for letters.



How hard are these to cryptanalyze? Caesar? General?

Table of contents I

Table of contents II

Table of contents III

Caesar cipher

- "Earliest" known, simplest, substitution cipher (used by Julius Caesar), e.g.

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

- Replace each letter of the alphabet with the letter standing 3 places further down the alphabet (wrapping around):

```
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
```

- Mathematically, give each letter a number:

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: $C = E(3, P) = (P + 3) \mod 26$
- In general, for $K \in \{1, \ldots, 25\}$:

$$C = E(K, P) = (P + K) \mod 26$$
$$P = D(K, C) = (C - K) \mod 26$$

Brute-force attack

If it is known that a given ciphertext is a Caesar cipher, then brute-force cryptanalysis is easy: simply try all 25 possible 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

Why is brute-force cryptanalysis possible?

- 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.
- What generally makes brute-force cryptanalysis impractical is the use of an algorithm that employs a large number of keys, e.g.
  - triple DES algorithm makes use of a 168-bit key, giving a key space of 2^{168} ($> 3.7 \times 10^{50}$) possible keys.
- If language of plaintext is unknown, then output may not be recognizable, e.g.
  - L WRSL QRQ DYHYDQR QLSRWL = i topi non avevano nipoti
    (the mice had no grandsons)

Substitution ciphers: a little bit of history (& geography)

- **Kama Sutra cipher**:
  - Kama Sutra: a text written in the 4th century AD by the Brahmin scholar Vatsyayana, but based on manuscripts dating back to the 4th century BC.
  - The Kama-sutra recommends that women should study 64 arts, including cooking, dressing, massage and the preparation of perfumes.
  - The list also includes some less obvious arts, including conjuring, chess, bookbinding and carpentry.
  - Number 45 on the list is **mlecchita-vikalpa**, the art of secret writing, advocated in order to help women conceal the details of their liaisons.
  - One of the recommended techniques involves randomly pairing letters of the alphabet, and then substituting each letter in the original message with its partner.

Table of contents I

Table of contents II

Table of contents III

Mono-alphabetic substitution ciphers

Key idea

Generalise Caesar cipher by allowing an arbitrary substitution.

- **Permutation** of a finite set S of elements: an ordered sequence of all elements of S, each element appearing exactly once.

  6 permutations of $S = \{a, b, c\}$: *abc*, *acb*, *bac*, *bca*, *cab*, *cba*

  In general: $n!$ permutations of a set of n elements (1^{st} element can be chosen in 1 of n ways, 2^{nd} in $n - 1$ ways, etc.).
- If the "cipher" line of a Caesar cipher can be any permutation of the 26 alphabetic characters, then there are 26! ($> 4 \times 10^{26}$) possible keys.

Such an approach is referred to as a **mono-alphabetic substitution cipher**, because a single cipher alphabet (mapping from plain alphabet to cipher alphabet) is used per message.

Mono-alphabetic substitution ciphers

- Let $\mathcal{K}$ be the set of all permutations on the alphabet $\mathcal{A}$.
  Define for each $e \in \mathcal{K}$ an encryption transformation E_e on strings $m = m_1 m_2 \cdots m_n \in \mathcal{M}$ as

$$E_e(m) = e(m_1)e(m_2)\cdots e(m_n) = c_1 c_2 \cdots c_n = c$$

- To decrypt c, compute the inverse permutation $d = e^{-1}$ and

$$D_d(c) = d(c_1)d(c_2)\cdots d(c_n) = m$$

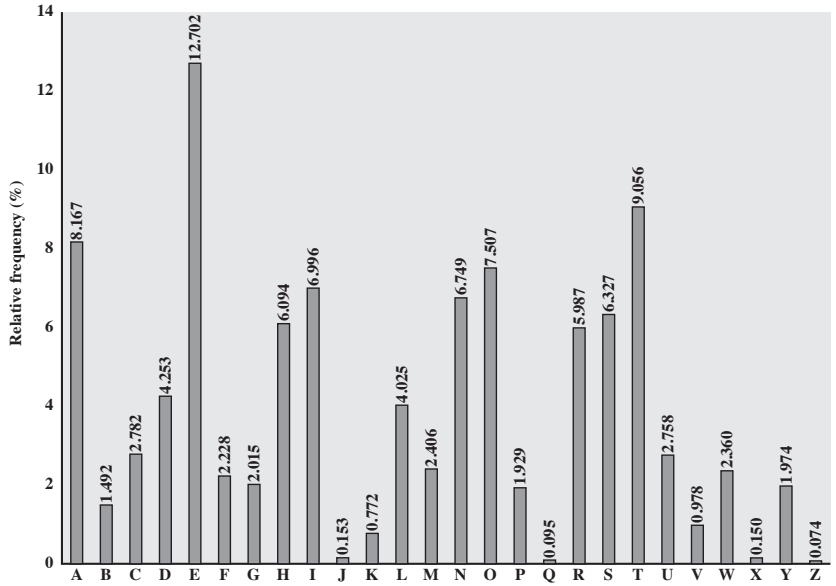
- E_e is a **mono-alphabetic substitution cipher**.

Example:

Plain:	ABCDEFGHIJKLMNOPQRSTUVWXYZ
Cipher:	DKVQFIBJWPESCXHTMYAUOLRGZN
Plaintext:	IFWEWISHTOREPLACELETTERS
Ciphertext:	WIRFRWAJUHYFTSDVFSFUUFYA

(In)security of substitution ciphers

- Key spaces are typically huge.
  26 letters $\Rightarrow 26! = 4 \times 10^{26}$ possible keys.
- This looks quite secure, doesn't it? Wrong!
- Easy to crack using frequency analysis (letters, digram, etc.).
- Frequencies for English based on data-mining books/articles:



- Easy to apply, except for short, atypical texts, e.g.,
  From Zanzibar to Zambia and Zaire, ozone zones make zebras run zany zigzags.
  $\Rightarrow$ More sophistication required to mask statistical regularities.

Example

Given ciphertext:

UZQSOVUOHXMOPVGPOZPEVSGZWSZOPFPESXUDBMETSXAIZ
VUEPHZHMDZSHZOWSFPAPPDTSVPQUZWYMXUZUHSX
EPYEPOPDZSZUFPOMBZWPFUPZHMDJUDTMOHMQ

- Count relative letter frequencies.
- Since **P** and **Z** occur most frequently, guess they correspond to **E** and **T** respectively.
- Count relative **digram** (a sequence of 2 letters, a.k.a. **digraph**) frequencies.
- Since **ZW** occurs most frequently, guess it corresponds to **TH** (which is the digram occurring most frequently in English).
- Hence **ZWP** is **THE**.
- By proceeding with trial and error finally get:

IT WAS DISCLOSED YESTERDAY THAT SEVERAL INFORMAL BUT DIRECT CONTACTS HAVE BEEN MADE WITH POLITICAL REPRESENTATIVES OF THE VIET CONG IN MOSCOW

Cryptanalysis: a little bit of history (& geography)

- The Abbasid caliphate (or dynasty), started in 750 AC, heralded golden age of Islamic civilisation (arts and sciences flourished).
- A wealthy and peaceful society, which relied on an effective system of administration, and in turn the administrators relied on secure communication achieved through the use of encryption.
  - Many administrative manuals, such as the tenth-century Adab al-Kuttab ("The Secretaries' Manual"), include sections devoted to cryptography — mainly monoalphabetic substitution ciphers.
- Invention of **cryptanalysis** required scholarship in many disciplines, including mathematics, statistics, linguistics and religion:
  - Theologians established the chronology of Muhammad's revelations in the Koran by counting the **frequencies** of words contained in each revelation and considering that certain words had evolved relatively recently.
  - They also analysed individual letters, and in particular they discovered that some letters are more common than others (e.g., a and l are the most common in Arabic).

Table of contents I

Table of contents II

Table of contents III

Homophonic substitution ciphers

- Mono-alphabetic ciphers are easy to break because they reflect the frequency data of the original alphabet.
- Countermeasure: provide multiple substitutes (i.e., *homophones*) for a single letter to make frequency analysis more difficult.

Homophonic substitution cipher

- To each $a \in \mathcal{A}$ associate a set $H(a)$ of strings of t symbols, where $H(a), a \in \mathcal{A}$ are pairwise disjoint.
- Replace each a with a randomly chosen string from $H(a)$.
- To decrypt a string c of t symbols, one must determine an $a \in \mathcal{A}$ such that $c \in H(a)$.
- The key for the cipher is the sets $H(a)$.

- **Example:** $\mathcal{A} = \{x, y\}$, $H(x) = \{00, 10\}$, and $H(y) = \{01, 11\}$. The plaintext xy encrypts to one of 0001, 0011, 1001, 1011.
- Cost: data expansion and more work for decryption.

Cryptanalysis still relatively straightforward

- Cryptanalysis relatively straightforward even with homophones:
  - each element of plaintext affects only one element of ciphertext,
  - multiple-letter patterns (e.g., digram frequencies) still survive in the ciphertext.

- Two principal methods used in substitution ciphers to lessen the extent to which the structure of plaintext survives in ciphertext:
  1. encrypt multiple letters of plaintext, e.g., the **Playfair cipher** (or the **Hill cipher**, which we will not discuss here)
  2. use multiple cipher alphabets (polyalphabetic substitution), e.g., **Vigenère cipher**

Table of contents I

Table of contents II

Table of contents III

Playfair cipher

- **Uses a 5×5 matrix of letters constructed using a keyword.**
- Example (solved by Lord Peter Wimsey in Dorothy Sayers's "Have His Carcase"):
  - Pick keyword, here: monarchy.
  - Construct matrix: fill in letters of keyword (minus duplicates) left2right & top2bottom, and remaining letters in alphabetic order, where I and J count as one letter.
  - Plaintext is encrypted two letters at a time:

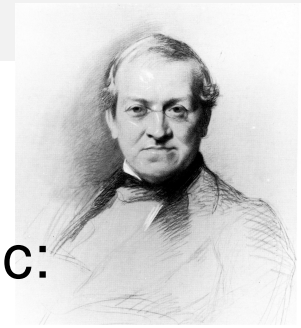
M	O	N	A	R
C	H	Y	B	D
E	F	G	I/J	K
L	P	Q	S	T
U	V	W	X	Z

1. If a pair is a repeated letter, insert filler like 'X' (e.g., "BALLOON" $\rightsquigarrow$ "BA LX LO ON"). Add an 'X' also at the end, if needed (or any other character, like the 'A' in this example).
2. If both letters fall in the same row, replace each with letter to right, wrapping back to start from end (e.g., "AR" is encrypted as "RM").
3. If both letters fall in the same column, replace each with the letter below it, wrapping to top from bottom (e.g., "MU" is encrypted as "CM").
4. Otherwise each letter is replaced by the letter in the same row and in the column of the other letter of the pair (e.g., "HS" becomes "BP" and "EA" becomes "IM", or "JM", as the encipherer wishes).

Plaintext : TH EQ UI CK BR OW NF OX IU MP SO VE RT HE LA ZY DO G
Plaintext formatted: TH EQ UI CK BR OW NF OX IU MP SO VE RT HE LA ZY DO GA
Ciphertext : PD GL XE DE DA NV OG AV EX OL PA UF DZ CF SM WD HR IN

- To decrypt, use the inverse (opposite) of rules 2 and 3, and the 1st as-is (dropping any extra "X"s that do not make sense in the final message when finished) and the 4th as-is.

Playfair cipher: security



- Security much improved over monoalphabetic: $26 \times 26 = 676$ digrams vs. 26 letters.
- Would need a 676 entry frequency table to analyse, and correspondingly more ciphertext.
- Invented in 1854 by British scientist Sir Charles Wheatstone, but it bears the name of his friend Baron Playfair of St. Andrews, who championed the cipher at the British foreign office).
- Playfair cipher was for a long time considered unbreakable:
  - used as the standard field system by British Army in WWI and still partly used by U.S. Army and other Allied forces during WWII.
- However, breaking it is relatively easy:
  - still leaves much of the structure of the plaintext language intact
  - a few hundred letters of ciphertext are generally sufficient

Table of contents I

Table of contents II

Table of contents III

Polyalphabetic substitution ciphers

Polyalphabetic substitution cipher

- **Use different monoalphabetic substitutions as one proceeds through the plaintext message.**
  Idea (L.B. Alberti): conceal distribution using family of mappings.
- Two general features common to all such ciphers:
  1. A set of related monoalphabetic substitution rules is used.
  2. A key determines which particular rule is chosen for a given transformation.

- Best known, and perhaps simplest, is the **Vigenère cipher**, by Blaise de Vigenère, from court of Henry III of France (16^{th} century).
  - Set of related monoalphabetic substitution rules consists of the 26 Caesar ciphers with shifts of 0 through 25.
  - Each cipher is denoted by a key letter, which is the ciphertext letter that substitutes for the plaintext letter A.
    Thus, a Caesar cipher with a shift of 3 is denoted by key value D.

Vigenère cipher: idea

A polyalphabetic substitution cipher based on a *tableau* where each row is a Caesar Cipher with incremental shift:

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

Vigenère cipher: details

- Assume
  - a sequence of plaintext letters $P = p_0, p_1, p_2, \ldots, p_{n-1}$,
  - a key consisting of the sequence of letters $K = k_0, k_1, k_2, \ldots, k_{m-1}$.

  Typically $m < n$.
- Sequence of ciphertext letters $C = C_0, C_1, C_2, \ldots, C_{n-1}$:

$$
\begin{aligned}
C &= C_0, C_1, C_2, \ldots, C_{n-1} \\
&= E(K, P) \\
&= E((k_0, k_1, k_2, \ldots, k_{m-1}), (p_0, p_1, p_2, \ldots, p_{n-1})) \\
&= (p_0 + k_0) \mod 26, (p_1 + k_1) \mod 26, \ldots, \\
&\quad (p_{m-1} + k_{m-1}) \mod 26, (p_m + k_0) \mod 26, (p_{m+1} + k_1) \mod 26, \\
&\quad \ldots (p_{2m-1} + k_{m-1}) \mod 26, \ldots \\
C_i &= (p_i + k_{i \bmod m}) \mod 26
\end{aligned}
$$

  - First letter of the key is added to first letter of plaintext, mod 26, second letters are added, and so on through the first m letters of plaintext.
  - For next m letters of the plaintext, the key letters are repeated.
  - Process continues until all of plaintext sequence is encrypted.
- Decryption: $p_i = (C_i - k_{i \bmod m}) \mod 26$

Vigenère cipher: example and strength

- Needs a key as long as message (usually: a repeating keyword).
- Example: if the keyword is “deceptive", the message “we are discovered save yourself" is encrypted as

```
key:          deceptivedeceptivedeceptive
plaintext:    wearediscoveredsaveyourself
ciphertext:   ZICVTWQNGRZGVTWAVZHCQYGLMGJ
```

Expressed numerically:

key	3	4	2	4	15	19	8	21	4	3	4	2	4	15
plaintext	22	4	0	17	4	3	8	18	2	14	21	4	17	4
ciphertext	25	8	2	21	19	22	16	13	6	17	25	6	21	19

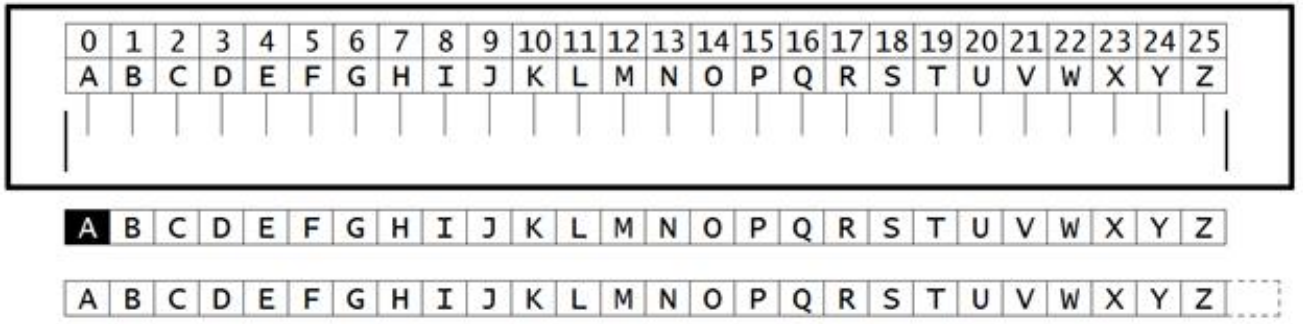
key	19	8	21	4	3	4	2	4	15	19	8	21	4
plaintext	3	18	0	21	4	24	14	20	17	18	4	11	5
ciphertext	22	0	21	25	7	2	16	24	6	11	12	6	9

- Strength: multiple ciphertext letters for each plaintext letter, one for each unique letter of the keyword.
  - Hence, letter frequency information is obscured.
- However, not all knowledge of the plaintext structure is lost.
  - Better than Playfair, but considerable frequency info remains.

Aids for en/de-cryption with polyalphabetic ciphers

- Implementing polyalphabetic ciphers by hand can be very tedious.
- Various aids were devised to assist the process, e.g., **Saint-Cyr Slide** is a simple manual aid:
  - a slide with repeated alphabet,
  - line up plaintext “A" with key letter, e.g., “C",
  - then read off any mapping for key letter.

Can bend round into a cipher disk or expand into a Vigenère Tableau.



(J.-G.-H.-V.-F.-A.-A. Kerckhoffs von Nieuwenhof popularised and named it, after the French National Military Academy where the methods were taught)

Kasiski Method

- For some centuries the Vigenère cipher was le *chiffre indéchiffrable* (the unbreakable cipher).
- Broken by Charles Babbage (“inventor" of the computer) in 1854 but kept secret (possibly because of the Crimean War).
- Method independently reinvented by Friedrich Kasiski (Prussia, 1863).
  - repetitions in ciphertext give clues to period
  - so find same plaintext an exact period apart which results in the same ciphertext
  - of course, could also be random fluke
  - e.g., repeated “VTW” in previous example
  - suggests size of 3 or 9
  - then attack each monoalphabetic cipher individually using same techniques as before
- See Stalling’s “Cryptography and Network Security” book for a sketch of a method of breaking Vigenère.

Zimmermann telegrams: a little bit of history

- However, lack of major advances meant that various polyalphabetic substitution ciphers were used into the 20th century.
- One very famous incident was the breaking of the Zimmermann telegram in WWI which resulted in the USA entering the war.
  - A 1917 diplomatic proposal from the German Empire for Mexico to join in alliance if USA entered WWI against Germany.
  - Intercepted and decoded by British cryptographers of Room 40.

"We intend to begin on the first of February unrestricted submarine warfare. We shall endeavor in spite of this to keep the United States of America neutral. In the event of this not succeeding, we make Mexico a proposal of alliance on the following basis: make war together, make peace together, generous financial support and an understanding on our part that Mexico is to reconquer the lost territory in Texas, New Mexico, and Arizona. The settlement in detail is left to you. You will inform the President of the above most secretly as soon as the outbreak of war with the United States of America is certain and add the suggestion that he should, on his own initiative, invite Japan to immediate adherence and at the same time mediate between Japan and ourselves. Please call the President's attention to the fact that the ruthless employment of our submarines now offers the prospect of compelling England in a few months to make peace." Signed, ZIMMERMANN

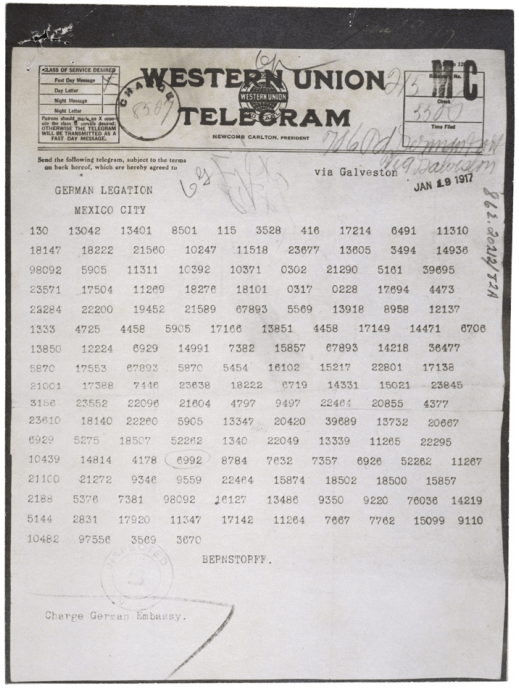

WESTERN UNION
TELEGRAM
via Galveston
JAN 19 1917
GERMAN LEGATION
MEXICO CITY
130 13042 13401 8501 115 3528 416 17214 6491 11310
18147 18222 21560 10247 11518 23677 13605 3494 14936
98092 5905 11311 10392 10371 0302 21290 5161 39695
23571 17504 11269 18276 18101 0317 0228 17694 4473
23284 22200 19452 21589 67893 5569 13918 8958 12137
1333 4725 4458 5905 17166 13851 4458 17149 14471 6706
13850 12224 6929 14991 7382 15857 67893 14218 36477
5870 17553 67893 5870 5454 16102 15217 22801 17138
21001 17388 7446 23638 18222 6719 14331 15021 23845
3156 23552 22096 21604 4797 9497 22464 20855 4377
23610 18140 22260 5905 13347 20420 39689 13732 20667
6929 5275 18507 52262 1340 22049 13339 11265 22295
10439 14814 4178 6992 8784 7632 7357 6926 52262 11267
21100 21272 9346 9559 22464 15874 18502 18500 15857
2188 5376 7381 98092 16127 13486 9350 9220 76036 14219
5144 2831 17920 11347 17142 11264 7667 7762 15099 9110
10482 97556 3569 3670
BEPNSTORFF.
Charge German Embassy.

Table of contents I

Table of contents II

Table of contents III

Vernam cipher: XOR

- Gilbert Vernam (AT&T engineer, 1918) proposed a system where keyword is as long as plaintext and has no statistical relationship to it.
- It works on binary data (bits) rather than letters, using **XOR** $\oplus$:

$$\begin{aligned} 0 \oplus 0 &= 0 \\ 0 \oplus 1 &= 1 \\ 1 \oplus 0 &= 1 \\ 1 \oplus 1 &= 0 \end{aligned}$$

so that

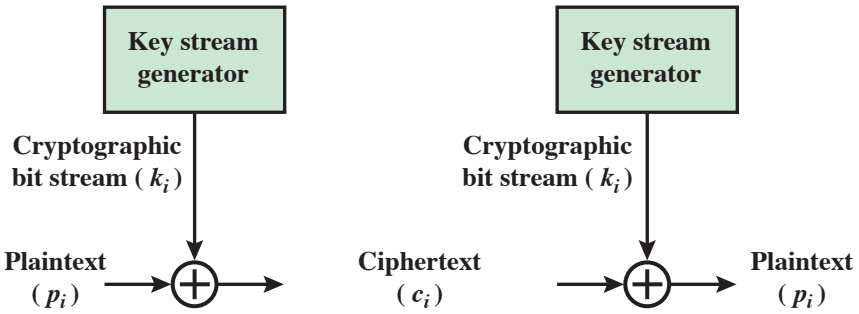
$$\begin{aligned} a \oplus a &= 0 \\ a \oplus 0 &= a \\ a \oplus b &= b \oplus a \\ a \oplus b \oplus b &= a \\ (a \oplus b) \oplus c &= a \oplus (b \oplus c) \end{aligned}$$

XOR can be used as polyalphabetic cipher:

$$\begin{aligned} P \oplus K &= C \\ C \oplus K &= P \end{aligned}$$

Vernam cipher: idea

$$c_i = p_i \oplus k_i$$
$$p_i = c_i \oplus k_i$$



- $p_i/c_i/k_i = i^{\text{th}}$ binary digit of plaintext/ciphertext/key.
- Ciphertext generated by bitwise XOR of plaintext and key.
- Decryption: simply same bitwise operation (by properties of XOR).
- Essence of this cipher: means of construction of the key.
- Vernam proposed use of a running loop of tape that eventually repeated the key (hence: very long but repeating keyword).
- Difficult to break if key is long, but still breakable with sufficient ciphertext, use of known or probable plaintext sequences, or both.

