PennState

CSE 443: Introduction to Computer Security
Module:

Prof. Syed Rafiul Hussain
Department of Computer Science and Engineering
The Pennsylvania State University

A historical moment …

- Mary Queen of Scots is being held by Queen Elizabeth …
  - … and accused of treason.
  - All communication with co-conspirators encrypted.

- Walsingham needs to prove complicity.

http://5010.mathed.usu.edu/Fall2014/KKing/sigmary.html

Intuition

- Cryptography is the art (and sometimes science) of secret writing
  - Less well known is that it is also used to guarantee other properties, e.g., authenticity of data
  - This is an enormously deep and important field
  - However, much of our trust in cryptographic systems is based on faith (particularly in efficient secret key algorithms)
  - … ask Mary Queen of Scots how that worked out.

- This set of lectures will provide the intuition and some specifics of modern cryptography, seek others for additional details (Menezes et. al.).

Cryptography

- Cryptography (cryptographer)
  - Creating ciphers
- Cryptanalysis (cryptanalyst)
  - Breaking ciphers

- The history of cryptography is an arms race between cryptographers and cryptanalysts

Goals of Cryptography

- The most fundamental problem cryptography addresses:
  - ensure security of communication over insecure medium

- What does secure communication mean?
  - confidentiality (privacy, secrecy)
    - only the intended recipient can see the communication
  - integrity (authenticity)
    - the communication is generated by the alleged sender
- What does insecure medium mean?
  - Two possibilities:
    - Passive attacker: the adversary can eavesdrop
    - Active attacker: the adversary has full control over the communication channel

Approaches to Secure Communication

- Steganography
  - “covered writing”
  - hides the existence of a message
  - depends on secrecy of method

- Cryptography
  - “hidden writing”
  - hide the meaning of a message
  - depends on secrecy of a short key, not method

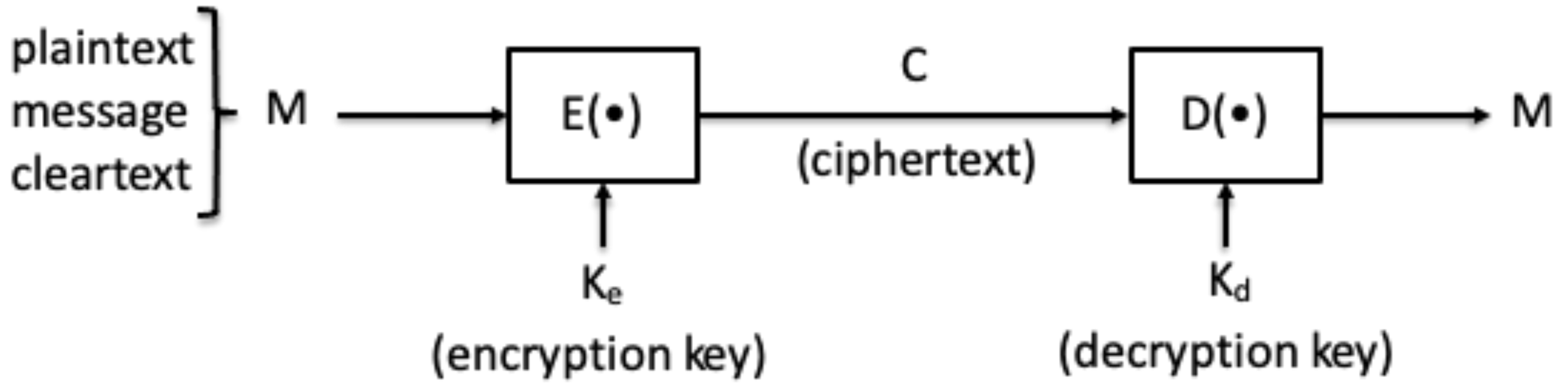
Cryptography < Security

- Cryptography isn't the solution to security
  - Buffer overflows, worms, viruses, trojan horses, SQL injection attacks, cross-site scripting, bad programming practices, etc.

- It's a *tool*, not a *solution*

- Even when used, difficult to get right
  - Choice of encryption algorithms
  - Choice of parameters
  - Implementation
  - Hard to detect errors
    - Even when crypto fails, the program may still work
    - May not learn about crypto problems until after they've been exploited

Basic Terminology in Cryptography



More formally:

$E(K_e, M) = C$

$D(K_d, C) = M$

And has the property:

$D(K_d, E(K_e, M)) = M$

Main Types of Cryptography

- Secret key == symmetric key
  - Encryption and decryption keys are the same

- Public key == asymmetric key
  - Encryption and decryption keys differ!