A small example of how perfect secrecy is achievable

- Mr X is about to make a decision that will have serious repercussions on the share value of a company.
  - If he makes the decision “buy”, then the shares will increase in value.
  - If he makes the decision “sell”, then the shares will collapse.
- Suppose also that it is publicly known that Mr X will soon be transmitting one of these two messages to his broker.
  - Anyone who received this decision before the broker would have the opportunity to use that information to either make a profit or to avoid a disastrous loss.
- At any time, anyone is free to guess what the message will be and act accordingly.
  - They have a 50% chance of being right... such an action would be nothing more than gambling.

A small example of how perfect secrecy is achievable

- Mr X wants to be able to send his message over a **public** network.
- In order to protect their interests, Mr X and his broker decide to encrypt the message that will convey the decision.
- Since a substitution cipher would be easy to break with such a short (and predictable) message, they decide to use a system with two keys, K_1 and K_2 are **equally likely**.
  - K_1 encrypts "buy" to 0 and "sell" to 1:
    $$E_{K_1}(\text{"buy"}) = 0 \text{ and } E_{K_1}(\text{"sell"}) = 1.$$
  - K_2 encrypts "buy" to 1 and "sell" to 0:
    $$E_{K_2}(\text{"buy"}) = 1 \text{ and } E_{K_2}(\text{"sell"}) = 0.$$
- If the attacker intercepts a 0, then all that he can deduce is that the message might be "sell" if K_2 was used, or "buy" if K_1 was used.
- Since each key is equally likely, the attacker is forced to guess which key was used: the chances of guessing correctly are 50%.

A small example of how perfect secrecy is achievable

In essence:

- Before the ciphertext was intercepted, the attacker's only option was to try to guess the message.
- Once the ciphertext was intercepted, the attacker could also guess the key.
- Since the number of keys is the same as the number of messages, the chances of either guess being correct are equal.

This is **perfect secrecy** (but, as we will see, it comes at a price).

Table of contents I

Table of contents II

Table of contents III

One-time pad

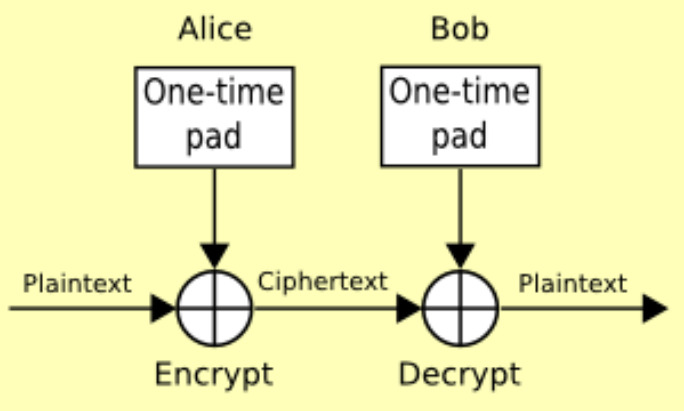
One-time pad

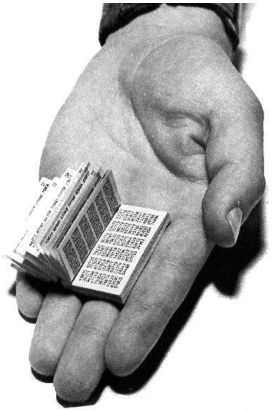
(improvement to Vernam proposed by Joseph Mauborgne, US Army Signal Corp officer)

Use a truly **random key** that is

- as long as the message, so that the key need not be repeated,
- used to encrypt and decrypt a single message, and then discarded.

- Each new message P requires a new key of same length as P.
- Produces random output with no statistical relation to plaintext.
- **Unbreakable: C contains no information whatsoever about P.** Only cryptosystem that exhibits so-called *perfect secrecy*.





One-time pad: example

- Consider a scheme with 27 characters (26 plus space character).
- Exhaustive search of all possible keys yields many legible plaintexts, with no way of knowing which was the intended one (NOTE that the following don't really match as there is a +1):

```
ciphertext: ANKYODKYUREPFJBYOJDSPLREYIUNOFDOIUERFPLUYTS
key:        pxlmvmsydofuyrvzwc tnlebnecvgdupahfzzlmnyih
plaintext:  mr mustard with the candlestick in the hall

ciphertext: ANKYODKYUREPFJBYOJDSPLREYIUNOFDOIUERFPLUYTS
key:        mfugpmiydgaxgoufhkllmhsqdqogtewbqfgyovuhwt
plaintext:  miss scarlet with the knife in the library
```

- Suppose a cryptanalyst managed to find these two keys.
  - Two plausible plaintexts are produced.
  - Which is the correct decryption (i.e., which is the correct key)?
- If the actual key were produced in a truly random fashion, then cryptanalyst cannot say that one key is more likely than the other.
- Given any P of equal length to C, there is K that produces that P.

No patterns or regularities: if stream of characters that constitute K is truly random, then so will be stream of characters that constitute C.

One-time pad: practical difficulties

- Two fundamental practical difficulties:
  - Making large quantities of random keys.
  - Key distribution and protection, where for every message to be sent, a key of equal length is needed by both sender and receiver.
- Hence:
  - limited utility,
  - useful primarily for low-bandwidth channels requiring very high security (Moscow–Washington communication previously secured this way).

Table of contents I

Table of contents II

Transposition ciphers

Transposition cipher

Perform some sort of permutation on the plaintext letters.
Works on blocks of letters of the plaintext.

More formally:

- For block length t, let $\mathcal{K}$ be the set of permutations on $\{1, \ldots, t\}$. For each $e \in \mathcal{K}$ and $m \in \mathcal{M}$

$$E_e(m) = m_{e(1)} m_{e(2)} \cdots m_{e(t)}$$

- The set of all such transformations is called a transposition cipher.
- To decrypt $c = c_1 c_2 \cdots c_t$ compute $D_d(c) = c_{d(1)} c_{d(2)} \cdots c_{d(t)}$, where d is inverse permutation.
- Letters unchanged so one can exploit frequency analysis for dipthongs, tripthongs, words, etc.

Let us see three examples.

Table of contents I

Table of contents II

Rail fence cipher

Rail fence cipher (a.k.a. Zig-zag cipher)

Plaintext is written down as a sequence of diagonals and then read off as a sequence of rows. (Beware: variants exist under the same name.)

- For example, to encipher the message

  MEET ME AFTER THE TOGA PARTY

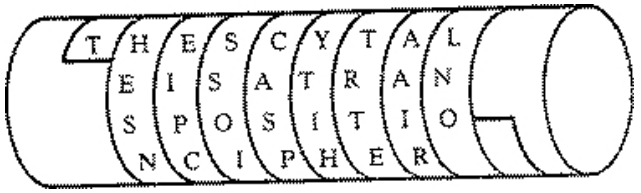
  with a rail fence of depth 2, we write:

  M E M A T R H T G P R Y
  E T E F E T E O A A T

  so that the ciphertext is

  M E M A T R H T G P R Y E T E F E T E O A A T

- Idea goes back to Greek scytale.



- Trivial to cryptanalyze.
  Hence, more clever transposition ciphers have been devised.

Rail fence cipher

- As another example, to encipher the message

  WE ARE DISCOVERED FLEE AT ONCE

  with a rail fence of depth 3, we write:

  W E C R L T E
  E R D S O E E F E A O C
  A I V D E N

  so that the ciphertext is

  W E C R L T E E R D S O E E F E A O C A I V D E N
- **Decryption**: reconstruct the diagonal grid used to encrypt the message.
  - Start by making a grid with as many rows as the key is, and as many columns as the length of the ciphertext.
  - Place the first letter in the top left square, and dashes diagonally downwards where the letters will be.
  - When we get back to the top row, we place the next letter in the ciphertext.
  - Continue like this across the row, and start the next row when you reach the end.

Rail fence cipher: decryption example

- If we receive the ciphertext “TEKOOHRACIRMNREATANFTETYTGHH” encrypted with a key of 4, we have a table with 4 rows because the key is 4, and 28 columns as the ciphertext has length 28.
- We start by placing the “T” in the first square, and then dash the diagonal down spaces until we get back to the top row, and place the “E” here. Continuing to fill the top row we get:

```
T       E       K       O       O
  -   -   -   -   -   -   -   -   -
    -   -   -   -   -   -   -   -
      -       -       -       -
```

- Continuing this row-by-row, we get the successive stages shown below:

```
T       E       K       O       O
  H   R   A   C   I   R   M   N   R
    -   -   -   -   -   -   -   -
      -       -       -       -

T       E       K       O       O
  H   R   A   C   I   R   M   N   R
    E   A   T   A   N   F   T   E   T
      -       -       -       -
```

Rail fence cipher: decryption example

- and finally

```
T       E       K       O       O
  H   R   A   C   I   R   M   N   R
    E   A   T   A   N   F   T   E   T
      Y       T       G       H       H
```

- From this we can now read the plaintext off following the diagonals to get:

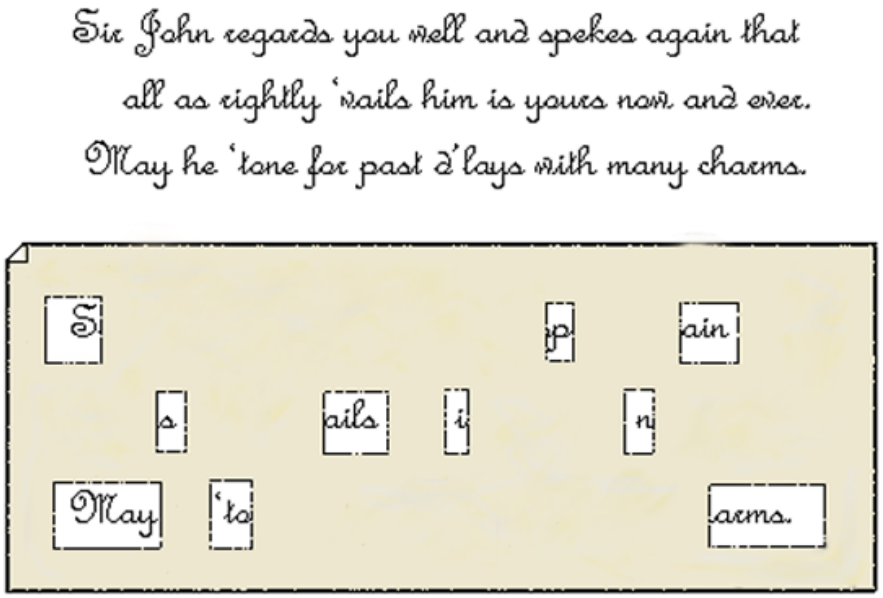
THEY ARE ATTACKING FROM THE NORTH

Table of contents I

Table of contents II

Cardano grille

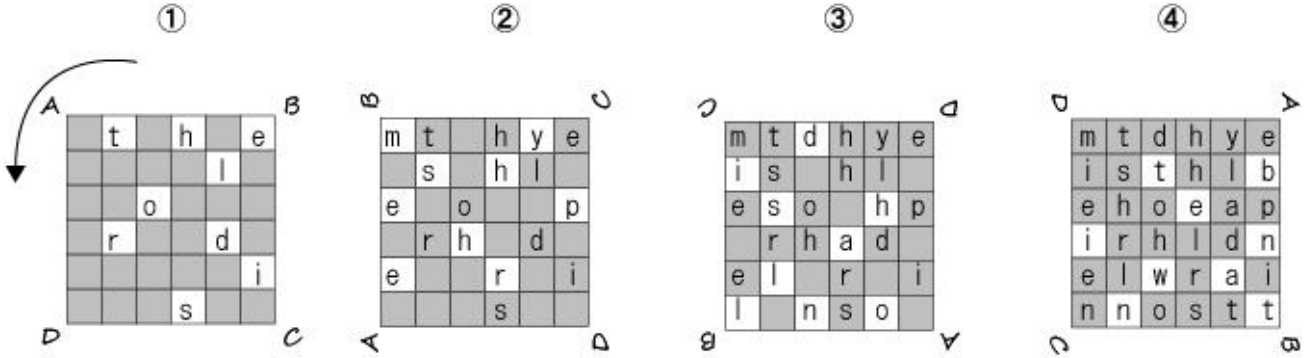
- Girolamo Cardano (Italy, 16th cent., mathematician and Kabbalist).
- **Use a mask ("grille") with precut holes.**
  - Encoder writes plaintext in holes, removes mask, fills remainder with blind text, retaining appearance of an innocuous message.
  - Decryption: recipient must possess an identical mask (or must know spacing that created it).



- Skipping letters within an otherwise plausible ciphertext.
- Is example of steganography, but provides basis for transposition.

Rotating (turning) grille (18th century)

- **Grille**: a sheet with a grid of squares, some of which are cut out.
- Example:
  - 6 $\times$ 6 grid of squares, of which 9 are cut out.
  - Plaintext: The Lord is my shepherd. I shall not be in want.
  - Write first 9 letters in each square cut out, left2right, top2bottom.
  - Turn grille by 90° in predetermined direction (e.g., counterclockwise).
  - Write next 9 letters... until grille is filled.
  - Read ciphertext (left2right, top2bottom):
    mtdhyeisthlbehoeapirhldnelwrainnostt.

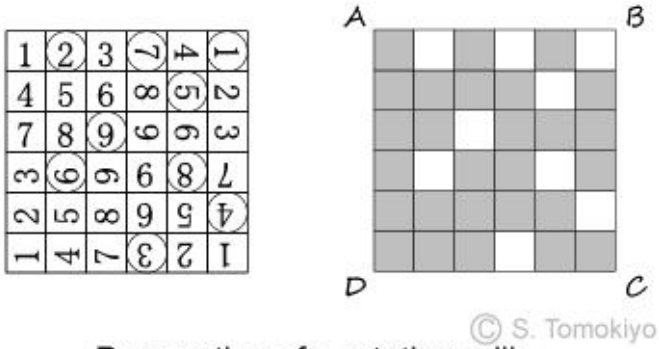


Enciphering with a rotating grille. © S. Tomokiyo

- Deciphering: write ciphertext in 6 $\times$ 6 grid and use rotating grid.

Rotating (turning) grille: preparation

- *When enciphering, every time the grille is turned, the cut out squares are precisely positioned at squares not yet filled.*
- Procedure to select squares of the grille to be cut out:
  - divide grid in 3 $\times$ 3 quadrants, number squares of each quadrant,
  - among four squares numbered “1”, select one to be cut out (e.g., rightmost square of top row) to ensure that each of these four squares is exposed exactly once during enciphering,
  - among four squares numbered “2”, one is selected to be cut out...



Preparation of a rotating grille.

By choosing thus exactly one square to be cut out from among the four squares bearing the same number, it can be ensured that each square can be filled at one of the four rotating positions.

Table of contents I

Table of contents II

Columnar transposition cipher

- Write message in a rectangle, row by row, and read message off, column by column, but *permute the order of the columns*.
- The order of the columns thus is the key to the algorithm.
- For example, with key 4312567 (and with padding to fill the grid)

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						

  To encrypt, start with column labeled 1, write down all letters in that column, proceed with column labeled 2, etc.
- A pure transposition cipher is easily recognized and attacked: ciphertext has same letter frequencies as original plaintext.

Multiple-stage columnar transposition cipher

- A transposition cipher (columnar or not) can be made significantly more secure by **performing more than one stage of transposition**.
- Result: more complex permutation that is not easily reconstructed.
- Let's reencrypt foregoing message

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						

using the same algorithm:

Key:	4	3	1	2	5	6	7
Plaintext:	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
Ciphertext:	NSCYAUOPTTWLTMDNAOIEPAXTTOKZ						

Multiple-stage columnar transposition cipher

- To visualize this double transposition, designate the 28 letters in original plaintext by their position:

  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

  First transposition, still quite regular structure (+7 in blocks of 4!):

  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

  Second: less structured permutation (cryptanalysis more difficult):

  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

Multiple stages of encryption can produce an algorithm that is significantly more difficult to cryptanalyze

This is as true of substitution ciphers as it is of transposition ciphers.

Before we see examples of this (rotor machines, DES, ...), let us say a few words about steganography.

Table of contents I

Table of contents II

Steganography: techniques

Steganography

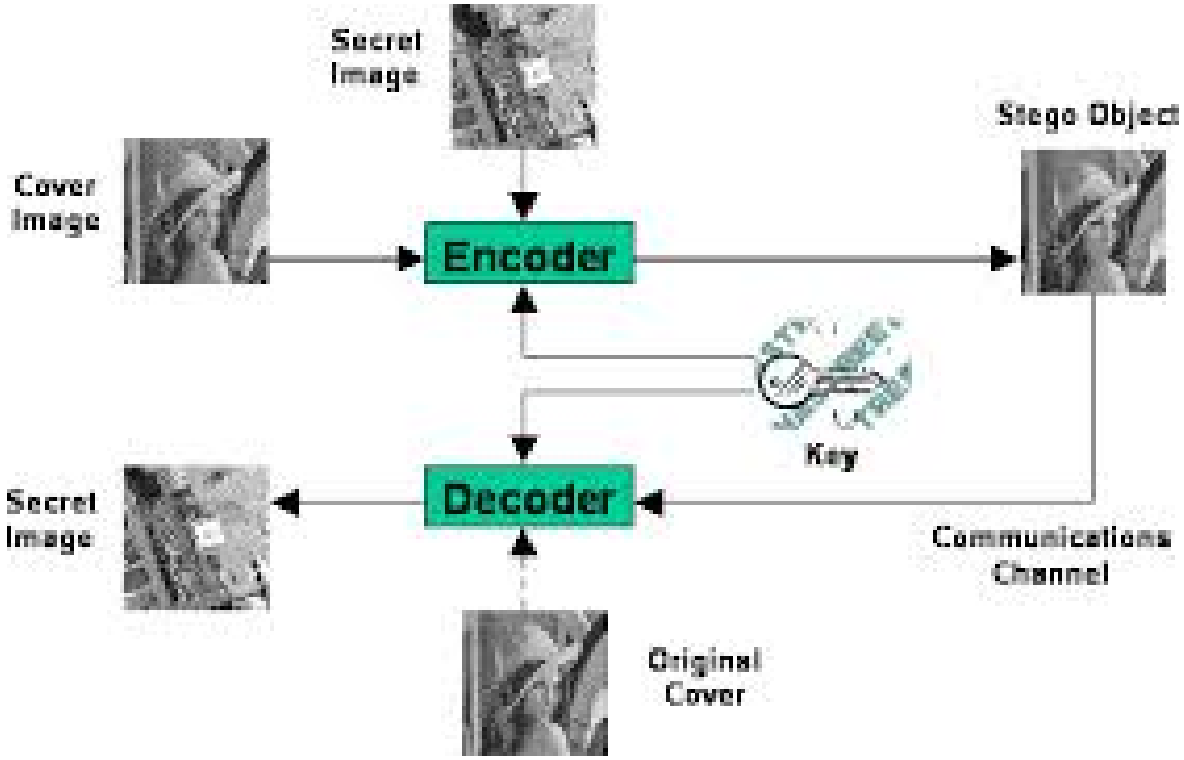
- **Conceal the existence of the message.**
- Whereas the methods of cryptography render the message unintelligible to outsiders by various transformations of the text.

There are several softwares available online.
Example techniques:

- **Arrangement of words or letters** within an apparently innocuous text spells out the real message (e.g., sequence of first letters of each word of the overall message spells out the hidden message).
- **Invisible ink**: Substances used for writing that leave no visible trace until heat or some chemical is applied to the paper.
- **Pin punctures**: Small pin punctures on selected letters are ordinarily not visible unless the paper is held up in front of a light.

Steganography: techniques

- **Least significant bits of frames of an image** (or sound file), e.g.
  - Resolution 2048 $\times$ 3072 pixels.
  - Each pixel contains 24 bits of RGB color information.
  - Least significant bit of each 24-bit pixel can be changed without greatly affecting the quality of the image.
  - Result: hide a 2.3-megabyte message in a single digital snapshot.



Steganography: advantages and disadvantages

- Disadvantages with respect to encryption:
  - Requires quite a lot of overhead to hide a relatively few bits of information.
  - Once the system is discovered, it becomes virtually worthless.
    - Can be overcome if insertion method depends on some sort of key.
    - Alternatively, a message can be first encrypted and then hidden using steganography.

- Advantages with respect to encryption:
  - Can be employed by parties who have something to lose should the fact of their secret communication (not necessarily the content) be discovered.
  - Encryption flags traffic as important or secret or may identify the sender or receiver as someone with something to hide.

Watermarking and DRM

Watermarking: hiding digital information in a carrier signal

- A kind of marker covertly embedded in a noise-tolerant signal such as audio (e.g, mp3 file) or image data (e.g., mpeg file).
- **Digital watermarks** typically used
  - to verify authenticity or integrity of carrier signal or
  - to show the identity of its owners.

  Trace copyright infringements, authenticate banknotes, etc.
- Like traditional watermarks, a digital watermark is
  - **perceptible under certain conditions**, i.e., using some algorithm,
  - **imperceptible anytime else** (it should not distort carrier signal).

Watermarking and DRM

Digital Rights Management (DRM)

A class of technologies that are used by hardware manufacturers, publishers, copyright holders, and individuals with the intent to control the use of digital content and devices after sale

- Ongoing debate: does DRM
  - help fight piracy
    or
  - somehow encourage piracy
    ?
- Usability often an enemy of security.
  - When they are seen as conflicting (and not as complementary or even collaborating) requirements.

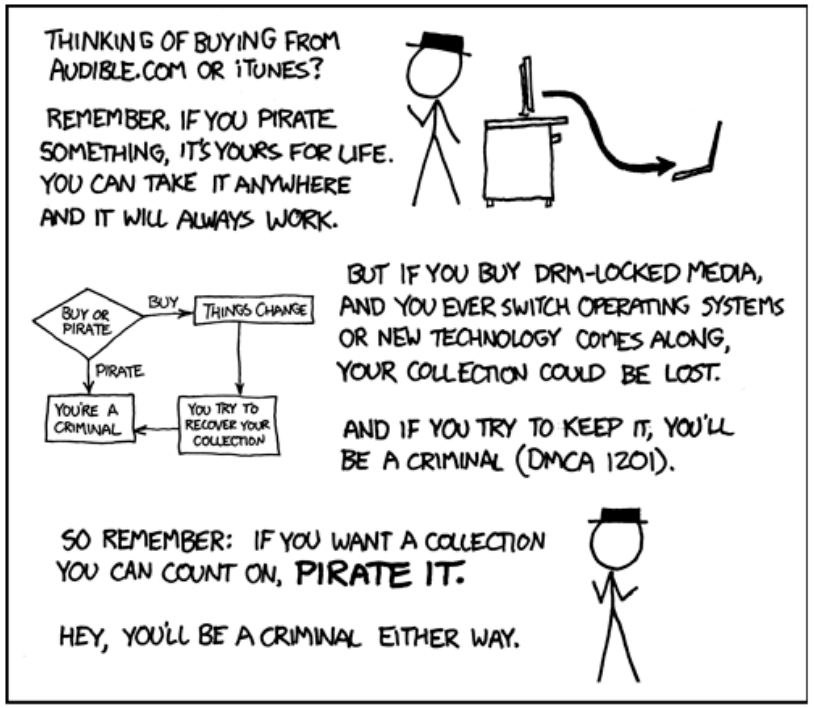


Table of contents I

Table of contents II

Composite (product) ciphers: idea

- Ciphers based on just substitutions or transpositions are not secure.
- Ciphers can be combined. However . . .
  - two substitutions are really only one more complex substitution,
  - two transpositions are really only one transposition,
  - but a substitution followed by a transposition makes a new harder cipher.
- **Product ciphers chain substitution-transposition combinations.**
- Bridge from classical to modern ciphers.
- Difficult to do by hand $\leadsto$ invention of cipher machines.

Rotor machines

- Most common complex ciphers in use before modern ciphers.
- Widely used in WW2:
  - Enigma (Germany), Hagelin (Allied Forces), Purple (Japan).
- Implemented a very complex, varying substitution cipher.
- **Used a series of cylinders, each giving one substitution, which rotated and changed after each letter was encrypted.**
  - With 3 cylinders: $26^3 = 17576$ alphabets.
  - For every complete rotation of inner cylinder, middle cylinder rotates one pin position; for every complete rotation of middle cylinder, outer cylinder rotates one pin position.