- We'll start with symmetric key

Security Principle

- Kerckhoff's Principle:
  - A cryptosystem should be secure even if everything about the system, except the key, is public knowledge.

- Shannon's Maxim
  - The enemy knows the system

  - The security by obscurity does not work!
  - Should assume that the adversary knows the algorithm.
    The only secret that the adversary is assumed to not know is the key.
  - What is the difference between algorithm and key?

Encryption algorithm

- Algorithm used to make content unreadable by all but the intended receivers
  - E(plaintext,key) = ciphertext
  - D(ciphertext,key) = plaintext
- Algorithm is public, key is private

Hardness

- Inputs
  - Plaintext P
  - Ciphertext C
  - Encryption key ke
  - Decryption key kd
  - D(E(P, ke),kd) = P
- Computing P from C is hard, P from C with kd is easy
  - for all Ps with more than negligible probability
  - This is known as a TRAPDOOR function
    - y = fk(x) is easy, but x = fk-1(y) infeasible if y is known and k is unknown.
  - Devil is in the details ....

Example: Caesar Cipher

- Shift cipher
- Every character is replaced with the character three slots to the right

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
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

- Q: What is the key?

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

- The Key Space: [1 .. 25]
- Encryption given a key K:
  - each letter in the plaintext P is replaced with the K'th letter following corresponding number (shift right)
- Decryption given K:

Cryptanalyze this ….

- "GUVF VF N TERNG PYNFF"

Cryptanalysis of ROTx

- Goal: to find plaintext of encoded message
- Given: ciphertext
- How: simply try all possible keys
  - Known as a brute force attack
  - key space is small (<= 26 possible keys).

I T F D V S J U Z B M E Q S J W B D Z

2 U G E W T K V A C N F R T H X C E A

3 V H F X U L W B D Q G S U L Y D F B

S E C U R I T Y A N D P R I V A C Y

General Mono-alphabetic Substitution Cipher

- A substitution cipher replaces one symbol for another in the alphabet
  - Caesar cipher and rot13 are a specific kind (rotation)
  - The most common is a random permutation cipher

$$\begin{tikzcd}[column sep=small]
A \arrow[d] & B \arrow[d] & C \arrow[d] & D \arrow[d] & E \arrow[d] & F \arrow[d] & G \arrow[d] & H \arrow[d] & I \arrow[d] & J \arrow[d] & K \arrow[d] & L \arrow[d] & M \arrow[d] \\
C & M & T & E & F & H & P & U & D & X & N & Z & L
\end{tikzcd}$$