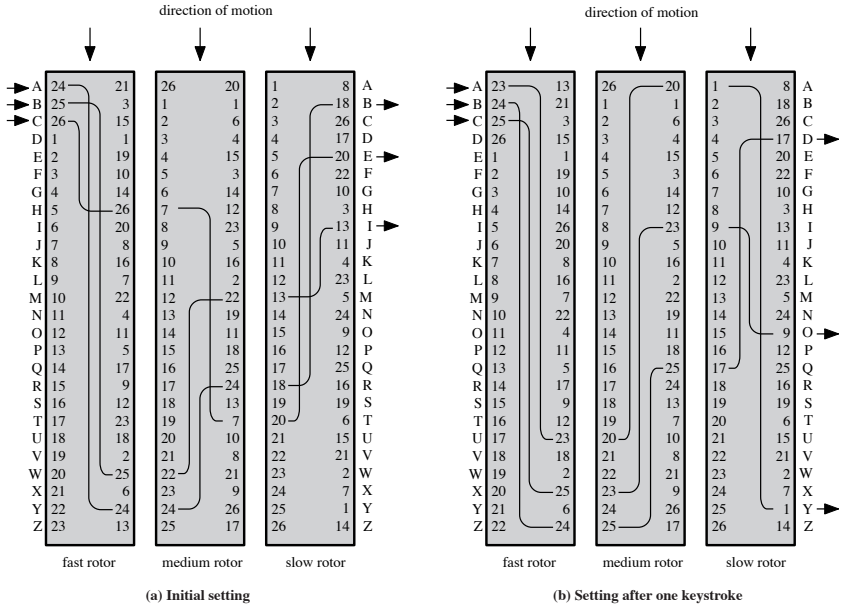


Table of contents I

Table of contents II

Motivation for the Feistel Cipher Structure

- Many symmetric block encryption algorithms in current use are based on a structure called **Feistel block cipher** (IBM, '70s). Let's thus examine the design principles of the Feistel cipher.

Block cipher: encrypts n-bit plaintext into n-bit ciphertext.

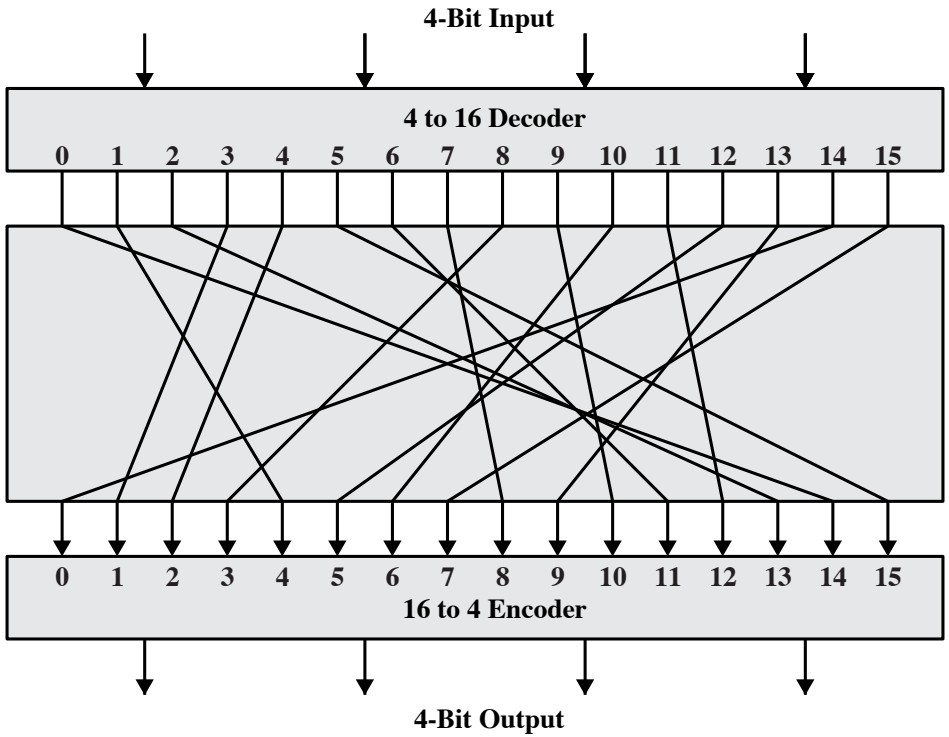
- There are 2^n possible different plaintext blocks.
- **If limit to reversible mappings: $2^n!$ different transformations.**
  - For encryption to be **reversible** (i.e., decryption to be possible), each of 2^n plaintext blocks must produce a unique ciphertext block.
  - Example for $n = 2$: ciphertext 01 could come from plaintext 10 or 11

Reversible mapping	
Plaintext	Ciphertext
00	11
01	10
10	00
11	01

Irreversible mapping	
Plaintext	Ciphertext
00	11
01	10
10	01
11	01

$2^n!$ since 1st plaintex: 2^n cipher-blocks to choose from, 2nd: 2^n-1, ...

Ideal (most general form of) block cipher: e.g., $n = 4$



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

- Can be used to define any reversible mapping plaintext-ciphertext.
- 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.
- Encryption/decryption mappings can be defined by a tabulation.

Ideal (most general form of) block cipher: problems

- **Small block size** (e.g., $n = 4$): equivalent to a classical substitution cipher and thus easily attackable.
- **Large block size**: not practical implementation and performance.
  - Mapping is the key.
  - Block n: key $n \times 2^n$ bits.
    - Block $n = 4$: key 64 bits = (4 bits) $\times$ (16 rows).
    - Block $n = 64$: key 2^{70} bits = $64 \times 2^{64} \approx 10^{21}$!
- Feistel's suggestion: invertible product cipher, i.e.,
  - approximation to ideal block cipher for large n, built out of components that are easily realizable.

Feistel cipher

Product cipher

Execution of 2 or more simple ciphers in sequence so that the result (product) is cryptographically stronger than any of the component ciphers.

- Idea: cipher with k-bit key and n-bit blocks, allowing a total of 2^k possible transformations rather than $2^n!$.
- Alternates substitutions and permutations (transpositions):

  Substitution: Each plaintext element or group of elements is uniquely replaced by a corresponding ciphertext element or group of elements.

  Permutation: A sequence of plaintext elements is replaced by a permutation of that sequence.

Permutation changes order of elements without adding/deleting/replacing them.

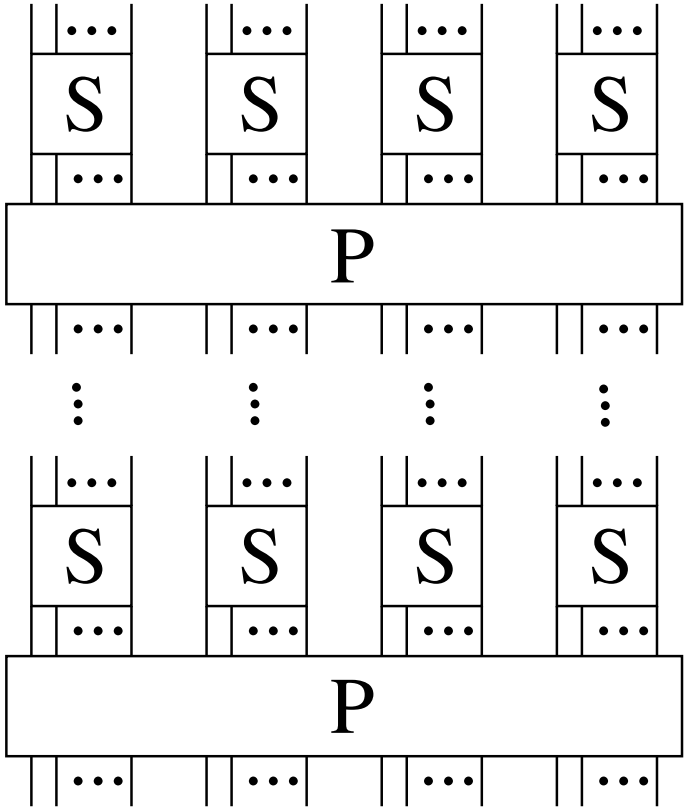
S-boxes and P-boxes: confusion and diffusion

Feistel cipher is practical application of a proposal by Shannon to develop a product cipher that alternates *confusion* and *diffusion* functions.

Product ciphers chain combinations of substitutions and transpositions.

- **S-Boxes** “confuse" input bits.
- **P-Boxes** “diffuse” bits across S-box inputs.

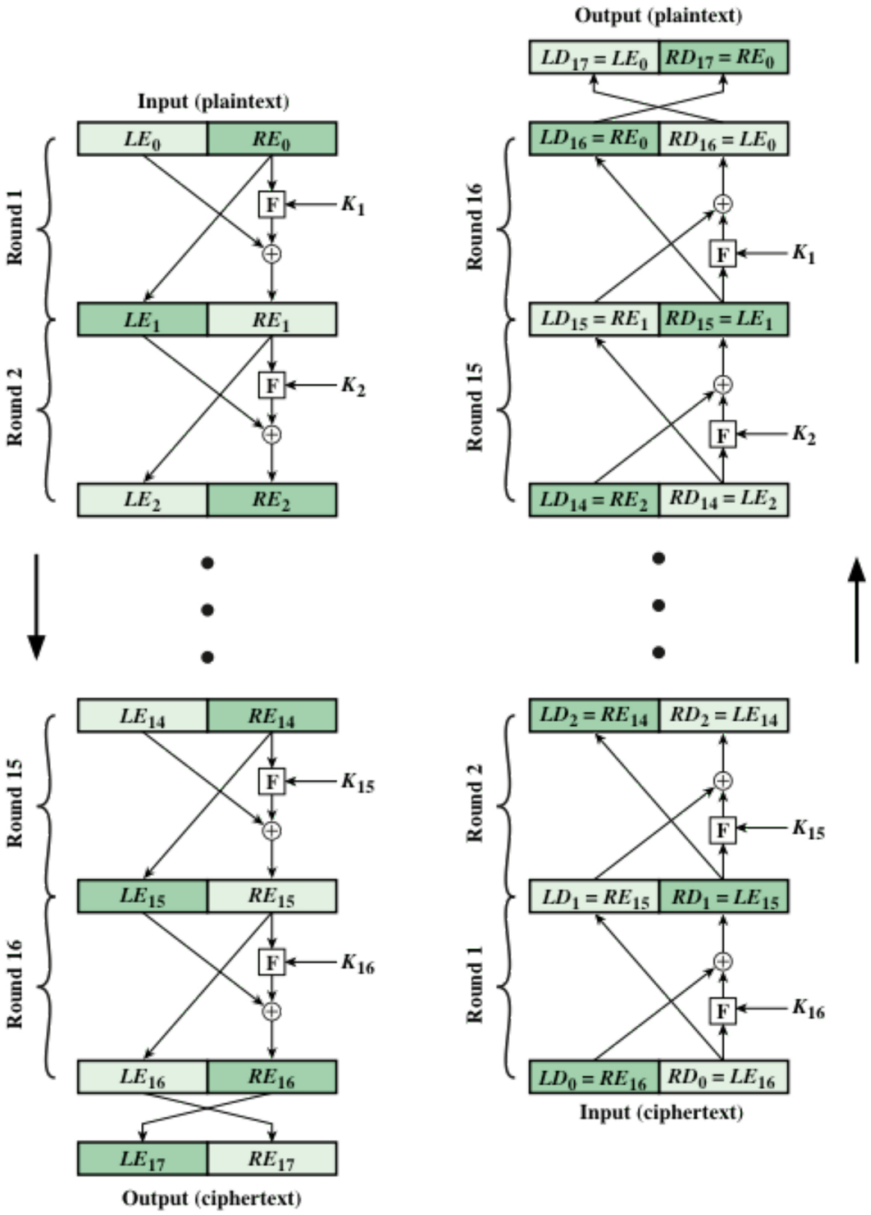
But note that permutation does not directly equate with diffusion (permutation, by itself, does not change the statistics of the plaintext at the level of individual letters or permuted blocks).



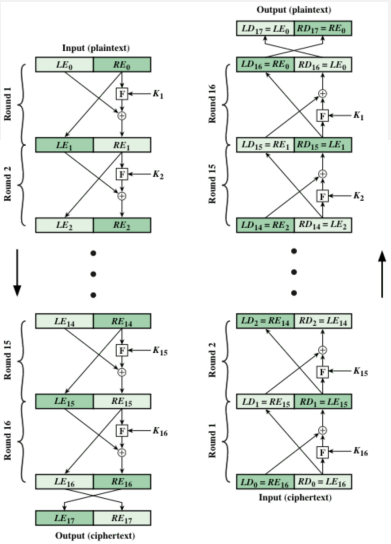
Feistel encryption / decryption (16 rounds)

In a nutshell:

- **Partition** input block into two halves
- Process through **multiple rounds** which
  - perform a **substitution** on left data half
    (based on round function of right half and subkey)
  - then have **permutation** swapping halves

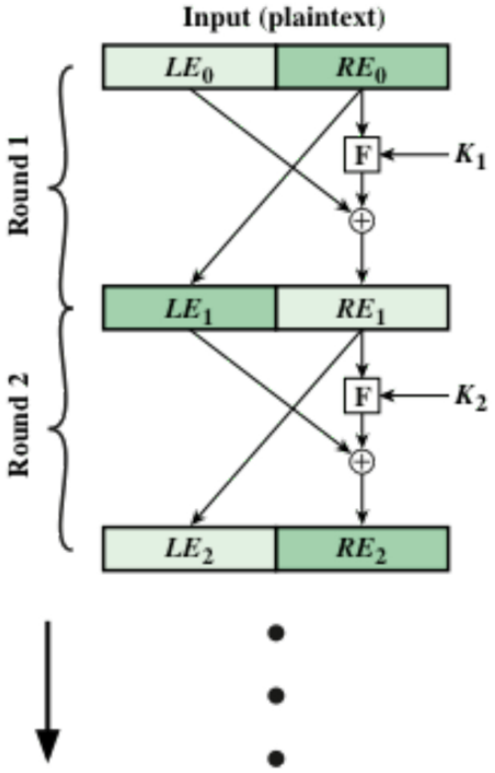


Feistel encryption (16 rounds): details



- **Encryption input**:
  - $2w$-bit plaintext block
  - key K
- **Plaintext block divided into 2 halves LE_0 and RE_0**, which
  - pass through n rounds of processing (here $n = 16$) and then
  - combine to produce ciphertext block
- Each **round** i has as inputs
  - LE_{i-1} and RE_{i-1} derived from previous round
  - subkey K_i derived from overall K.

Subkeys K_i different from K and each other

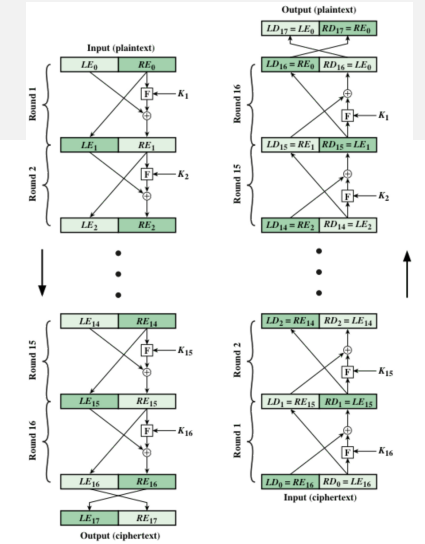


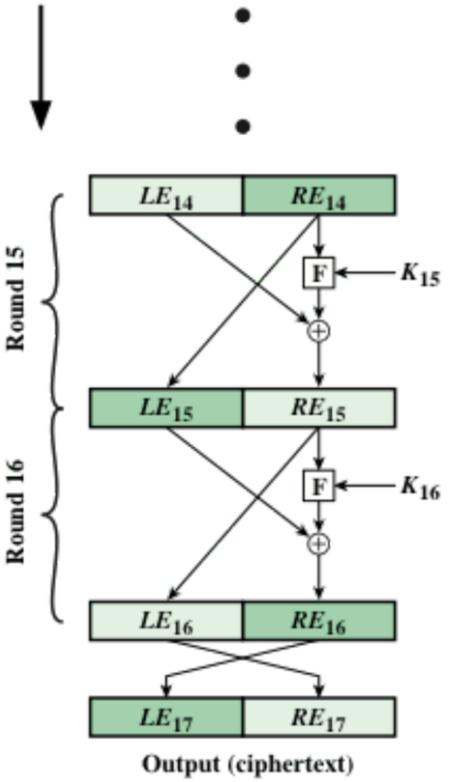
Feistel encryption (16 rounds): details

- A **substitution** is performed on LE_i by applying a **round function** F to RE_i and then XORing output with LE_i
- F has same general structure for each round but is parameterized by round subkey K_i

$$\mathrm{F}(RE_i, K_{i+1}) : w \text{ bits} \times y \text{ bits} \rightarrow w \text{ bits}$$

- Then a **permutation** is performed: interchange of two halves of data.
- Final round followed by an interchange that undoes interchange that is part of final round (to simplify decryption).

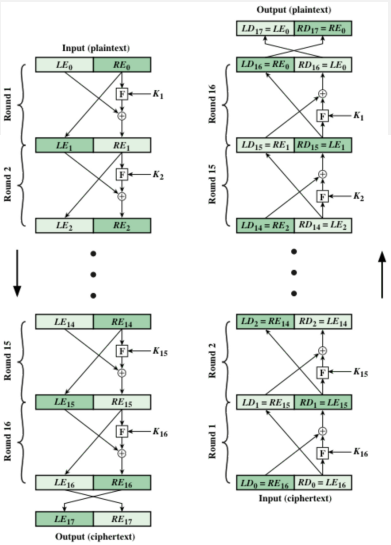




Parameters and design features of Feistel cipher

- **Block size**:
  - Larger means greater security but reduced en-/de-cryption speed.
  - Greater security achieved by greater diffusion.
  - Traditionally: 64-bit blocks reasonable tradeoff.
  - New AES uses a 128-bit block size.
- **Key size**:
  - Larger increases security but may decrease en-/de-cryption speed.
  - Greater security achieved by greater resistance to brute-force attacks and greater confusion.
  - Keys 64 bits or less are now inadequate.
  - 128 bits has become a common size.
- **Number of rounds**:
  - Single round offers inadequate security.
  - Multiple rounds offer increasing security (typical size: 16 rounds).
- **Subkey generation algorithm**:
  - Greater complexity means greater resistance to cryptanalysis.
- **Round function** F:
  - Greater complexity means greater resistance to cryptanalysis.

Feistel decryption (16 rounds): details

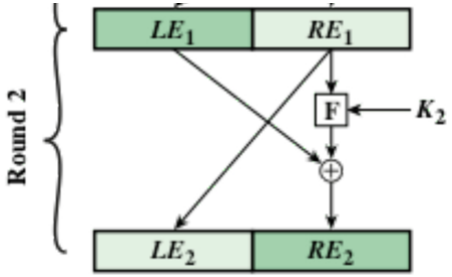


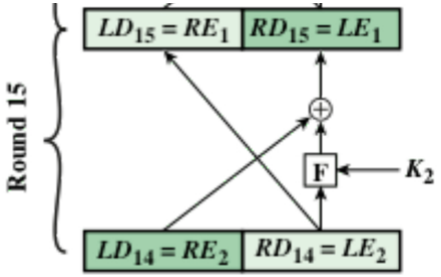
- Decryption essentially same as encryption.
- **Decryption input**:
  - ciphertext
  - subkeys K_i in reverse order (K_n to K_1)
- **Intermediate value** of decryption process

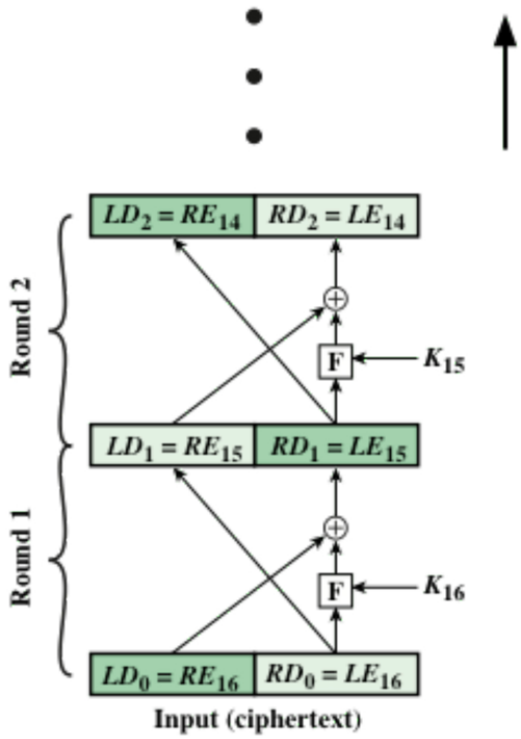
$$LD_{16-i} \parallel RD_{16-i} = RE_i \parallel LE_i$$

equal to corresponding value of encryption process with two halves of value swapped

$$LE_i \parallel RE_i$$

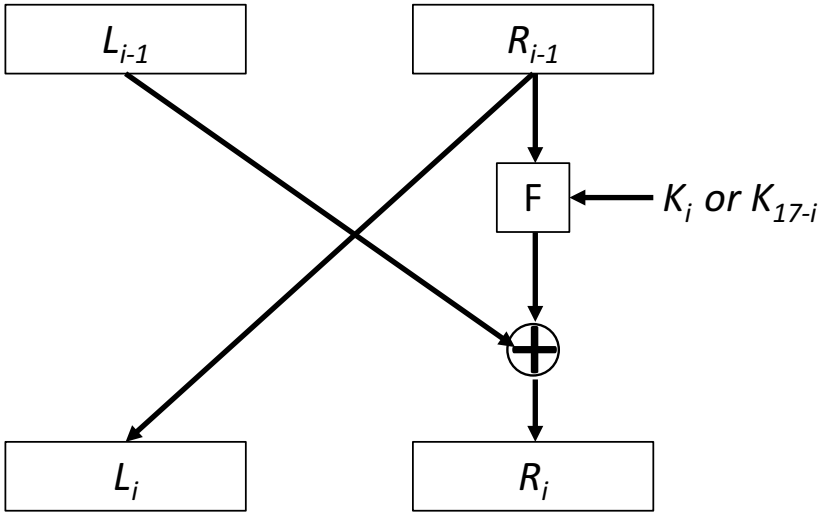






In general: Encryption / Decryption rounds

One of the strengths of the Feistel cypher is that each round of encryption or decryption has the following general shape



where the only difference is that

- at encryption round i, the key K_i is used,
- at decryption round i, the key K_{17-i} is used (for decryption, we take the keys in reverse order).

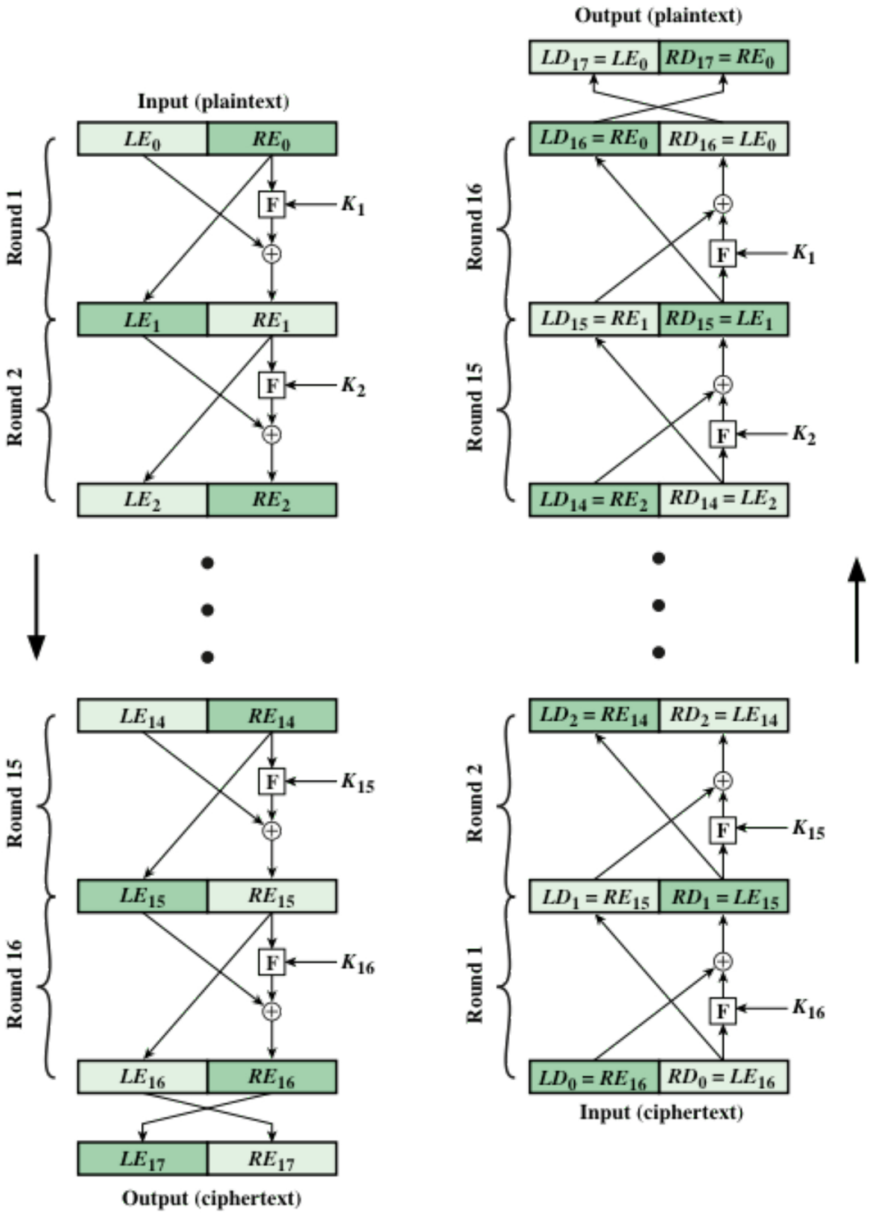
Feistel encryption / decryption (16 rounds): math

- Let us walk through the figure to see the math at work.
- That is, let us take the ciphertext

$$LE_{17} \parallel RE_{17} = RE_{16} \parallel LE_{16}$$

(which is equal to the 32-bit swap of output of round 16) and use it as input to the same algorithm.

- Let's assume that $LD_0 = RE_{16}$ and that $RD_0 = LE_{16}$.
We show that output of decryption round 1 is equal to 32-bit swap of input to encryption round 16.



Show that output of decryption round 1 is equal to 32-bit swap of input to encryption round 16

- Encryption side:

$$\begin{aligned} LE_{16} &= RE_{15} \\ RE_{16} &= LE_{15} \oplus \mathrm{F}(RE_{15}, K_{16}) \end{aligned}$$

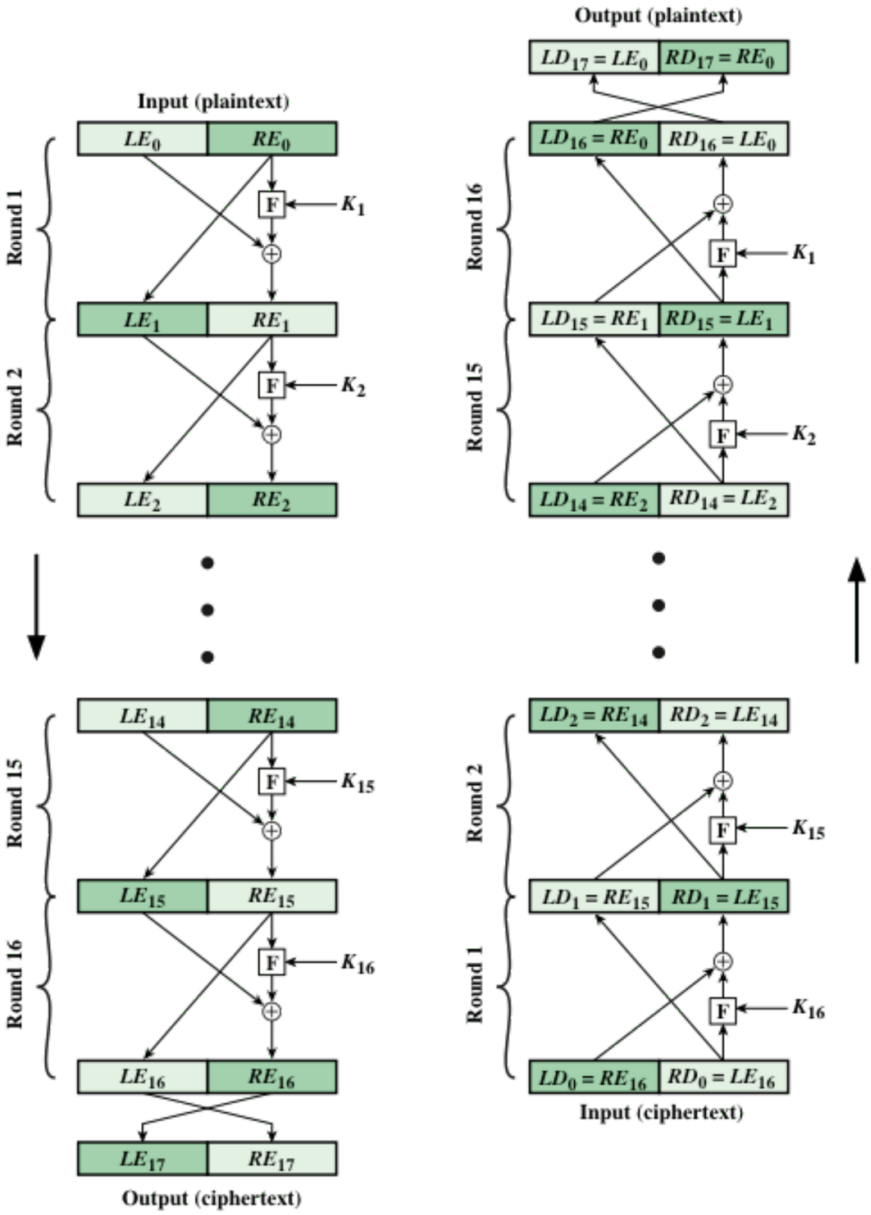
- Decryption side:

$$\begin{aligned} LD_1 &= RD_0 = LE_{16} = RE_{15} \\ RD_1 &= LD_0 \oplus \mathrm{F}(RD_0, K_{16}) \\ &= RE_{16} \oplus \mathrm{F}(RE_{15}, K_{16}) \\ &= (LE_{15} \oplus \mathrm{F}(RE_{15}, K_{16})) \\ &\quad \oplus \mathrm{F}(RE_{15}, K_{16}) \\ &= LE_{15} \end{aligned}$$

Hence:

$$LD_1 \parallel RD_1 = RE_{15} \parallel LE_{15}$$

which is what we wanted to show.



Correspondence holds all the way through the 16 iterations

- Inputs to i^{th} iteration of encryption can be recast as a function of the outputs:

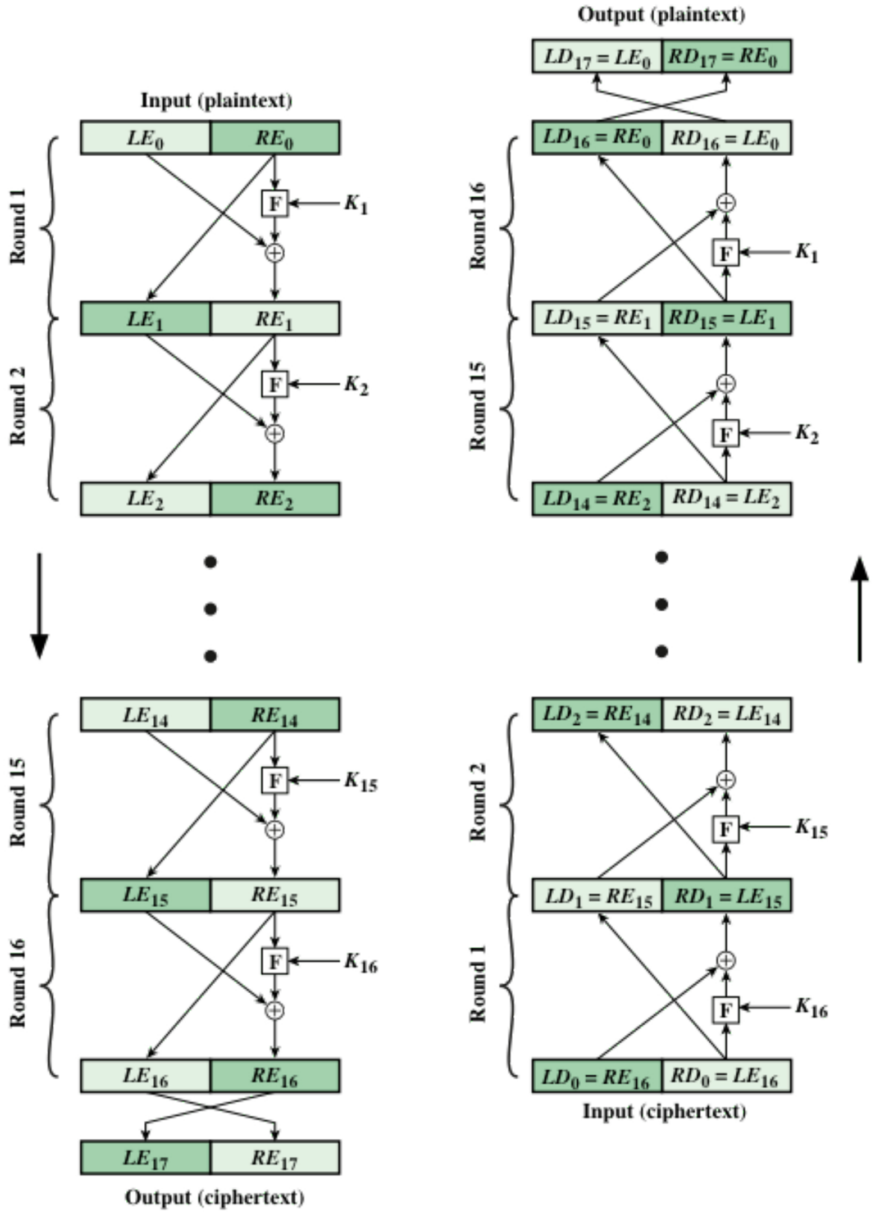
$$\begin{aligned} LE_i &= RE_{i-1} \\ RE_i &= LE_{i-1} \oplus \mathrm{F}(RE_{i-1}, K_i) \end{aligned}$$

Rearranging terms:

$$\begin{aligned} RE_{i-1} &= LE_i \\ LE_{i-1} &= RE_i \oplus \mathrm{F}(RE_{i-1}, K_i) \\ &= RE_i \oplus \mathrm{F}(LE_i, K_i) \end{aligned}$$

- Finally, output of last round of decryption is $RE_0 \,\|\, LE_0$.

  A 32-bit swap recovers the original plaintext.

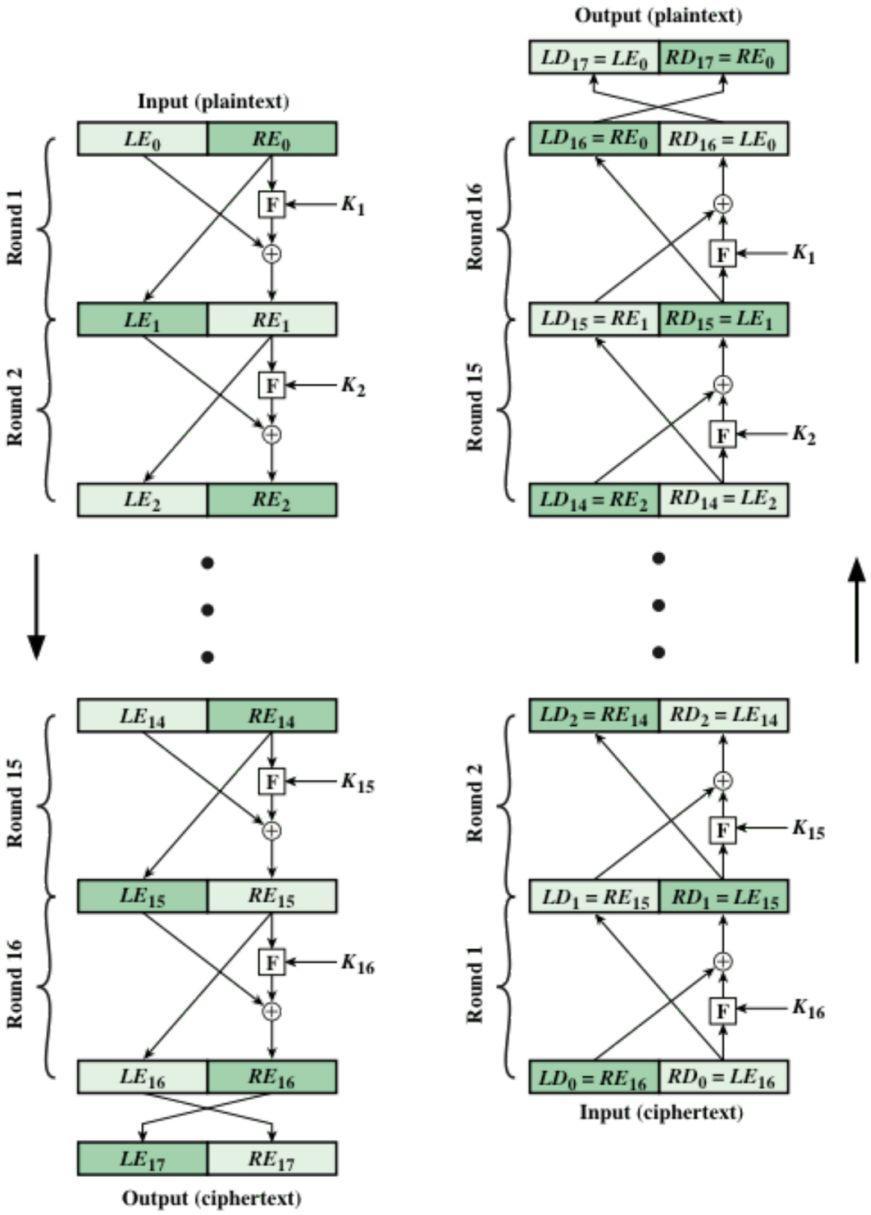


General result

- In general, for $1 \leq i \leq 16$, we can prove that the output of decription round $16 - i$ (i.e., $LD_{16-i} \parallel RD_{16-i}$), is equal to the 32-bit swap of the input to encryption round $i + 1$ (i.e., $LE_i \parallel RE_i$).
- That is, we can prove the equation

$$LD_{16-i} \parallel RD_{16-i} = RE_i \parallel LE_i$$

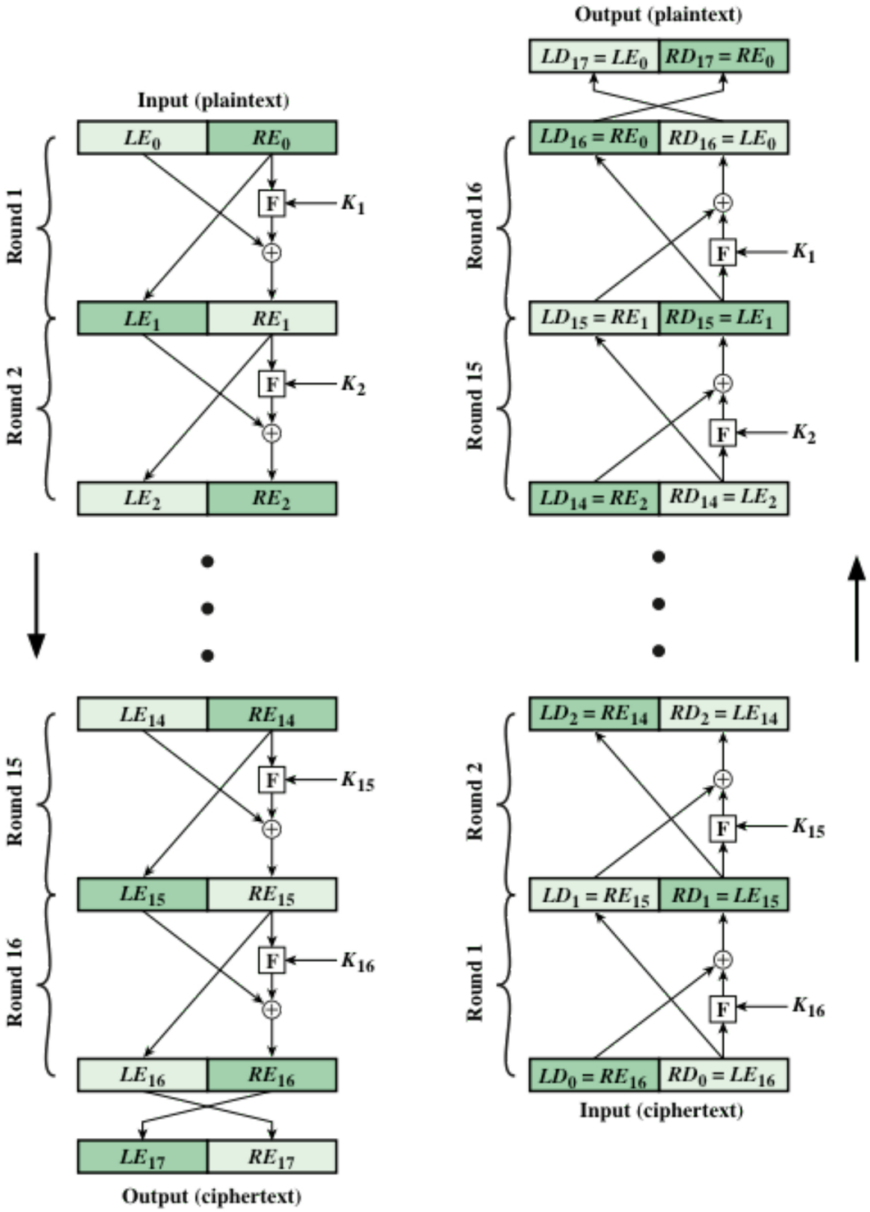
for $1 \leq i \leq 16$



General result

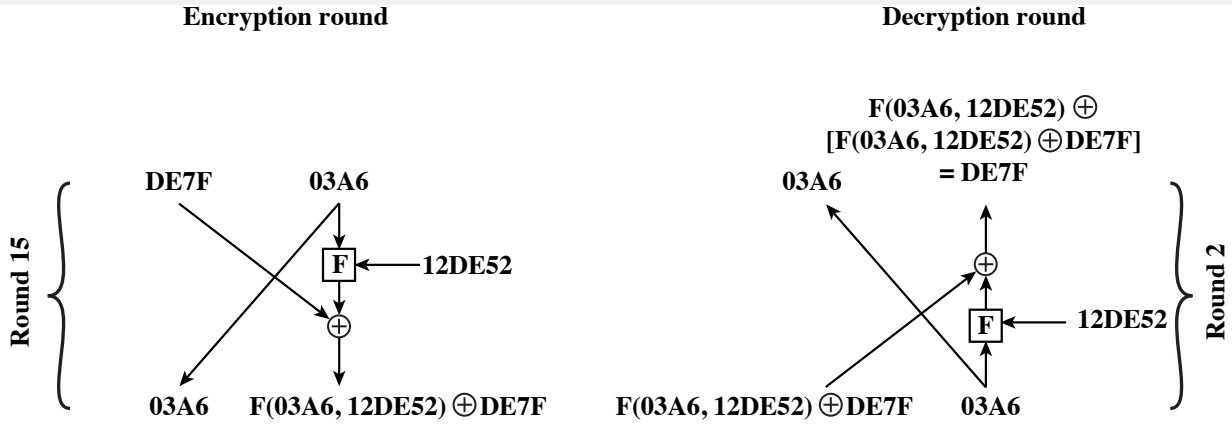
- We carry out the proof by induction, by assuming that we have already proved the equivalence of the output of the decryption round $16 - i - 1 = 15 - i$ is equal to 32-bit swap of the input to encryption round $i + 1 - 1 = i$, i.e., by assuming $LD_{15-i} \parallel RD_{15-i} = RE_{i+1} \parallel LE_{i+1}$,
- and then showing that the output of decryption round $16 - i$ is equal to 32-bit swap of the input to encryption round $i + 1$:

$$
\begin{aligned}
LD_{16-i} &= RD_{16-i-1} = RD_{15-i} = LE_{i+1} = RE_i \\
RD_{16-i} &= LD_{16-i-1} \oplus \mathrm{F}(RD_{16-i-1}, K_{i+1}) \\
&= LD_{15-i} \oplus \mathrm{F}(RD_{15-i}, K_{i+1}) \\
&= RE_{i+1} \oplus \mathrm{F}(RE_i, K_{i+1}) \\
&= (LE_{i+1-1} \oplus \mathrm{F}(RE_{i+1-1}, K_{i+1})) \\
&\quad \oplus \mathrm{F}(RE_i, K_{i+1}) \\
&= LE_i
\end{aligned}
$$



Feistel cipher: a specific example

- Consider encryption round 15, corresponding to decryption round 2.



- Suppose that
  - 32-bit blocks (two 16-bit halves)
  - 24-bit key
  - at end of encryption round 14, value of intermediate block (in hexadecimal) is DE7F03A6 ($LE_{14} =$ DE7F and $RE_{14} =$ 03A6)
  - value of K_{15} is 12DE52
- Then $LE_{15} =$ 03A6 $= RD_1$ and $RE_{15} = \mathrm{F}(\mathrm{03A6}, \mathrm{12DE52}) \oplus$ DE7F $= LD_1$.
- We prove that $LD_2 = RE_{14} =$ 03A6 and $RD_2 = LE_{14} =$ DE7F:

  $LD_2 = RD_1 =$ 03A6

  $RD_2 = LD_1 \oplus \mathrm{F}(\mathrm{03A6}, \mathrm{12DE52}) = (\mathrm{F}(\mathrm{03A6}, \mathrm{12DE52}) \oplus \mathrm{DE7F}) \oplus \mathrm{F}(\mathrm{03A6}, \mathrm{12DE52}) = \mathrm{DE7F}$

Table of contents I

Table of contents II

Table of contents III

Table of contents I

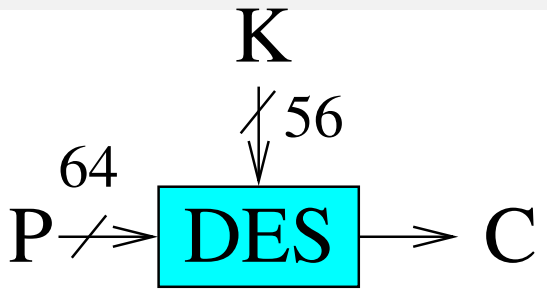
Table of contents II

Table of contents III

A little bit of history: LUCIFER, DES, triple DES

- **1971**: IBM project, led by Horst Feistel, develops **LUCIFER algorithm**:
  - A Feistel block cipher with **64-bit blocks** and **128-bit key**.
  - Sold to Lloyd's of London for use in a cash-dispensing system, also developed by IBM.
  - **Refined version** (effort lead by Walter Tuchman and Carl Meyer of IBM, with support of National Security Agency NSA): more resistant to cryptanalysis but **56-bit key** in order to fit on a single chip.
- **1973**: National Bureau of Standards (NBS) issues a request for proposals for a national cipher standard.
  - IBM's refined version of LUCIFER by far best algorithm proposed.
- **1977**: adopted as **Data Encryption Standard** (Federal Information Processing Standard 46 (FIPS PUB 46)) by National Bureau of Standards (now National Institute of Standards and Technology NIST).
- **1994**: NIST reaffirms DES for federal use for another 5 years, recommends use for applications other than protection of classified info.
- **1999**: NIST issues a new version (FIPS PUB 46-3), indicates DES should be used only for legacy systems and **triple DES** be used.

DES design controversy



- Although DES standard is public,
  there was considerable controversy over design:
  - in choice of 56-bit key (vs Lucifer 128-bit),
  - because design criteria were classified.
- Subsequent events and public analysis show in fact design was appropriate.
- Use of DES has flourished.
  - Especially in financial applications.
  - Still standardized for legacy application use.
  - Extensions (triple DES) to overcome short key length.