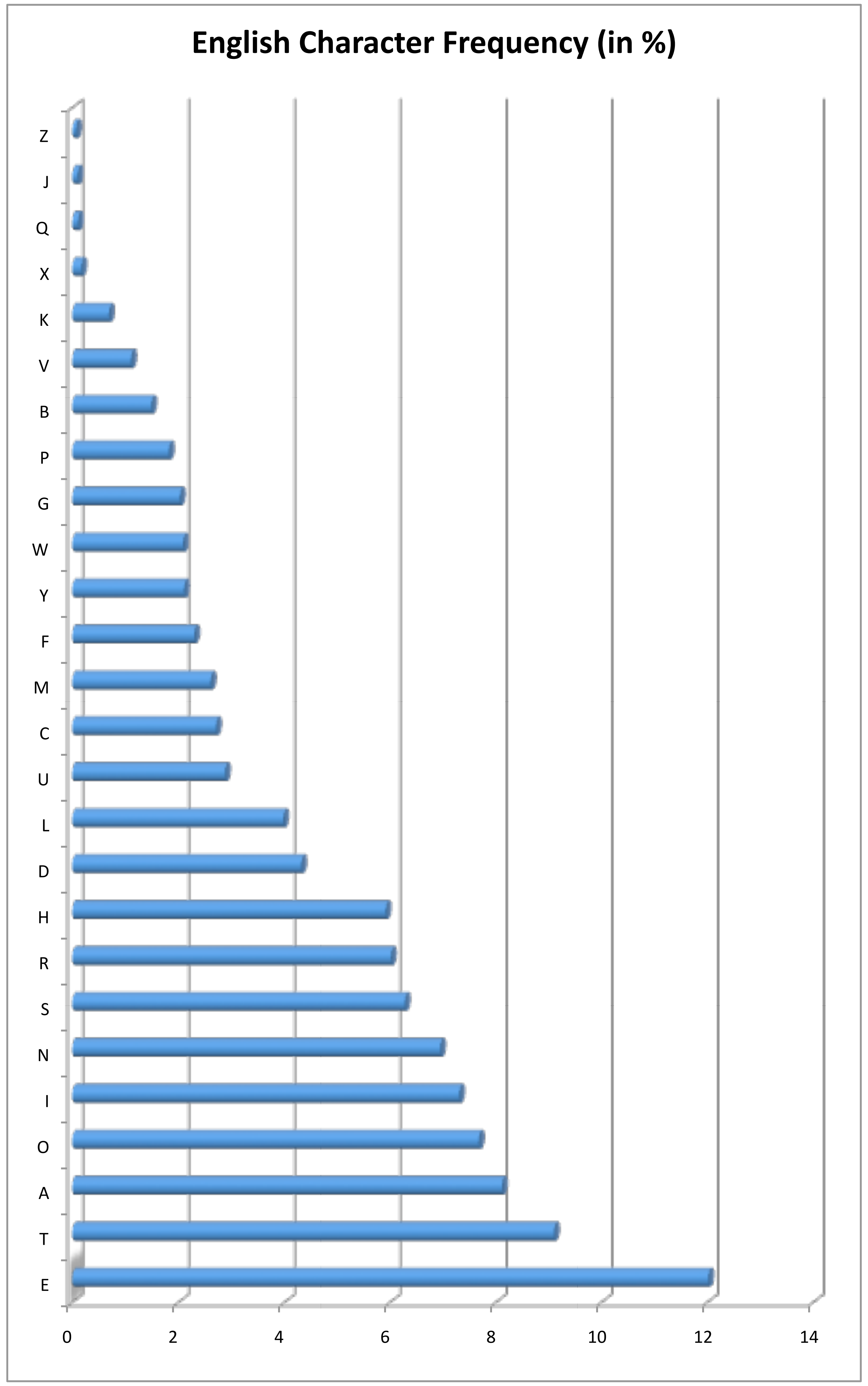
$$\begin{tikzcd}[column sep=small]
N \arrow[d] & O \arrow[d] & P \arrow[d] & Q \arrow[d] & R \arrow[d] & S \arrow[d] & T \arrow[d] & U \arrow[d] & V \arrow[d] & W \arrow[d] & X \arrow[d] & Y \arrow[d] & Z \arrow[d] \\
O & A & J & R & Y & I & G & W & V & B & S & Q & K
\end{tikzcd}$$

Strength of General Substitution Cipher

- Exhaustive search is difficult
  - key space size is $26! \approx 4\times10^{26}$
- Dominates the art of secret writing throughout the first millennium A.D.
- Thought to be unbreakable by many back then

Why are substitution ciphers

- Each language has certain features: frequency of letters, or of groups of two or more letters.
- Substitution ciphers preserve the language features.
- Substitution ciphers are breakable because they don't hide the underlying frequency of characters. You can use this information if you know the target language frequency count.
- For example, in English ...
  - e,t,a,o,i,n,s,r,h,d,l,u,c,m,f,y, w,g,p,b,v,k,x,q,j,z
- Q: how do you exploit this?



Using frequency ..

- Vg gbbx n ybg bs oybbq, fjrng naq grnef gb trg gb jurer jr ner gbqnl, ohg jr unir whfg ortha. Gbqnl jr ortva va rnearfg gur jbex bs znxvat fher gung gur jbeyq jr yrnir bhe puvyqera vf whfg n yvggyr ovg orggre guna gur bar jr vaunovg gbqnl.

Using frequency ..

- Vg gbbx n ybg bs oybbq, fjrng naq grnef gb trg gb jurer jr ner gbqnl, ohg jr unir whfg ortha. Gbqnl jr ortva va rnearfg gur jbex bs znxvat fher gung gur jbeyq jr yrnir bhe puvyqera vf whfg n yvggyr ovg orggre guna gur bar jr vaunovg gbqnl.

- It took a lot of blood, sweat and tears to get to where we are today, but we have just begun. Today we begin in earnest the work of making sure that the world we leave our children is just a little bit better than the one we inhabit today.

'r' appears very frequently so very likely is one of the top frequency letters.

Using frequency ..