DES design controversy

People have long questioned the security of DES. There has been much speculation on the key length, number of iterations, and design of the S-boxes. The S-boxes were particularly mysterious — all those constants, without any apparent reason as to why or what they're for. Although IBM claimed that the inner workings were the result of 17 man-years of intensive cryptanalysis, some people feared that the NSA embedded a trapdoor into the algorithm so they would have an easy means of decrypting messages.

— Bruce Schneier, Applied Cryptography p.278.

The National Security Agency also provided technical advice to IBM. And Konheim has been quoted as saying "we sent the S-boxes off to Washington. They came back and were all different. We ran our tests and they passed." People have pointed to this as evidence that the NSA put a trapdoor in DES.

— Bruce Schneier, Applied Cryptography p.279.

Table of contents I

Table of contents II

Table of contents III

DES encryption: overall scheme

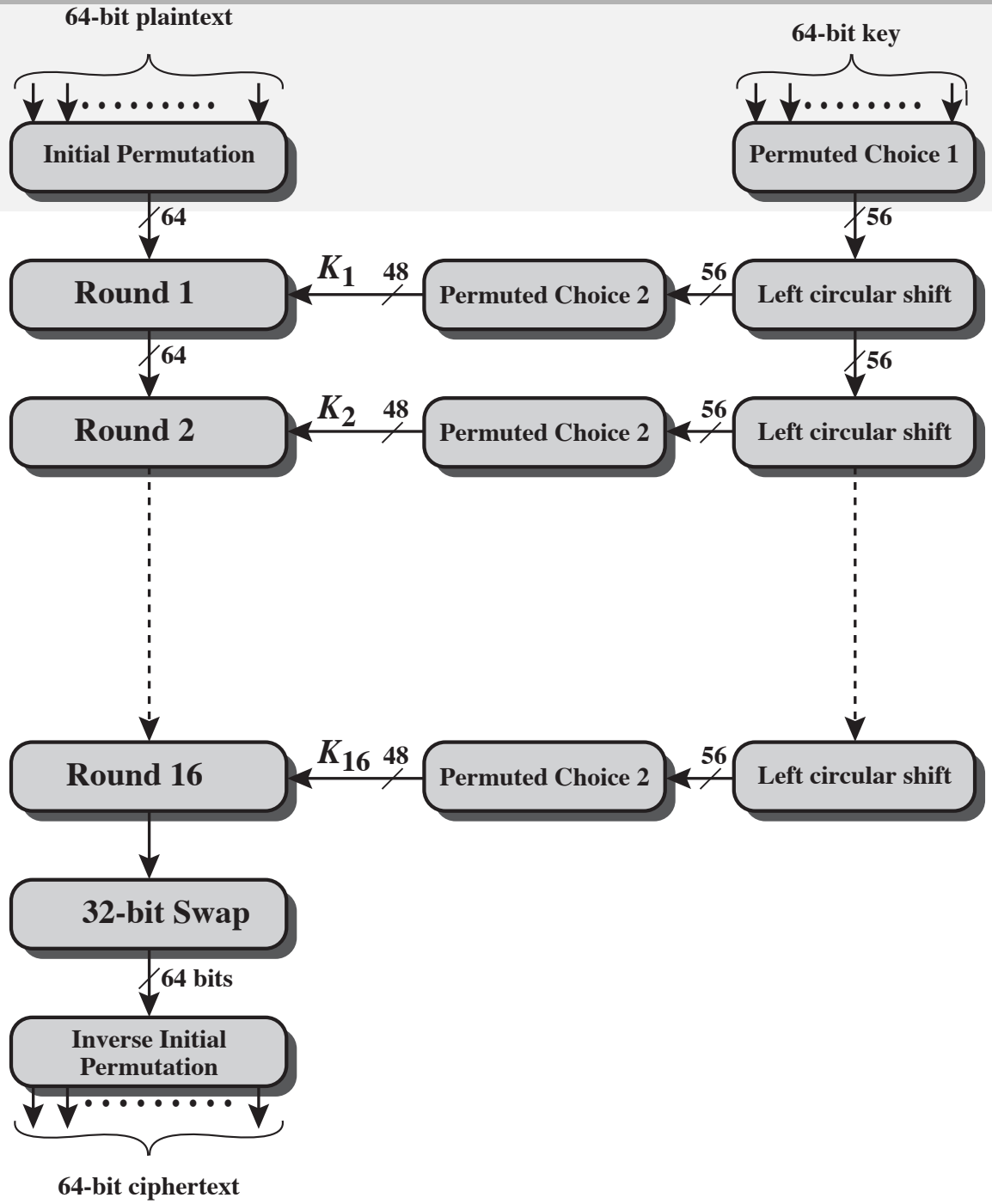
Block size: 64 bits

- Plaintext and ciphertext space
  $\mathcal{M} = \mathcal{C} = \{0,1\}^{64}$

- Input:
  - Plaintext: 64 bits
  - Key: 56 bits

  Actually: 64-bit key as input, but only 56 are used.

- Output: 64 bits



DES keys are all bitstrings of length 64 with the following property

If a 64-bit DES key is divided into eight bytes,
then the sum of the eight bits of each byte is odd.

- This means that 7 of the 8 bits determine the value of the 8^{th} bit. And that transmission errors of one bit can be spotted.
- Therefore the key space is

$$\mathcal{K} = \{(b_1, \ldots, b_{64}) \in \{0,1\}^{64} \mid \sum_{i=1}^{8} b_{8k+i} \equiv 1 \mod 2,\ 0 \leq k \leq 7\}$$

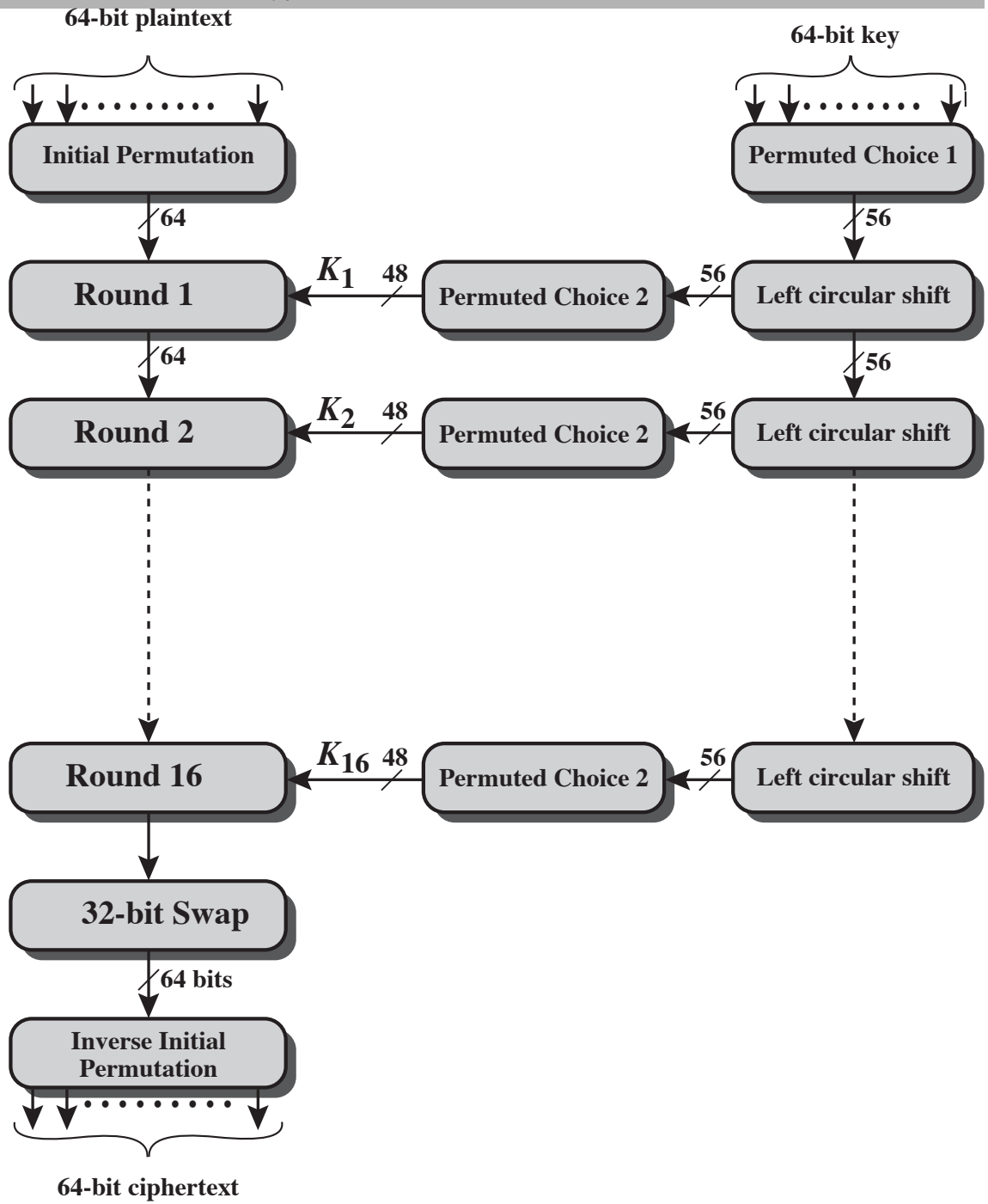
- Example: a valid hexadecimal DES key is 133457799BBCDFF1, whose binary expansion is

0	0	0	1	0	0	1	1
0	0	1	1	0	1	0	0
0	1	0	1	0	1	1	1
0	1	1	1	1	0	0	1
1	0	0	1	1	0	1	1
1	0	1	1	1	1	0	0
1	1	0	1	1	1	1	1
1	1	1	1	0	0	0	1

Plaintext processed in 3 phases:

1. **Initial permutation (IP)** rearranges bits to produce **permuted input**.
2. **16 rounds of both permutation and substitution functions**.
   - Output of 16^{th} round: 64 bits, function of input plaintext and key.
   - Left and right halves of output swapped to produce **preoutput**.
3. Preoutput is passed through $\mathbf{IP}^{-1}$ to finally produce the **64-bit ciphertext**.

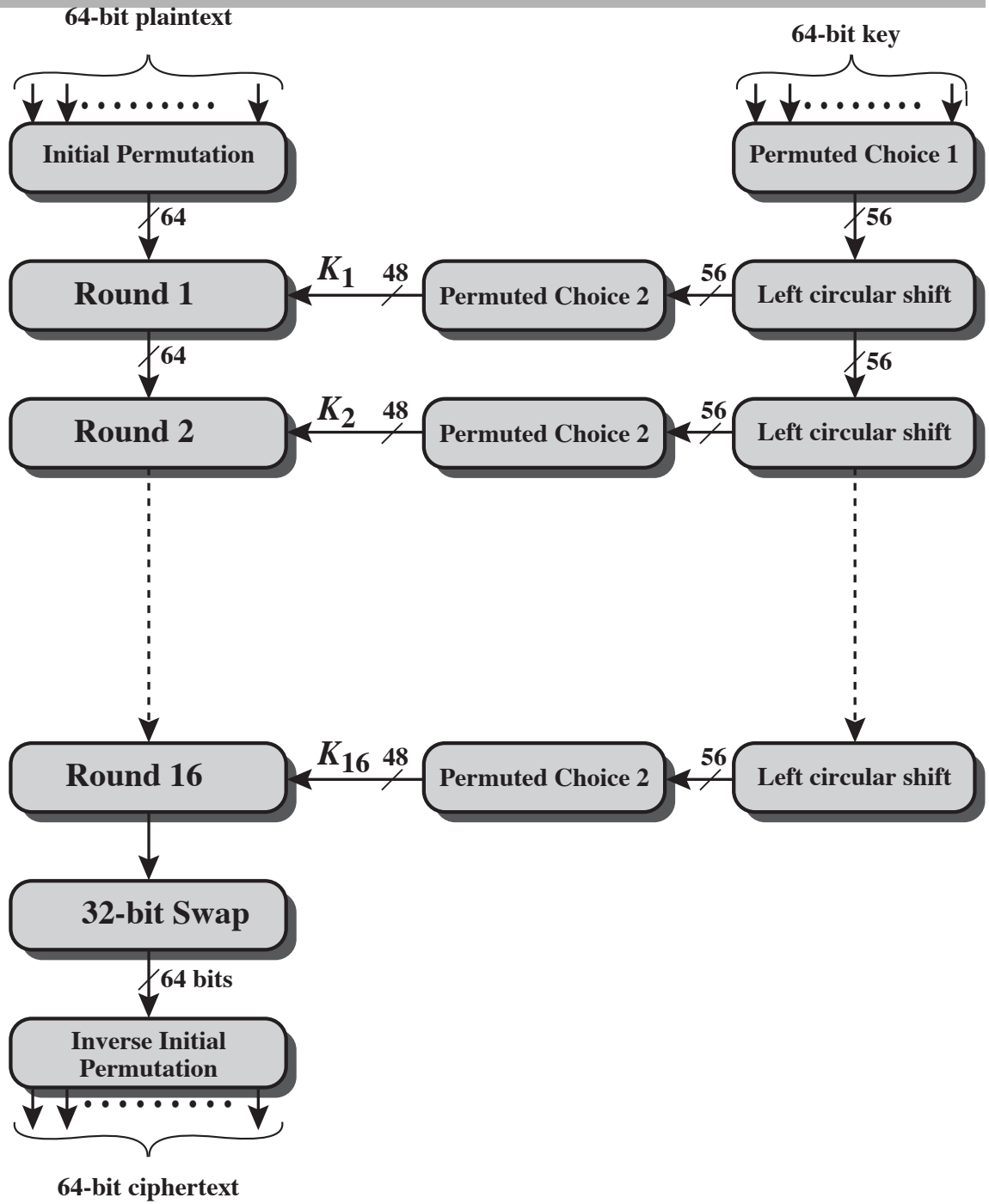
Except for initial and final permutations, DES has exact structure of a Feistel cipher.



Note: IP and IP^{-1} have no real cryptographic significance (included to facilitate loading blocks in and out of mid-1970s 8-bit based hardware).

Use of 56-bit key:

- Initially, key is passed through a **permutation** function.
- For each of 16 rounds, a **subkey** K_i is produced by combination of a **left circular shift** and a **permutation**.
  - Permutation function is the same for each round.
  - A different subkey is produced because of repeated shifts of key bits.



IP and IP^{-1} defined by tables:

- Input to a table consists of 64 bits numbered from 1 to 64.
- 64 entries in permutation table contain a permutation of numbers from 1 to 64.
- Each entry in permutation table indicates position of a numbered input bit in output, which also consists of 64 bits.
- **Even bits to LH half, odd bits to RH half**.
- Quite regular in structure (easy in HW).

(a) Initial Permutation (IP)

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
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

(b) Inverse Initial Permutation (IP^{-1})

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

Example:

- Consider 64-bit input M (left) and permutation $\mathrm{IP}(M)$ (right).

$$\begin{array}{cccccccc} M_1 & M_2 & M_3 & M_4 & M_5 & M_6 & M_7 & M_8 \\ M_9 & M_{10} & M_{11} & M_{12} & M_{13} & M_{14} & M_{15} & M_{16} \\ M_{17} & M_{18} & M_{19} & M_{20} & M_{21} & M_{22} & M_{23} & M_{24} \\ M_{25} & M_{26} & M_{27} & M_{28} & M_{29} & M_{30} & M_{31} & M_{32} \\ M_{33} & M_{34} & M_{35} & M_{36} & M_{37} & M_{38} & M_{39} & M_{40} \\ M_{41} & M_{42} & M_{43} & M_{44} & M_{45} & M_{46} & M_{47} & M_{48} \\ M_{49} & M_{50} & M_{51} & M_{52} & M_{53} & M_{54} & M_{55} & M_{56} \\ M_{57} & M_{58} & M_{59} & M_{60} & M_{61} & M_{62} & M_{63} & M_{64} \end{array}$$

$$\begin{array}{cccccccc} M_{58} & M_{50} & M_{42} & M_{34} & M_{26} & M_{18} & M_{10} & M_2 \\ M_{60} & M_{52} & M_{44} & M_{36} & M_{28} & M_{20} & M_{12} & M_4 \\ M_{62} & M_{54} & M_{46} & M_{38} & M_{30} & M_{22} & M_{14} & M_6 \\ M_{64} & M_{56} & M_{48} & M_{40} & M_{32} & M_{24} & M_{16} & M_8 \\ M_{57} & M_{49} & M_{41} & M_{33} & M_{25} & M_{17} & M_9 & M_1 \\ M_{59} & M_{51} & M_{43} & M_{35} & M_{27} & M_{19} & M_{11} & M_3 \\ M_{61} & M_{53} & M_{45} & M_{37} & M_{29} & M_{21} & M_{13} & M_5 \\ M_{63} & M_{55} & M_{47} & M_{39} & M_{31} & M_{23} & M_{15} & M_7 \end{array}$$

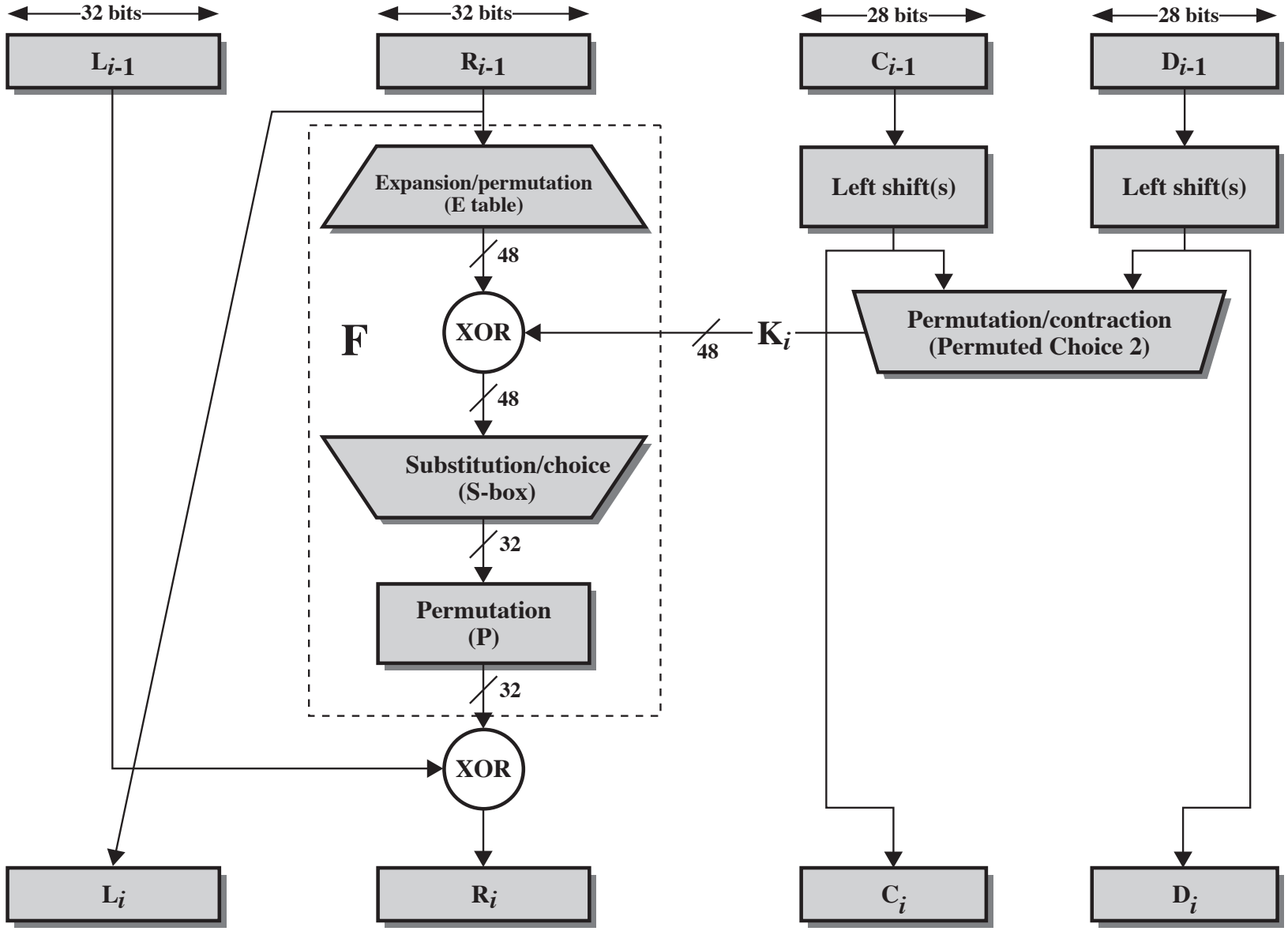
- $\mathrm{IP}^{-1}(\mathrm{IP}(M))$ restores original ordering of bits.

Table of contents I

Table of contents II

Table of contents III

DES encryption: details of single round

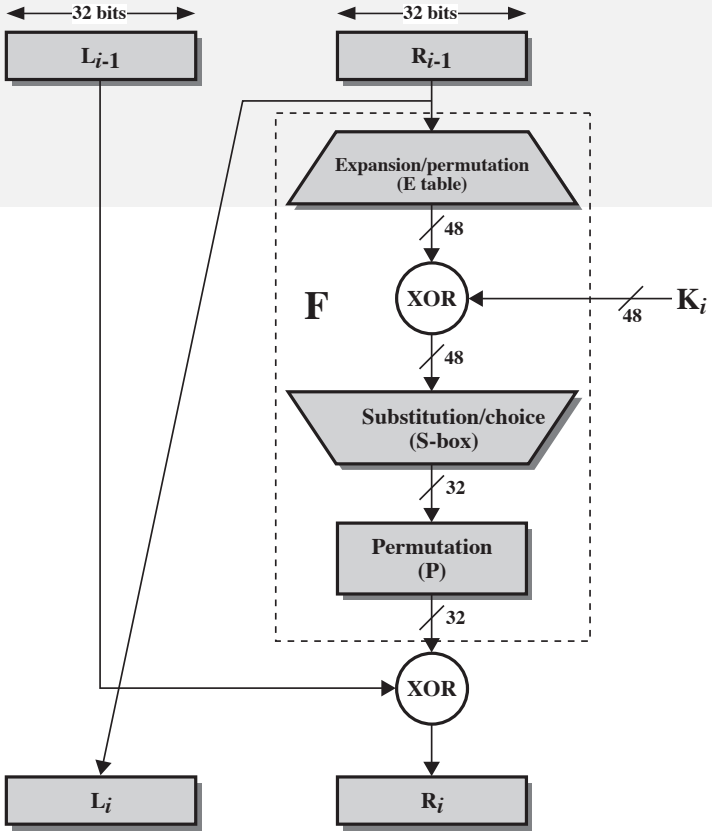


DES encryption: details of single round

- L and R halves of each 64-bit intermediate value treated as separate 32-bit quantities.
- As in any classic Feistel cipher, overall processing at each round is:

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

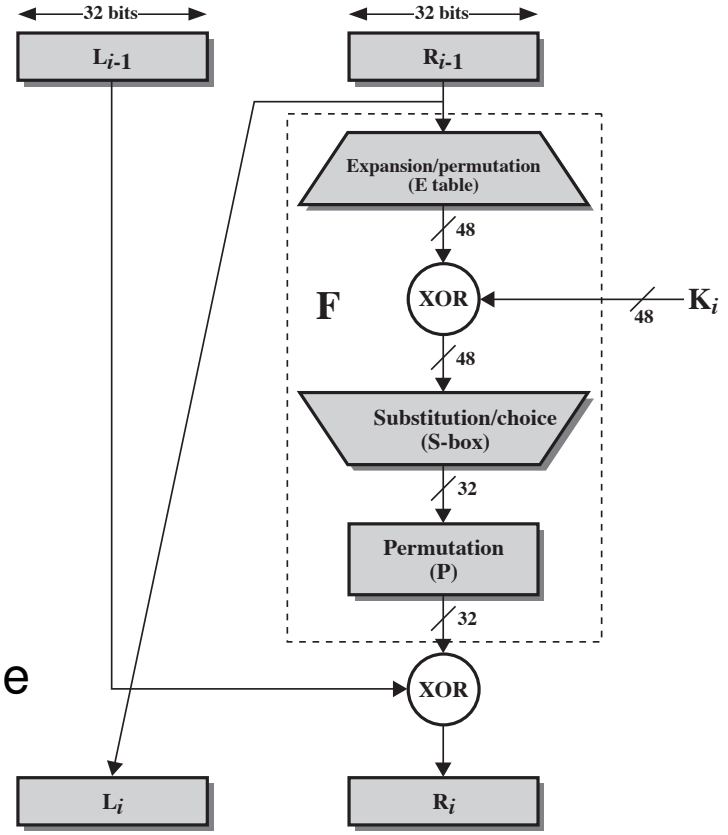
- Round key K_i is 48 bits.
- R_{i-1} is 32 bits.
  - Is expanded to 48 bits by using **Expansion Permutation (E)** table, which defines a permutation plus an expansion that involves duplication of 16 bits of R_{i-1}.
  - Resulting 48 bits are XORed with K_i.



(c) Expansion Permutation (E)

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

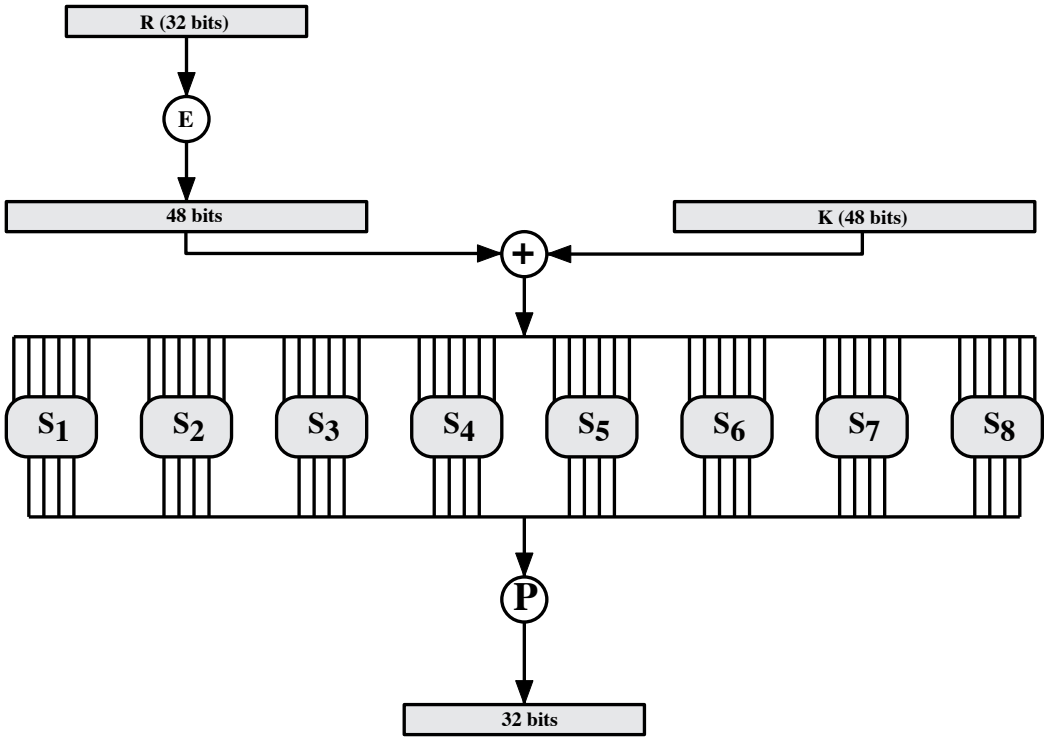
- 48-bit result of XOR passes through a substitution function (**S-box**) that produces a 32-bit output, which is permuted by using a table (**Permutation Function P**).

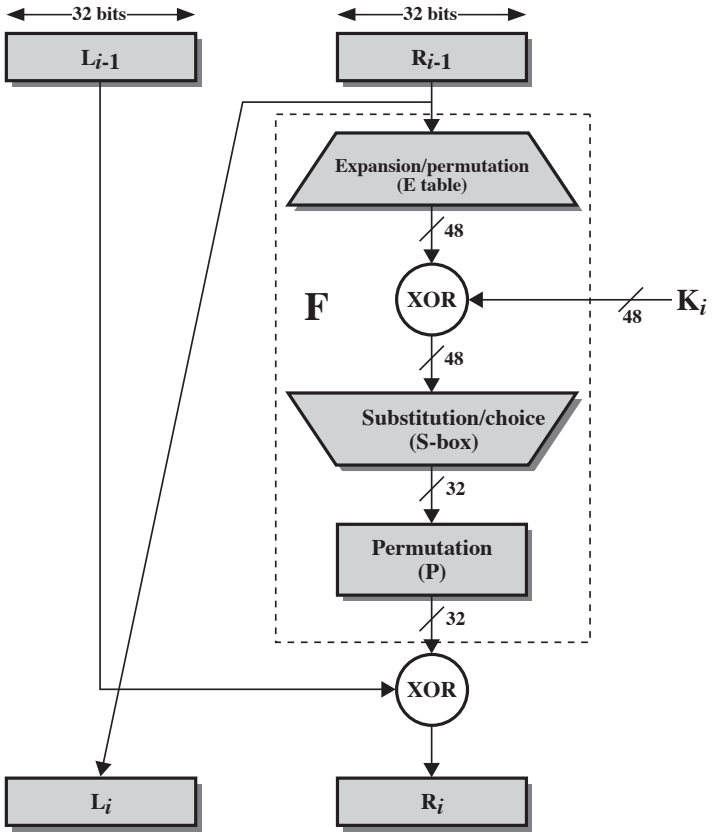


(d) Permutation Function (P)

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

- Role of S-boxes in F illustrated in this figure





- Substitution consists of a set of 8 S-boxes, each of which accepts 6 bits as input and produces 4 bits as output.
- These transformations are defined in table on following slide.

- First and last bits of the input to box S_i form a 2-bit binary number to select 1 of 4 substitutions defined by four rows in the table for S_i.
- Middle 4 bits select one of the 16 columns.
- Decimal value in the cell selected by the row and column is then converted to its 4-bit representation to produce output.
- Example: in S_1, for input 011001, the row is 01 (row 1) and the column is 1100 (column 12).

  The value in row 1, column 12 is 9, so the output is 1001.
- Each row of an S-box defines a general reversible substitution.

S_1

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

S_2

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

S_3

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

S_4

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

S_5

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

S_6

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

S_7

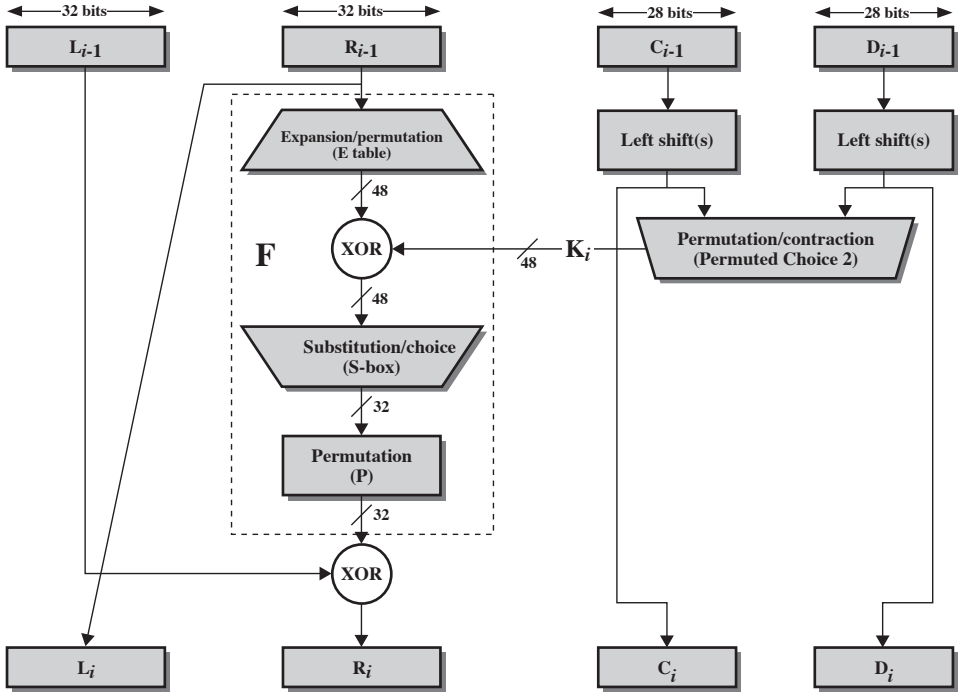
4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

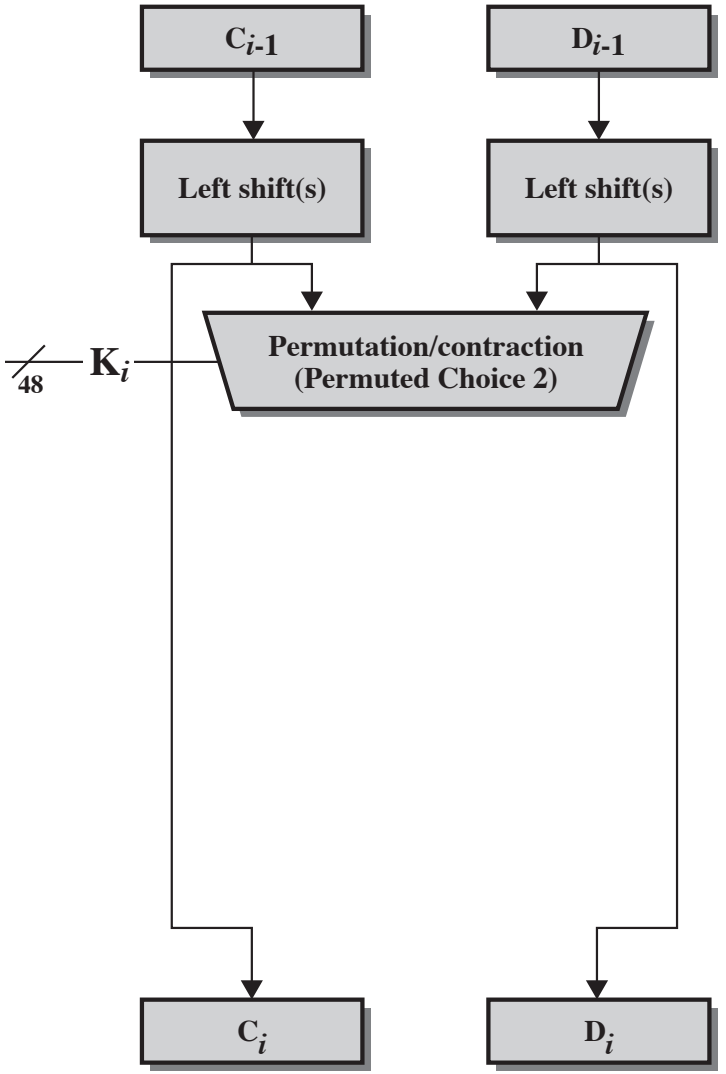
S_8

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

Key generation:

- 64-bit key used as input (bits 1..64).





- Every eighth bit is ignored, as indicated by lack of shading in table (a).
- Key first subjected to a permutation governed by table (b) **Permuted Choice One**.
- Resulting 56-bit key then treated as two 28-bit quantities C_0 and D_0.
- At each round, C_{i-1} and D_{i-1} separately subjected to a **circular left shift** (rotation) of 1 or 2 bits, as governed by table (d)
- These shifted values serve as input to
  - next round, and
  - **Permuted Choice Two** (table (c)), which produces a 48-bit output that serves as input to function $\mathrm{F}(R_{i-1}, K_i)$.

(a) Input Key

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

(b) Permuted Choice One (PC-1)

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

(c) Permuted Choice Two (PC-2)

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

(d) Schedule of Left Shifts

Round Number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Bits Rotated	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Table of contents I

Table of contents II

Table of contents III

DES decryption

As with any Feistel cipher, decryption uses the same algorithm as encryption, except that the application of the subkeys is reversed.

Table of contents I

Table of contents II

Table of contents III

DES: an example

We now work through an example in (almost full) detail.

Although you are not expected to duplicate the example by hand, you will find it informative.

DES: an example

- Consider $P = 0123456789ABCDEF$, whose binary expansion is

0	0	0	0	0	0	0	1
0	0	1	0	0	0	1	1
0	1	0	0	0	1	0	1
0	1	1	0	0	1	1	1
1	0	0	0	1	0	0	1
1	0	1	0	1	0	1	1
1	1	0	0	1	1	0	1
1	1	1	0	1	1	1	1

- Application of IP yields:

1	1	0	0	1	1	0	0
0	0	0	0	0	0	0	0
1	1	0	0	1	1	0	0
1	1	1	1	1	1	1	1
1	1	1	1	0	0	0	0
1	0	1	0	1	0	1	0
1	1	1	1	0	0	0	0
1	0	1	0	1	0	1	0

(a) Initial Permutation (IP)

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
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

- So we obtain

$$L_0 = 11001100000000001100110011111111$$
$$R_0 = 11110000101010101111000010101010$$

- We use key 133457799BBCDFF1, whose binary expansion is

0	0	0	1	0	0	1	1
0	0	1	1	0	1	0	0
0	1	0	1	0	1	1	1
0	1	1	1	1	0	0	1
1	0	0	1	1	0	1	1
1	0	1	1	1	1	0	0
1	1	0	1	1	1	1	1
1	1	1	1	0	0	0	1

- We compute the first round key. First every 8^{th} bit is ignored (as in table (a)):

(a) Input Key

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

0	0	0	1	0	0	1
0	0	1	1	0	1	0
0	1	0	1	0	1	1
0	1	1	1	1	0	0
1	0	0	1	1	0	1
1	0	1	1	1	1	0
1	1	0	1	1	1	1
1	1	1	1	0	0	0

- then a permutation governed by Permuted Choice One (table (b)):

(b) Permuted Choice One (PC-1)

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

1	1	1	1	0	0	0
0	1	1	0	0	1	1
0	0	1	0	1	0	1
0	1	0	1	1	1	1
0	1	0	1	0	1	0
1	0	1	1	0	0	1
1	0	0	1	1	1	1
0	0	0	1	1	1	1

- Resulting 56-bit key then treated as two 28-bit quantities C_0 and D_0.

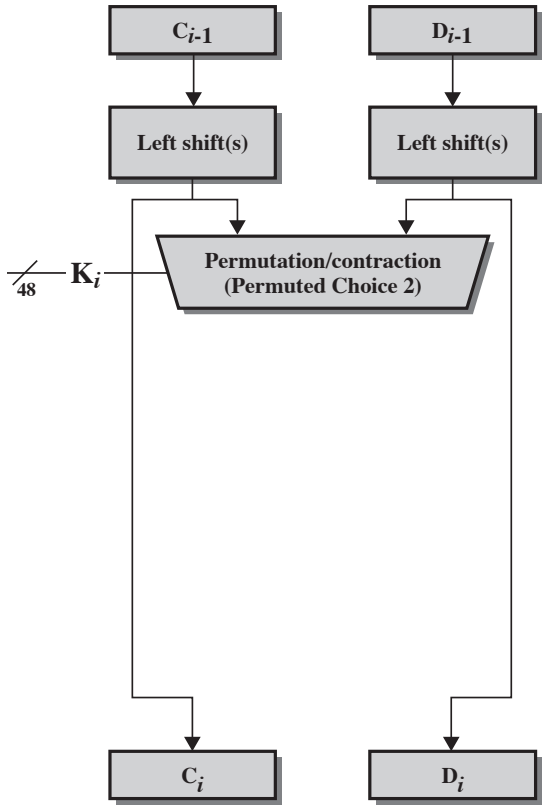
- $C_0 = 11110000110011001010101111$
  $D_0 = 01010101011001100111100011111$
- At each round, C_{i-1} and D_{i-1} separately subjected to a circular left shift (rotation) of 1 or 2 bits, as governed by table (d)
- For instance, C_1 and D_1 are obtained by a left shift 1:
  $C_1 = 11100001100110010101011111$
  $D_1 = 10101010110011001111100011110$
- These shifted values serve as input to
  - next round, and
  - Permuted Choice Two (table (c)), which produces a 48-bit output that serves as input to function $\mathrm{F}(R_{i-1}, K_i)$.
- From C_1 and D_1, by table (c), we obtain
  $K_1 =$
  000110110000001011101111111111000111000001110010

(c) Permuted Choice Two (PC-2)

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

(d) Schedule of Left Shifts

Round Number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Bits Rotated	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1



- Using

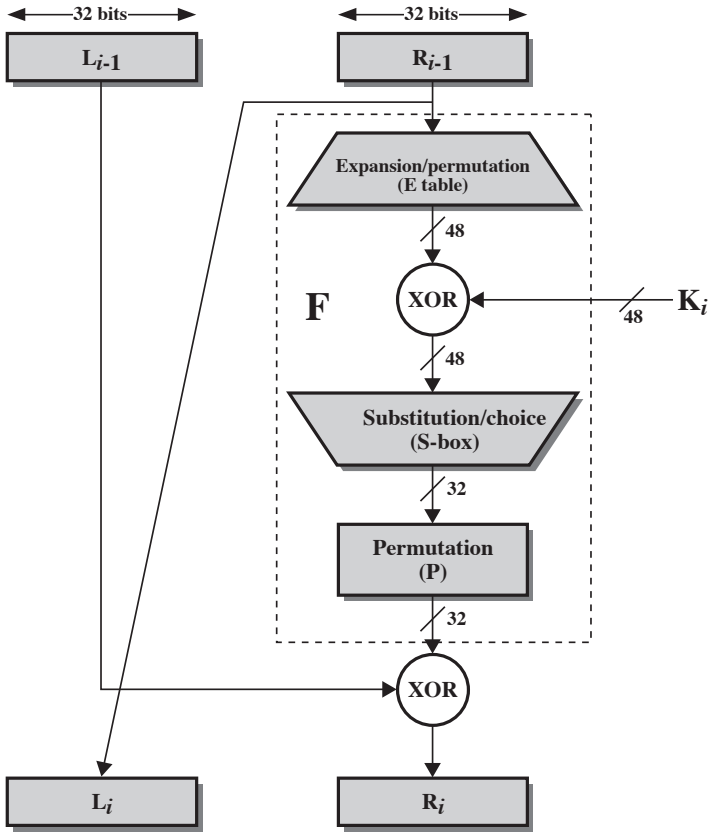
$$\begin{aligned} R_0 &= 11110000101010101111000010101010 \\ K_1 &= 00011011000000101110111111111100 \\ &\quad 011100000111010 \end{aligned}$$

and the expansion permutation E we obtain

$$\begin{aligned} \mathrm{E}(R_0) \oplus K_1 &= 01100001000010111101110101000 \\ &\quad 0011001100101001001111 \end{aligned}$$

which, given in input to S-box and permutation P, yields

$$\mathrm{F}(R_0, K_1) = 00100011010010101010100110111011$$



- Finally, an XOR with

$$L_0 = 11001100000000001100110011111111$$

yields

$$R_1 = 11101111010010100110010101000100$$

- The other rounds are computed analogously.

(c) Expansion Permutation (E)

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

Table of contents I

Table of contents II

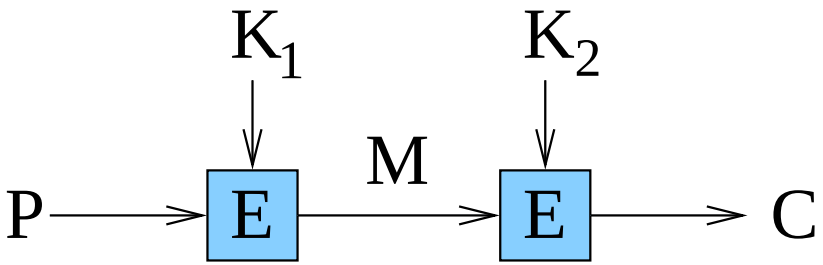
Table of contents III

Security of DES

- Since its invention, security of DES has been studied very intensively.
- Special techniques such as **differential cryptanalysis** and **linear cryptanalysis** have been invented to attack DES (see the literature, e.g., how to reduce key space from 2^{56} to 2^{43} using linear cryptanalysis),
- but the most successful attack has been an **exhaustive search of the key space**.
  - With special hardware or large networks of workstations, it is now possible to decrypt DES ciphertexts in a few days or even a few hours.
  - 7 hours with 1 million dollar computer in 1993.
- Soon DES will be breakable on a single PC.

Increasing DES Security: Double DES (a.k.a. 2DES)

- **Idea**: Perform two DES encryptions



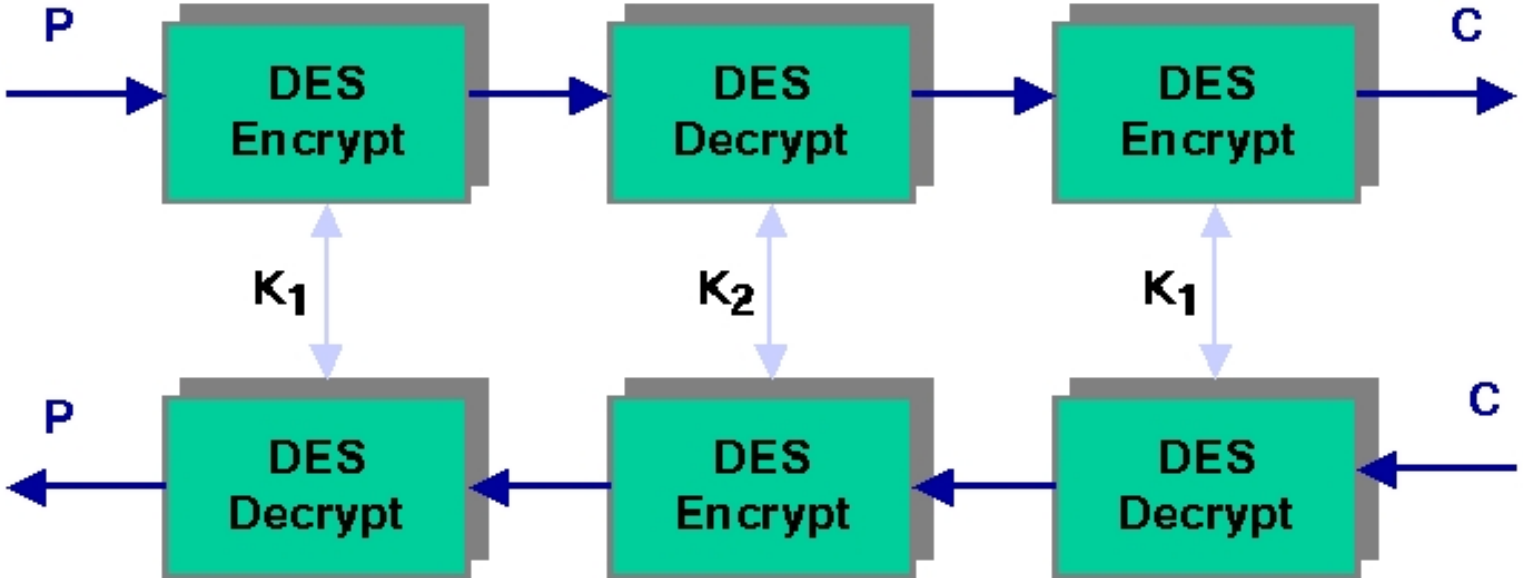
Equivalent to

112 bit keys?

- **Attack**: Meet-in-the-Middle
  - For $C = E(K_2, E(K_1, P))$ let $X = E(K_1, P) = D(K_2, C)$.
  - Given known P and C encrypt P for all 2^{56} possible K_1.
  - Store in table, sorted by X.
  - Decrypt C with all 2^{56} possible K_2 and look for a match.
  - Each hit is a candidate solution. Validate with additional plain/cipher-text pair.
  - A known plaintext attack against double DES (112 bit keys) will succeed with effort on the order of 2^{56} operations.

Increasing DES Security: Triple DES (a.k.a. 3DES)

- **Use 3 stages of encryption instead of 2, with 2 keys K_1, K_2:**



- **Compatibility** is maintained with standard DES ($K_2 = K_1$).
- No known practical attack
  $\Rightarrow$ brute-force search with 2^{112} operations.

- **Triple DES with 3 different keys K_1, K_2, K_3 exists** as well:
  - Effective key length of 168 bits.
  - Adopted by some Internet applications (e.g., PGP and S/MIME).