- Vg gbbx n ybg bs oybbq, fj**r**ng naq g**r**nef gb t**r**g gb ju**r**e**r** j**r** ne**r** gbqnl, ohg j**r** uni**r** whfg o**r**tha. Gbqnl j**r** o**r**tva va **r**nea**r**fg gu**r** jbex bs znxvat fhe**r** gung **gur** jbeyq j**r** y**r**ni**r** bhe puvyqe**r**a vf whfg n yvggy**r** ovg o**r**gg**r**e guna **gur** ba**r** j**r** vaunovg gbqnl.

- It took a lot of blood, sw**e**at and t**e**ars to get to wh**e**r**e** w**e** ar**e** today, but w**e** hav**e** just b**e**gun. Today w**e** b**e**gin in **e**arn**e**st th**e** work of making sur**e** that **the** world w**e** l**e**av**e** our childr**e**n is just a littl**e** bit b**e**tt**e**r than **the** on**e** w**e** inhabit today.

Repeat this process, picking out more letters, then common words, e.g., 'the'

... which gives (e to r), (g to t), and (u to h)

Defeat Frequency Analysis

- Use larger blocks as the basis of substitution. Rather than substituting one letter at a time, substitute 64 bits at a time, or 128 bits.
  - Leads to block ciphers such as DES & AES.

- Use different substitutions to get rid of frequency features.
  - Leads to polyalphabetical substituion ciphers
  - Stream ciphers

Vigenè re Cipher (Polyalphabetic Cipher)

Treat letters as numbers: [A=0, B=1, C=2, …, Z=25]

Number Theory Notation: $Z_n = \{0, 1, \ldots, n-1\}$

Definition:

Given m, a positive integer, $P = C = (Z_{26})^n$, and $K = (k_1, k_2, \ldots, k_m)$ a key, we define:

Encryption:

$e_k(p_1, p_2 \ldots p_m) = (p_1+k_1, p_2+k_2 \ldots p_m+k_m) \pmod{26}$

Decryption:

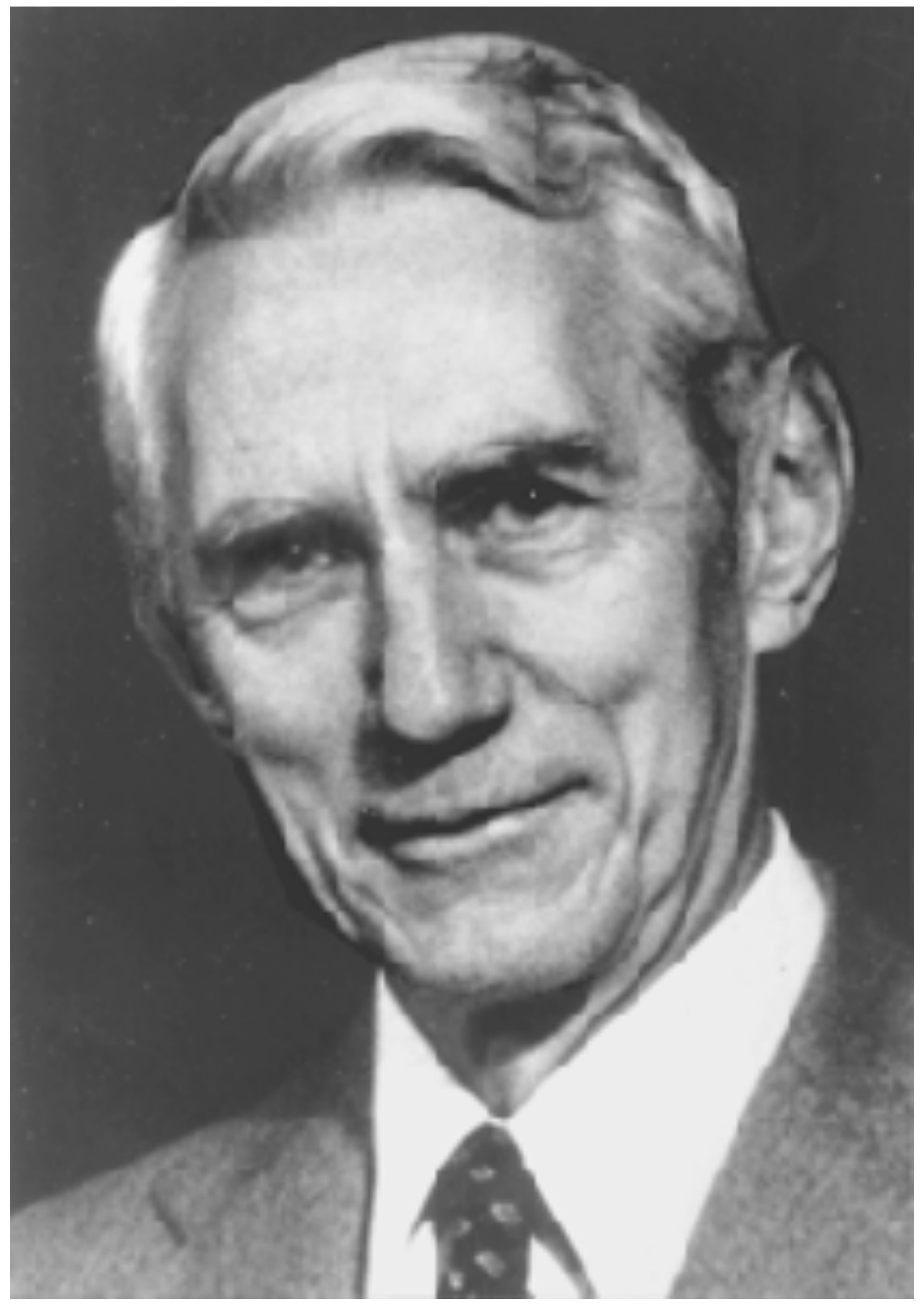
$d_k(c_1, c_2 \ldots c_m) = (c_1-k_1, c_2-k_2 \ldots c_m-k_m) \pmod{26}$