Table of contents I

Table of contents II

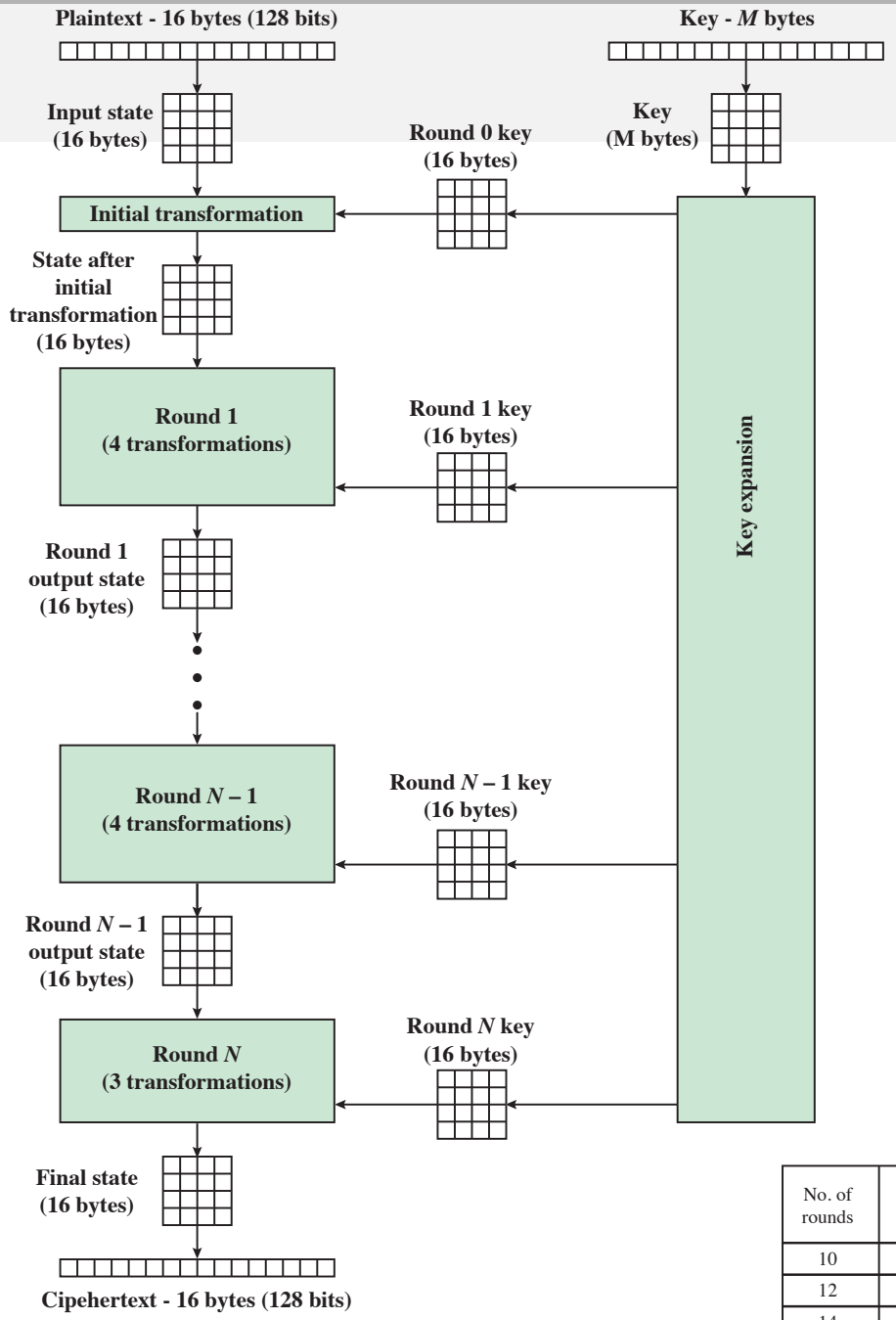
AES: Advanced Encryption Standard

AES

- A **symmetric block cipher** intended to replace DES for commercial applications.
  - **1997**: NIST initiated selection process for the successor of DES.
  - One of the submissions was the **Rijndael cipher** (named after its inventors Rijmen and Daemen).
  - **2001**: this encryption scheme standardized as the Advanced Encryption Standard.
  - AES special case of Rijndael, which admits more different block lengths and ciphertext spaces.
- AES uses a **128-bit block size** and a **key size of 128, 192, or 256 bits**.
- **AES does not use a Feistel structure**.
  Instead, each full round consists of **4 separate functions**: *byte substitution*, *permutation*, *arithmetic operations over a finite field*, and *XOR with a key*.

AES Encryption

- Plaintext block size of 128 bits, or 16 bytes.
- Key length can be 16, 24, or 32 bytes (128, 192, or 256 bits).
- Algorithm referred to as AES-128, AES-192, or AES-256, depending on key length.
- Cipher consists of *N* rounds, depending on key length:
  - 10 rounds for a 16-byte key
  - 12 rounds for a 24-byte key
  - 14 rounds for a 32-byte key
- For further details and examples, see references.



No. of rounds	Key Length (bytes)
10	16
12	24
14	32

Table of contents I

Table of contents II

Block cipher modes of operation

- A block cipher takes a fixed-length block of text of length b bits and a key as input and produces a b-bit block of ciphertext.
  - E.g., DES encrypts 64-bit blocks with 56-bit key.
- How is a block cipher used when messages exceed block-width?
  - If amount of plaintext to be encrypted is greater than b bits, then block cipher can still be used by breaking plaintext.
- NIST defined 5 modes of operation, intended for use with any symmetric block cipher, including triple DES and AES.

Mode of operation

A technique for enhancing the effect of a cryptographic algorithm or adapting the algorithm for an application, such as applying a block cipher to a sequence of data blocks or a data stream.

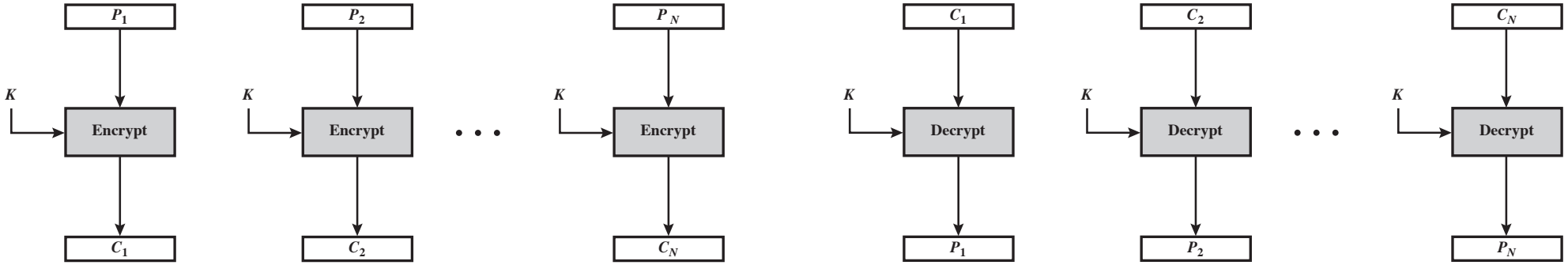
Mode	Description	Typical Application
Electronic Codebook (ECB)	Each block of 64 plaintext bits is encoded independently using the same key.	•Secure transmission of single values (e.g., an encryption key)
Cipher Block Chaining (CBC)	The input to the encryption algorithm is the XOR of the next 64 bits of plaintext and the preceding 64 bits of ciphertext.	•General-purpose block-oriented transmission •Authentication
Cipher Feedback (CFB)	Input is processed *s* bits at a time. Preceding ciphertext is used as input to the encryption algorithm to produce pseudorandom output, which is XORed with plaintext to produce next unit of ciphertext.	•General-purpose stream-oriented transmission •Authentication
Output Feedback (OFB)	Similar to CFB, except that the input to the encryption algorithm is the preceding encryption output, and full blocks are used.	•Stream-oriented transmission over noisy channel (e.g., satellite communication)
Counter (CTR)	Each block of plaintext is XORed with an encrypted counter. The counter is incremented for each subsequent block.	•General-purpose block-oriented transmission •Useful for high-speed requirements

We will focus on 4 modes: ECB, CBC, CFB and OFB.

Modes of operation: Electronic Codebook (ECB)

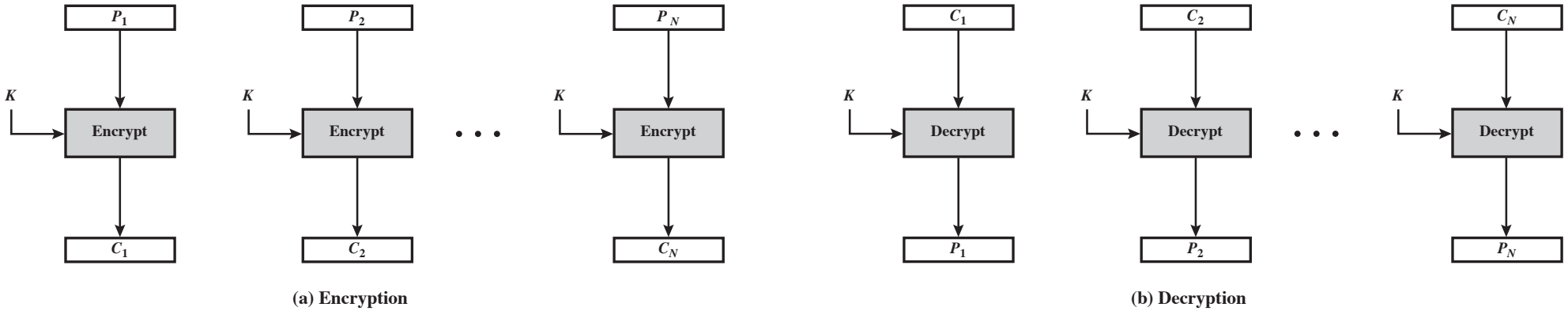
Electronic Codebook Mode (ECB)

- Simplest mode.
- Plaintext message broken into N independent blocks $P_1, \ldots, P_N$.
  - **Each plaintext block P_i (with $1 \leq i \leq N$) encrypted individually**, independently of other blocks: $C_i = E(K, P_i)$.
  - Requires last block be padded to a full b bits if it is a partial block.
  - Each P_i is a value that is substituted, like a codebook, hence name.
- **Each ciphertext block is decrypted also individually**, with same K: $P_i = D(K, C_i)$.



(a) Encryption (b) Decryption

ECB: limitations and ideal use



- For lengthy messages, ECB mode may not be secure:
  - If the same *b*-bit block of plaintext appears more than once in the message, it always produces the same ciphertext.
  - If the message is highly structured, it may be possible for a cryptanalyst to exploit these regularities.
    - Example: if it is known that the message always starts out with certain predefined fields, then the cryptanalyst may have a number of known plaintext-ciphertext pairs to work with.
- Hence: not recommended for messages longer than a block.
- **Ideal for a short amount of data, such as an encryption key.**
  - E.g., to transmit a DES or AES key securely.

Modes of operation: Cipher-block Chaining (CBC)

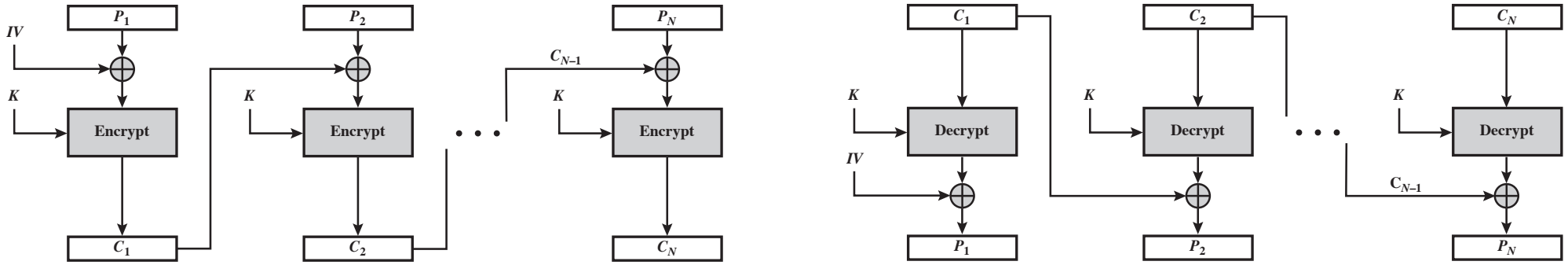
- To overcome security deficiencies of ECB: same plaintext block, if repeated, should produce different ciphertext blocks.

Cipher-block Chaining (CBC)

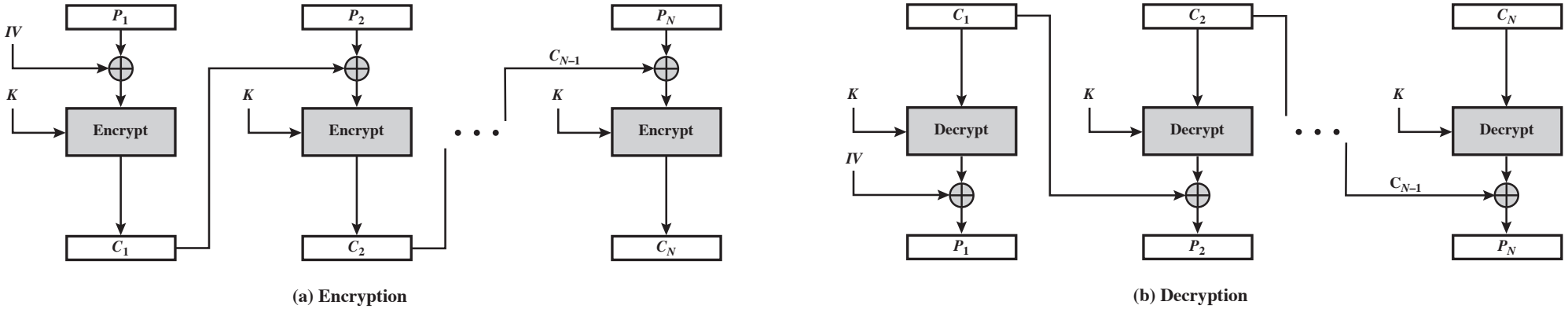
- **Cipher input is XOR of current plaintext block with preceding ciphertext block.**
- For $C_0 = IV$ (an **initialization vector**) and $1 < i \leq N$:

Encryption:	Decryption:
$C_1 = E(K, IV \oplus P_1)$	$P_1 = D(K, C_1) \oplus IV$
$C_i = E(K, C_{i-1} \oplus P_i)$	$P_i = D(K, C_i) \oplus C_{i-1}$

- Requires last block be padded to a full b bits if it is a partial block.



CBC: correctness and initialization vector



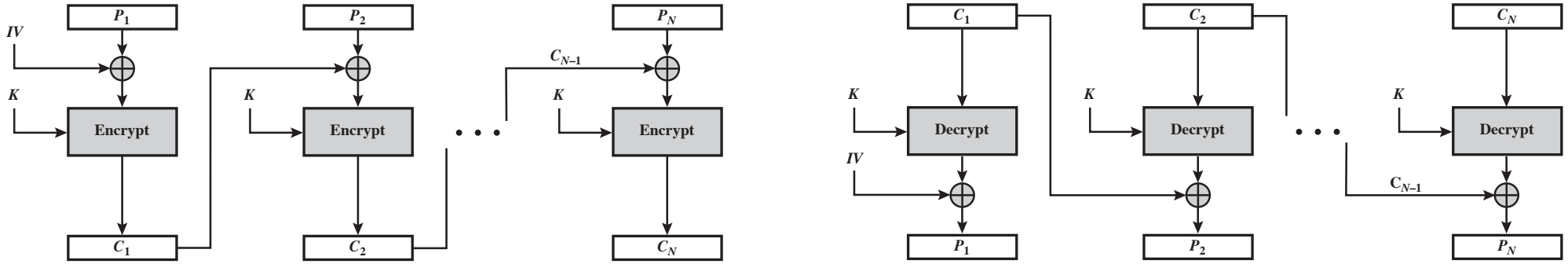
- **Correctness:**

$$\begin{aligned} P_i &= D(K, C_i) \oplus C_{i-1} \\ &= D(K, E(K, C_{i-1} \oplus P_i)) \oplus C_{i-1} \\ &= (C_{i-1} \oplus P_i) \oplus C_{i-1} \\ &= P_i \end{aligned}$$

- **Initialization vector** IV**:**
  - A data block that is that same size as cipher block.
  - IV must be known to both sender and receiver but be unpredictable by a third party and be protected against unauthorized changes.
    - This could be done by sending the IV using ECB encryption.

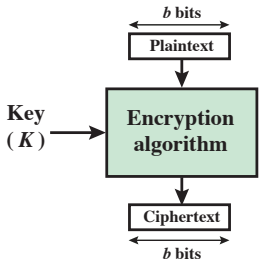
  Attacks based on prior knowledge of IV: see literature.

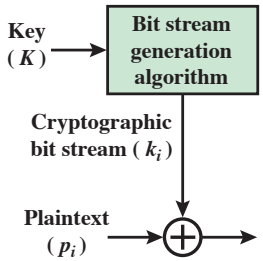
CBC: properties and use

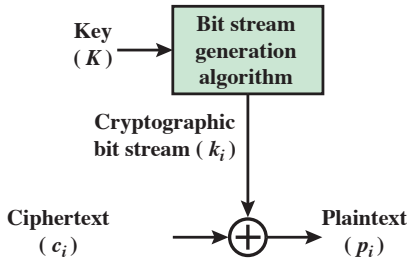


- Repeating patterns of b bits are not exposed:
  - input to the encryption function for each plaintext block bears no fixed relationship to the plaintext block.
- **Properties** (can be proved rigorously):
  - Identical plaintext blocks mapped to different ciphertext.
  - Chaining dependencies: C_i depends on all preceding plaintext.
  - Self-synchronizing: if an error occurs (changed bits, dropped blocks) in C_i but not C_{i+1}, then C_{i+2} is correctly decrypted.
- **CBC appropriate for encrypting messages of length greater than b bits.**
- Can be used for confidentiality and for authentication (see later).

From block cipher to stream cipher





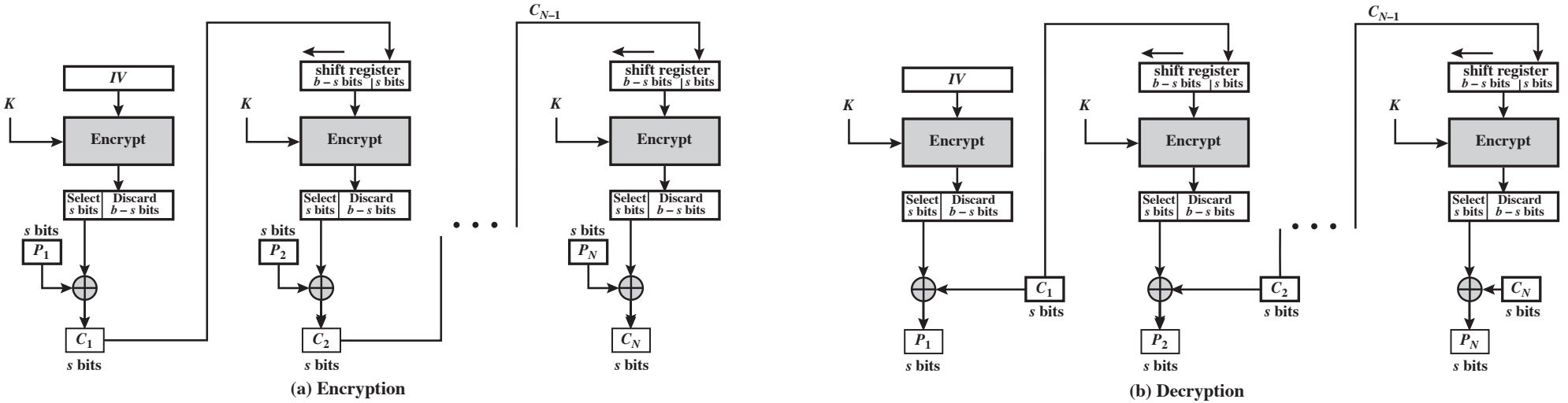


- It is possible to convert a block cipher into a stream cipher, using one of the modes CFB and OFB (and CTR).
- A stream cipher
  - eliminates need to pad a message to be an integral number of blocks,
  - it also can operate in real time: if character stream is being transmitted, each character can be encrypted and transmitted immediately.
- Desirable property of a stream cipher: ciphertext of same length as plaintext.
  - E.g., if 8-bit characters are being transmitted, each character should be encrypted to produce a ciphertext output of 8 bits.

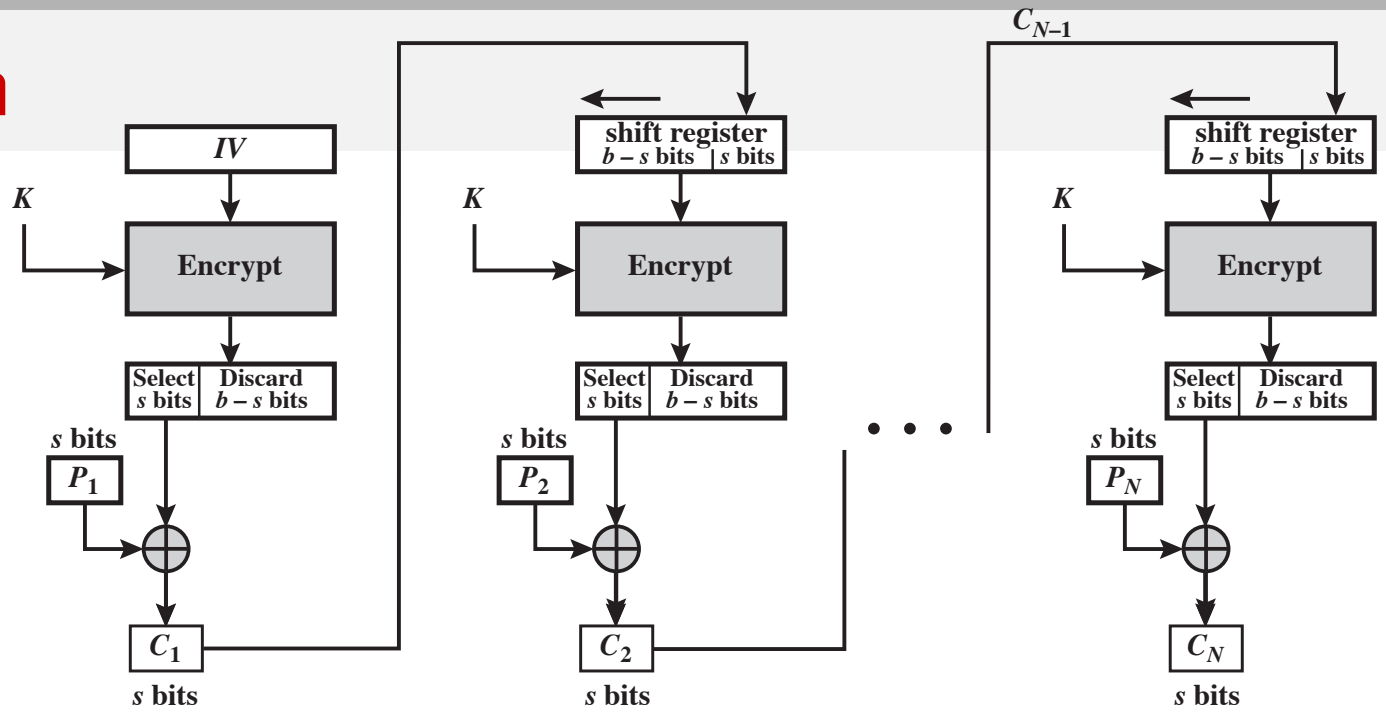
Modes of operation: Cipher FeedBack (CFB)

Cipher FeedBack (CFB)

- As with CBC, units of plaintext are chained together, so that ciphertext of any plaintext unit is a function of all the preceding plaintext.
- Rather than blocks of b bits, plaintext is divided into **segments** (a.k.a. "units of transmission") of s bits.
  - Common value is $s = 8$.