Example:

Plaintext:	C R Y P T O G R A P H Y
Key:	L U C K L U C K L U C K
Ciphertext:	N L A Z E I I B L J J I

Is there an unbreakable cipher?

- As it turns out, yes ….
  - (Claude Shannon proved it)



One-time Pad

- Fix the vulnerability of the Vigenere cipher by using very long keys
- Key is a random string that is at least as long as the plaintext
- Encryption is similar to shift cipher
- Invented by Vernam in the 1920s
-

One-Time Pad

- Let $Z_m = \{0,1,\ldots,m-1\}$ be the alphabet.
- Plaintext space = Ciphtertext space = Key space = $(Z_m)^n$
- The key is chosen uniformly randomly
- Plaintext $X = (x_1\ x_2 \ldots x_n)$
- Key $K = (k_1\ k_2 \ldots k_n)$
- Ciphertext $Y = (y_1\ y_2 \ldots y_n)$
- $e_k(X) = (x_1+k_1\quad x_2+k_2 \ldots x_n+k_n) \bmod m$
- $d_k(Y) = (y_1-k_1\quad y_2-k_2 \ldots\ y_n-k_n) \bmod m$

The one-time pad (OTP)

- Assume you have a secret bit string s of length n known only to two parties, Alice and Bob
  - Alice sends a message m of length of n to Bob
  - Alice uses the following encryption function to generate ciphertext bits:

$$\sum_{i=0}^{n} c_i = m_i \oplus k_i$$

  - E.g., XOR the data with the secret bit string
  - An adversary Mallory cannot retrieve any part of the data
- Simple version of the proof of security:
  - Assume for simplicity that value of each bit in k is equally likely, then you have no information to work with.

OTP Guarantees Perfect Secrecy

- The ciphertext reveals absolutely no information about the plaintext.
  - Pr [PT=m | CT=c] = Pr [PT = m].

- Simple example:
  - c = 0
  - if k = 0 them m = 0
  - if k = 1 then m = 1
  - Equal probability that it could be any message

Key Randomness in OTP

- One-Time Pad uses a very long key, what if the key is not chosen randomly, instead, texts from, e.g., a book are used as keys.
  - this is not One-Time Pad anymore
  - this does not have perfect secrecy
  - this can be broken
  - How?
- The key in One-Time Pad should never be reused.
  - If it is reused, it is Two-Time Pad, and is insecure!
  - Why?
    - $C1 = (M1 \oplus K); C2 = (M2 \oplus K)$
    - $(C1 \oplus C2) = (M1 \oplus K) \oplus (M2 \oplus K) =$ $M1 \oplus M2$
      - This is worse than it looks!

Usage of OTP

- To use one-time pad, one must have keys as long as the messages.
- To send messages totaling certain size, sender and receiver must agree on a shared secret key of that size.
  - Typically by sending the key over a secure channel
  - This is difficult to do in practice.
- Can't one use the channel for sending the key to send the messages instead?
- Why is OTP still useful, even though difficult to use?