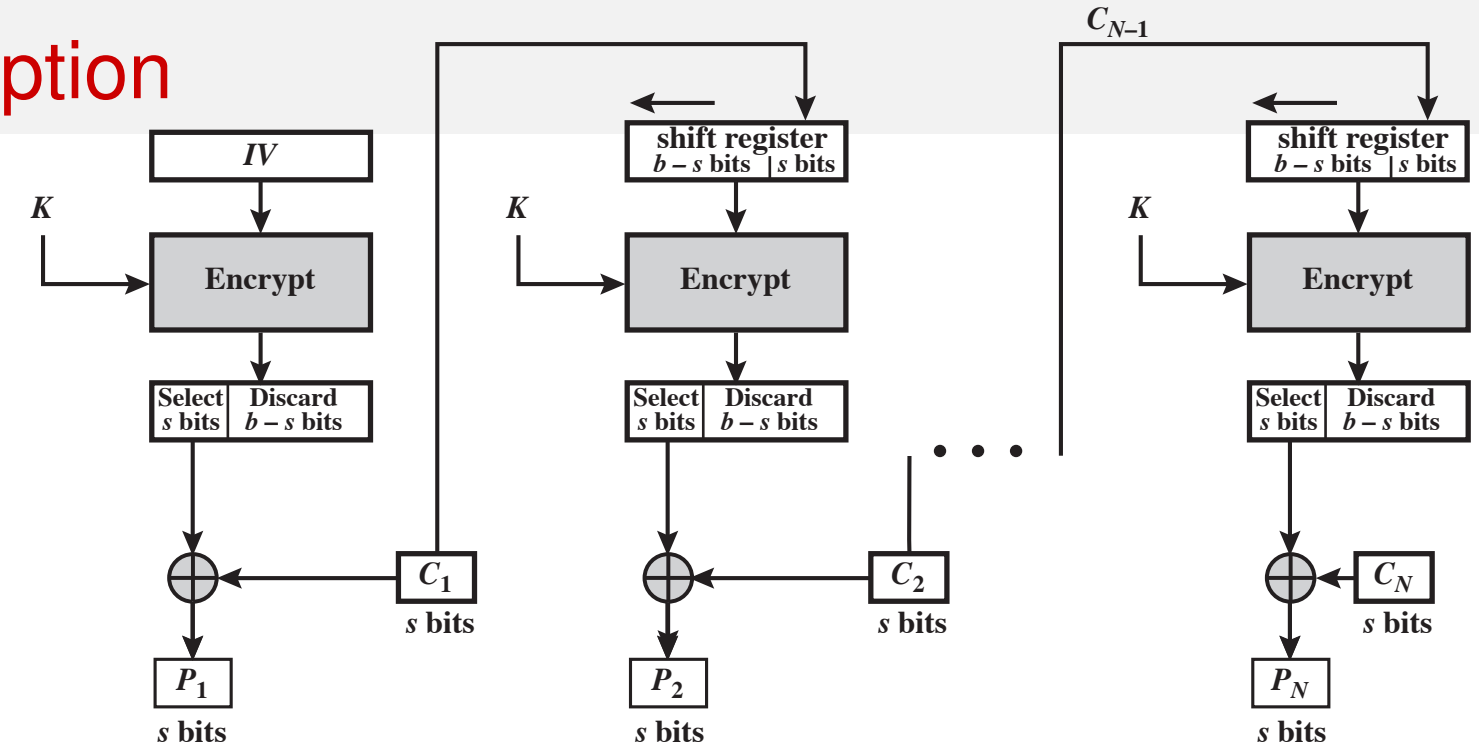


CFB: encryption



- Input: b-bit **shift register** initially set to some initialization vector.
- Let $MSB_s(X)$ be the **most significant s bits** of X.
- First unit of ciphertext $C_1 = P_1 \oplus MSB_s(E(K, IV))$ is XOR of
  - first segment of plaintext P_1
  - leftmost (most significant) s bits of output of encryption function
- Contents of shift register are shifted left by s bits, and C_1 is placed in rightmost (least significant) s bits of shift register.
- Process continues until all plaintext units have been encrypted.

CFB: decryption

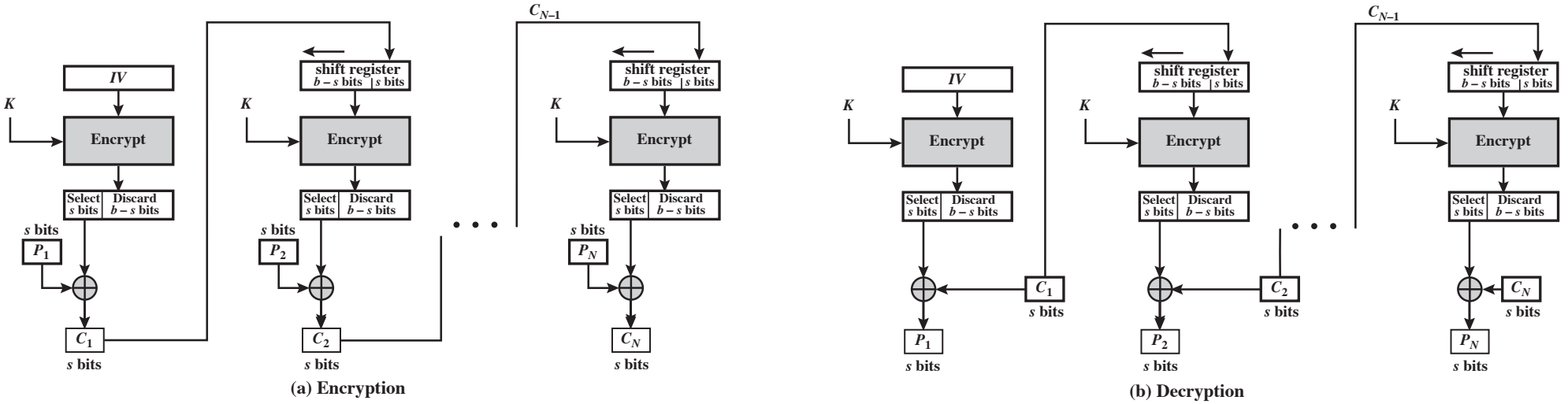


(b) Decryption

- Decryption uses same scheme, except that received ciphertext unit is XORed with output of encryption function to produce plaintext unit:

$$P_1 = C_1 \oplus MSB_s(E(K, IV))$$

  Note: encryption function is used, not decryption function.
- Same reasoning holds for subsequent steps in the process.



Encryption:

$$I_1 = IV$$
$$I_i = LSB_{b-s}(I_{i-1}) \,\|\, C_{i-1}$$
$$O_i = E(K, I_i)$$
$$C_i = P_i \oplus MSB_s(O_i)$$

Decryption:

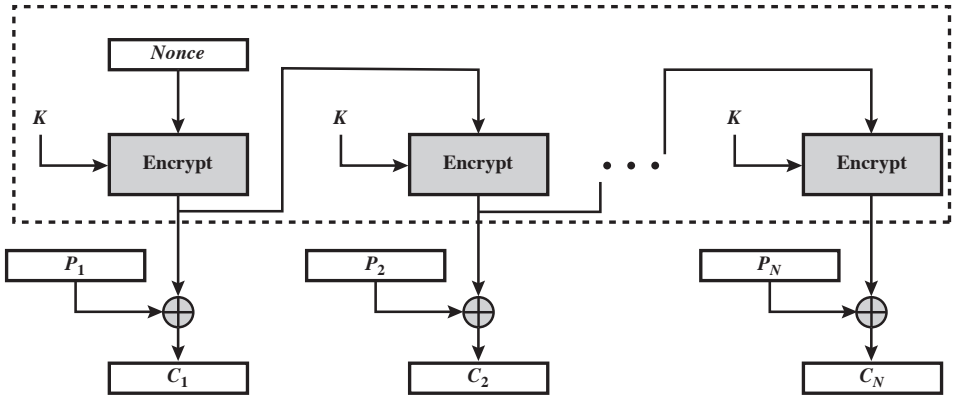
$$I_1 = IV$$
$$I_i = LSB_{b-s}(I_{i-1}) \,\|\, C_{i-1} \quad \text{for } i = 2, \ldots, N$$
$$O_i = E(K, I_i) \quad \text{for } i = 1, \ldots, N$$
$$P_i = C_i \oplus MSB_s(O_i) \quad \text{for } i = 1, \ldots, N$$

- Although CFB can be viewed as a stream cipher, it does not conform to the typical construction of a stream cipher:
  - Typical stream cipher takes as input some initial value and a key and generates a stream of bits, which is then XORed with plaintext bits.
  - CFB: stream of bits that is XORed with plaintext also depends on plaintext.

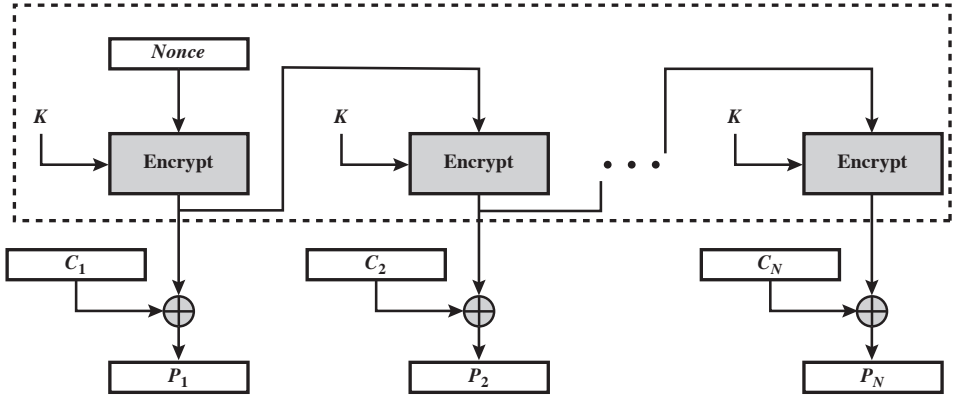
Modes of operation: Output FeedBack (OFB)

Output FeedBack (OFB)

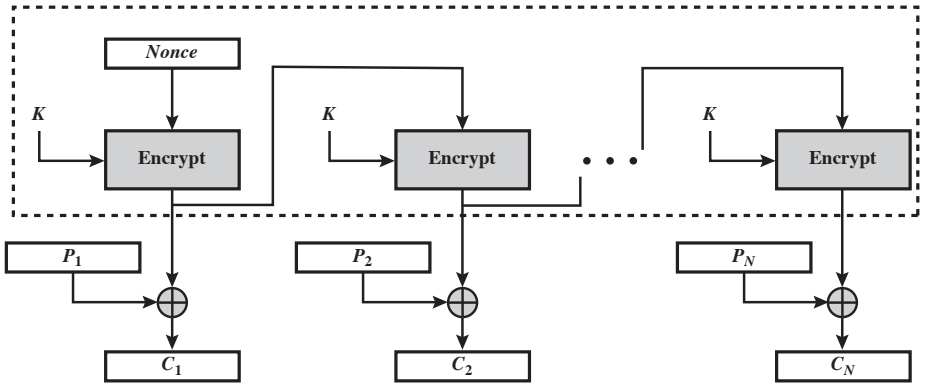
- Similar in structure to CFB, except for 2 differences:
  - Output of encryption function is fed back to shift register in OFB (CFB: ciphertext unit is fed back to the shift register)
  - OFB operates on full blocks, not on an s-bit subset.
- $C_i = P_i \oplus E(K, (\ldots))$ and $P_i = C_i \oplus E(K, (\ldots))$
- **Nonce**: time-varying value that has at most a negligible chance of repeating (e.g., random value that is generated anew for each use, a timestamp, a sequence number, or some combination of these).



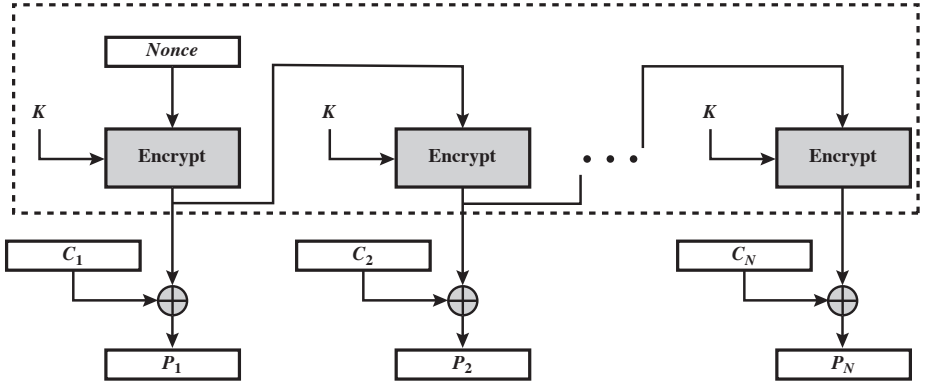
(a) Encryption



(b) Decryption



(a) Encryption



(b) Decryption

Encryption:

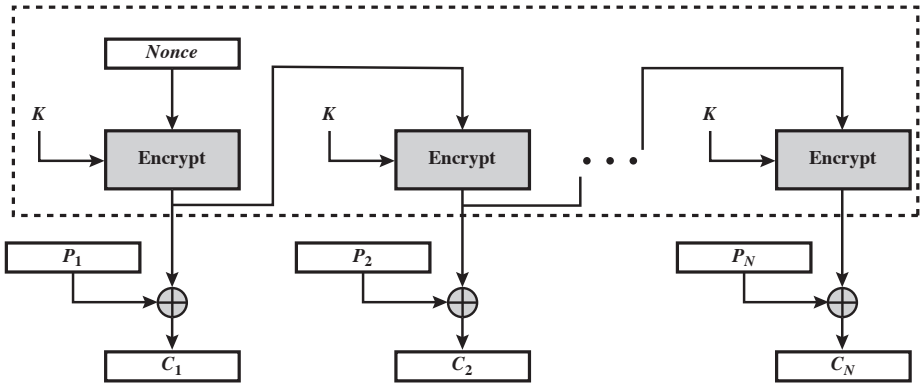
$I_1 = IV$
$I_i = O_{i-1}$
$O_i = E(K, I_i)$
$C_i = P_i \oplus O_i$
$C_N^* = P_N^* \oplus MSB_u(O_N)$

Decryption:

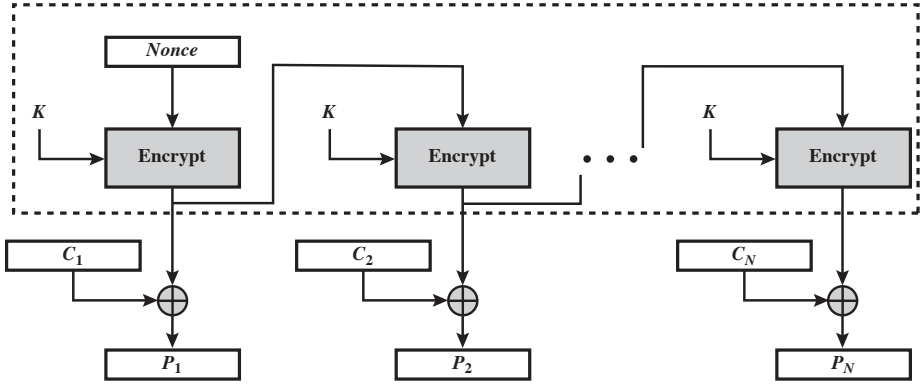
$I_1 = IV$
$I_i = O_{i-1}$
$O_i = E(K, I_i)$
$P_i = C_i \oplus O_i$
$P_N^* = C_N^* \oplus MSB_u(O_N)$

a nonce
for $i = 2, \ldots, N$
for $i = 1, \ldots, N$
for $i = 1, \ldots, N-1$

- Nonce $I_1 = IV$ unique to each execution of encryption operation:
  - Sequence of O_i depends only on key and IV and not on plaintext.
  - For given K and IV, stream O_i used to XOR with stream P_i is fixed.
  - If 2 different messages had an identical block of plaintext in identical position, an attacker could determine that portion of O_i stream.



(a) Encryption



(b) Decryption

Encryption:

$I_1 = IV$ a nonce

$I_i = O_{i-1}$ for $i = 2, \ldots, N$

$O_i = E(K, I_i)$ for $i = 1, \ldots, N$

$C_i = P_i \oplus O_i$ for $i = 1, \ldots, N-1$

$C_N^* = P_N^* \oplus MSB_u(O_N)$

Decryption:

$I_1 = IV$

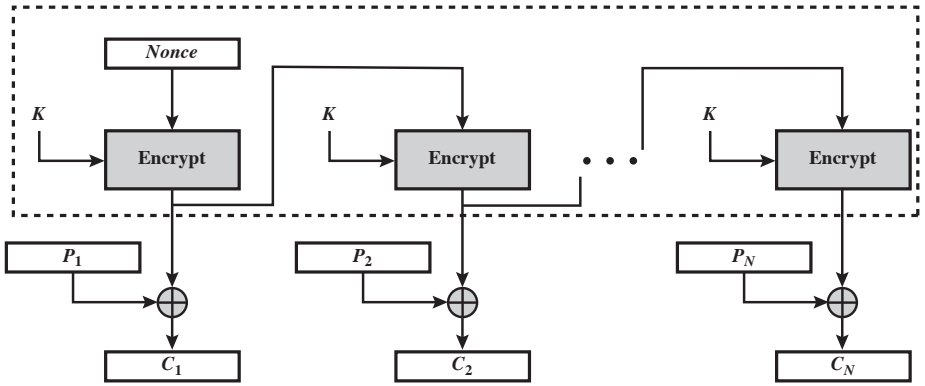
$I_i = O_{i-1}$

$O_i = E(K, I_i)$

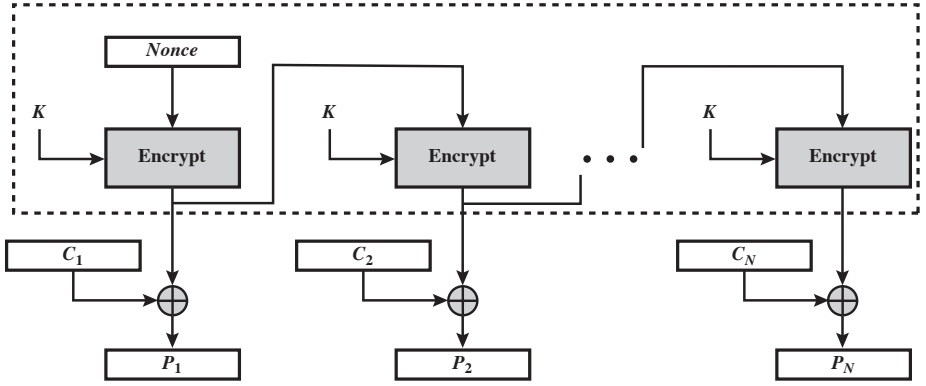
$P_i = C_i \oplus O_i$

$P_N^* = C_N^* \oplus MSB_u(O_N)$

- Let block size be b.
- If last plaintext block P_N contains $u < b$ bits (indicated by *), most significant u bits of last output block O_N are used for XOR.
- Remaining $b - u$ bits of the last output block are discarded.



(a) Encryption



(b) Decryption

Encryption:

$I_1 = IV$ a nonce

$I_i = O_{i-1}$ for $i = 2, \ldots, N$

$O_i = E(K, I_i)$ for $i = 1, \ldots, N$

$C_i = P_i \oplus O_i$ for $i = 1, \ldots, N-1$

$C_N^* = P_N^* \oplus MSB_u(O_N)$

Decryption:

$I_1 = IV$

$I_i = O_{i-1}$

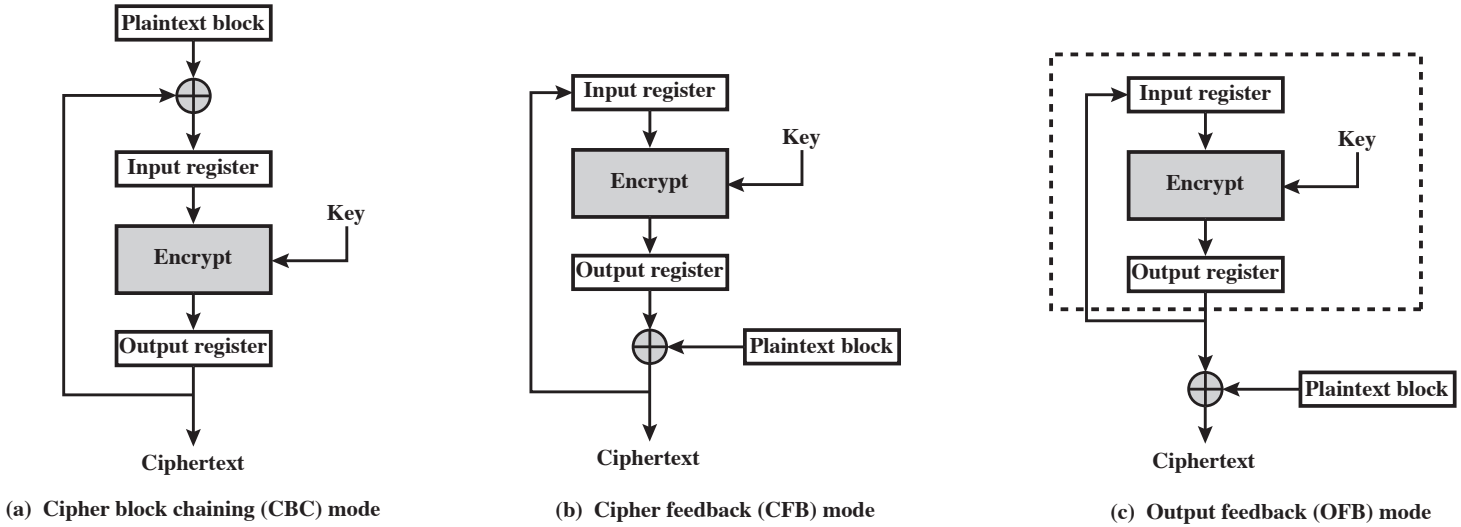
$O_i = E(K, I_i)$

$P_i = C_i \oplus O_i$

$P_N^* = C_N^* \oplus MSB_u(O_N)$

- OFB: structure of typical stream cipher (but one block at a time).
  - Generates a stream of bits as a function of an initial value and a key,
  - and that stream of bits is XORed with plaintext bits.
- Generated stream that is XORed with plaintext is itself independent of plaintext (highlighted by dashed boxes).
- OFB can alternatively be used keeping the shift register of CFB.

Feedback characteristic of modes of operation



(a) Cipher block chaining (CBC) mode (b) Cipher feedback (CFB) mode (c) Output feedback (OFB) mode

- All of NIST-approved block cipher modes of operation (except ECB) involve **feedback**.
  - Think of encryption as taking input from a input register whose length equals encryption block length and with output stored in an output register.
  - Input register updated one block at a time by feedback mechanism.
  - After each update, encryption algorithm is executed, producing a result in output register.
  - Meanwhile, a block of plaintext is accessed.
- OFB and CTR: output independent of both plain- and ciphertext.
  - Thus, they are natural candidates for stream ciphers that encrypt plaintext by XOR one full block at a time.

Bibliography

Most of the figures in this lecture are taken from:

- William Stallings. *Cryptography and Network Security. Principles and Practice*, 7th ed., Prentice Hall, 2016.

Other interesting books:

- Dieter Gollmann. *Computer Security*. Wiley, 2011.
- Bruce Schneier. *Applied Cryptography*, John Wiley & Sons, 1996 (and 20th anniversary edition in 2016).
- Alfred J. Menezes, Paul C. van Oorschot, Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996. Available online.
- Arthur E. Hutt, Seymour Bosworth, Douglas B. Hoyt. *Computer Security Handbook*. Wiley, 1995.

Try out some ciphers and cryptanalysis with CrypTool-Online:
`http://www.cryptool-online.org`
and `http://www.simonsingh.net/The_Black_Chamber/latinsquare.html`

Additional (non-textbook) references and websites

- Simon Singh. *The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptograhy*. Anchor Books, 2000.
- David Kahn. *The Codebreakers: The Story of Secret Writing*. Scribner, 1996.
- Bruce Schneier. *Secrets and Lies*. Wiley, 2000.
- The Enigma Machine: `http://www.cryptomuseum.com` and `http://www.codesandciphers.org.uk`