Information-Theoretic Security vs. Computational Security

- *Unconditional* or *information-theoretic security*: cryptosystem offers provable guarantees, irrespective of computational abilities of an attacker
  - Given ciphertext, the probabilities that bit i of the plaintext is 0 is p and the probability that it is 1 is (1-p)
  - E.g., one-time pad
  - often requires key sizes that are equal to size of plaintext

- *Conditional* or *computational security*: cryptosystem is secure assuming a computationally bounded adversary, or under certain hardness assumptions (e.g., P<>NP)
  - E.g., DES, 3DES, AES, RSA, DSA, ECC, DH, MD5, SHA
  - Key sizes are much smaller (~128 bits)

- Almost all deployed modern cryptosystems are conditionally secure

Stream Ciphers vs. Block Ciphers

- *Stream Ciphers*
  - Combine (e.g., XOR) plaintext with pseudorandom stream of bits
  - Pseudorandom stream generated based on key
  - XOR with same bit stream to recover plaintext
  - E.g., RC4, FISH

- *Block Ciphers*
  - Fixed block size
  - Encrypt block-sized portions of plaintext
  - Combine encrypted blocks (more on this later)
  - E.g., DES, 3DES, AES

Stream Ciphers

- A *stream cipher* is an algorithm that generates a long keystream from a (short) key
  - It's effectively acting as a random number generator with the key (and other information) as the seed
- The key stream can be used like the pad in OTP

Stream Cipher

- In OTP, a key is a random string of length at least the same as the message
- Stream ciphers:
  - Idea: replace "rand" by "pseudo rand"
  - Use Pseudo Random Number Generator
  - PRNG: $\{0,1\}^s \rightarrow \{0,1\}^n$
    - expand a short (e.g., 128-bit) random seed into a long (e.g., 196 bit) string that "looks random"
  - Secret key is the seed
  - $E_{key}[M]$ = M xor PRNG(key)
- Aside: Pseudorandomness
  - True randomness is (1) uniform distribution of bits, and (2) Independence
  - Pseudorandomness is: (1)Approximately uniform distribution of bits; and (2)Cannot be independent since they are deterministically generated!

RC4 Stream Cipher

- A proprietary cipher owned by RSA, designed by Ron Rivest in 1987.
- Became public in 1994.
- Simple and effective design.
- Variable key size (typical 40 to 256 bits),
- Output unbounded number of bytes.
- Widely used (web SSL/TLS, wireless WEP).
- Extensively studied, not a completely secure PRNG (keystream is not truly random), first part of output biased, when used as stream cipher, should use RC4-Drop[n]
    - Which drops first n bytes before using the output
    - Conservatively, set n=3072

Using Stream Cipher in Practice

- If the same key stream is used twice, then easy to break.
  - This is a fundamental weakness of stream ciphers; it exists even if the PRNG used in the ciphers is strong
- In practice, one key is used to encrypt many messages
  - Example: Wireless communication
  - Solution: Use Initial vectors (IV).
  - Ekey[M] = [IV, M xor PRNG(key || IV)]
    - IV is sent in clear to receiver;
  - IV needs integrity protection, but not confidentiality protection
  - IV ensures that key streams do not repeat, but does not increase cost of brute-force attacks
  - Without key, knowing IV still cannot decrypt

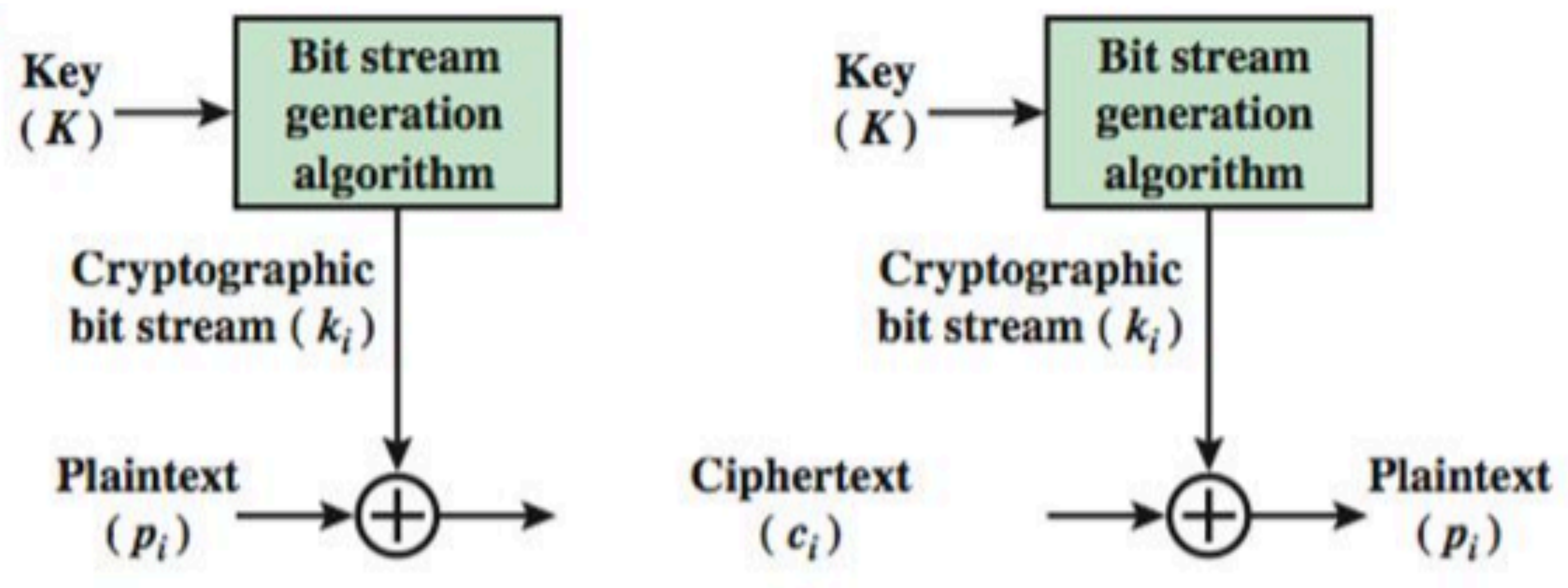
Shared key cryptography

- Traditional use of cryptography
- Symmetric keys, where a single key (k) is used for E and D

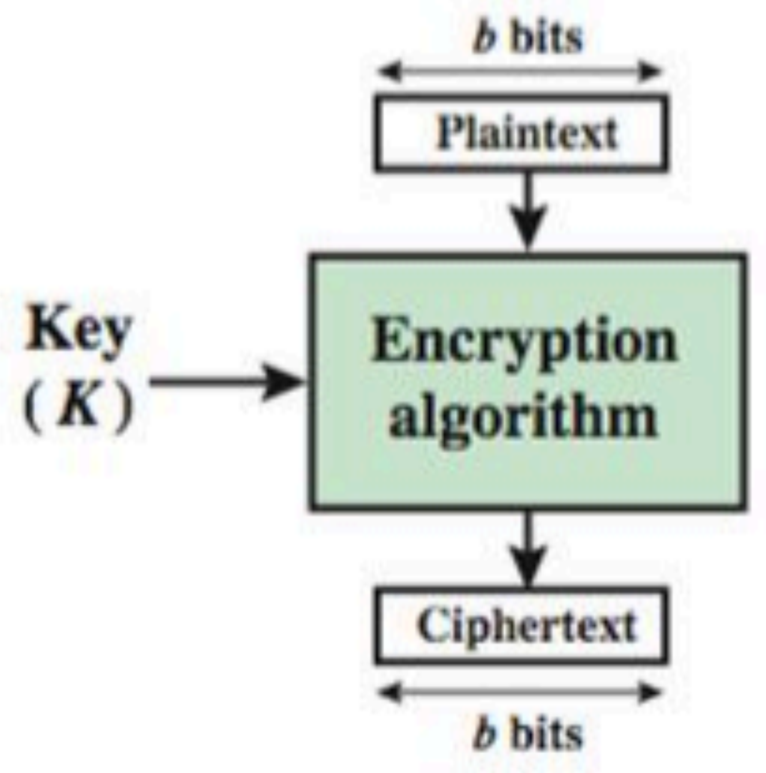
$$D(E(p, k), k)) = P$$

- All (intended) receivers have access to key
- Note: Management of keys determines who has access to encrypted data
  - E.g., password encrypted email
- Also known as symmetric key cryptography

Stream vs. Block Cipher


Key
(K)
Bit stream generation algorithm
Cryptographic bit stream (k_i)
Plaintext
(p_i)
Key
(K)
Bit stream generation algorithm
Cryptographic bit stream (k_i)
Ciphertext
(c_i)
Plaintext
(p_i)

(a) Stream Cipher Using Algorithmic Bit Stream Generator



(b) Block Cipher

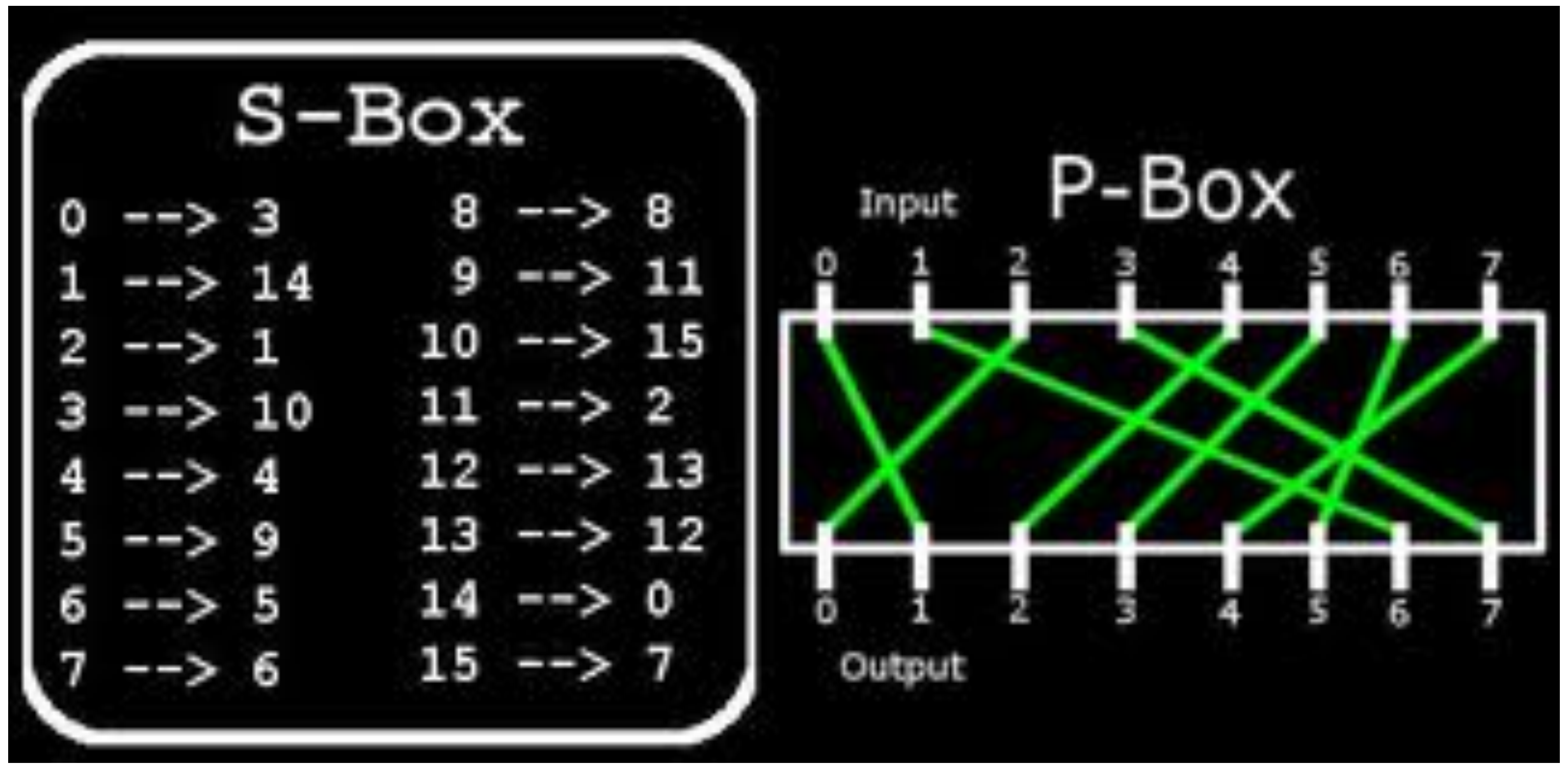
Block Cipher

- A block cipher is a function that replaces a *fixed length input* with a fixed length output
  - The input/output size is called the "block size"

- A block cipher can be thought of as a bijection on the input/output space
  - The *key* is the mapping of plaintext to ciphertext
  - More accurately, the key is used to deterministically generate the mapping.

Generic Block Encryption

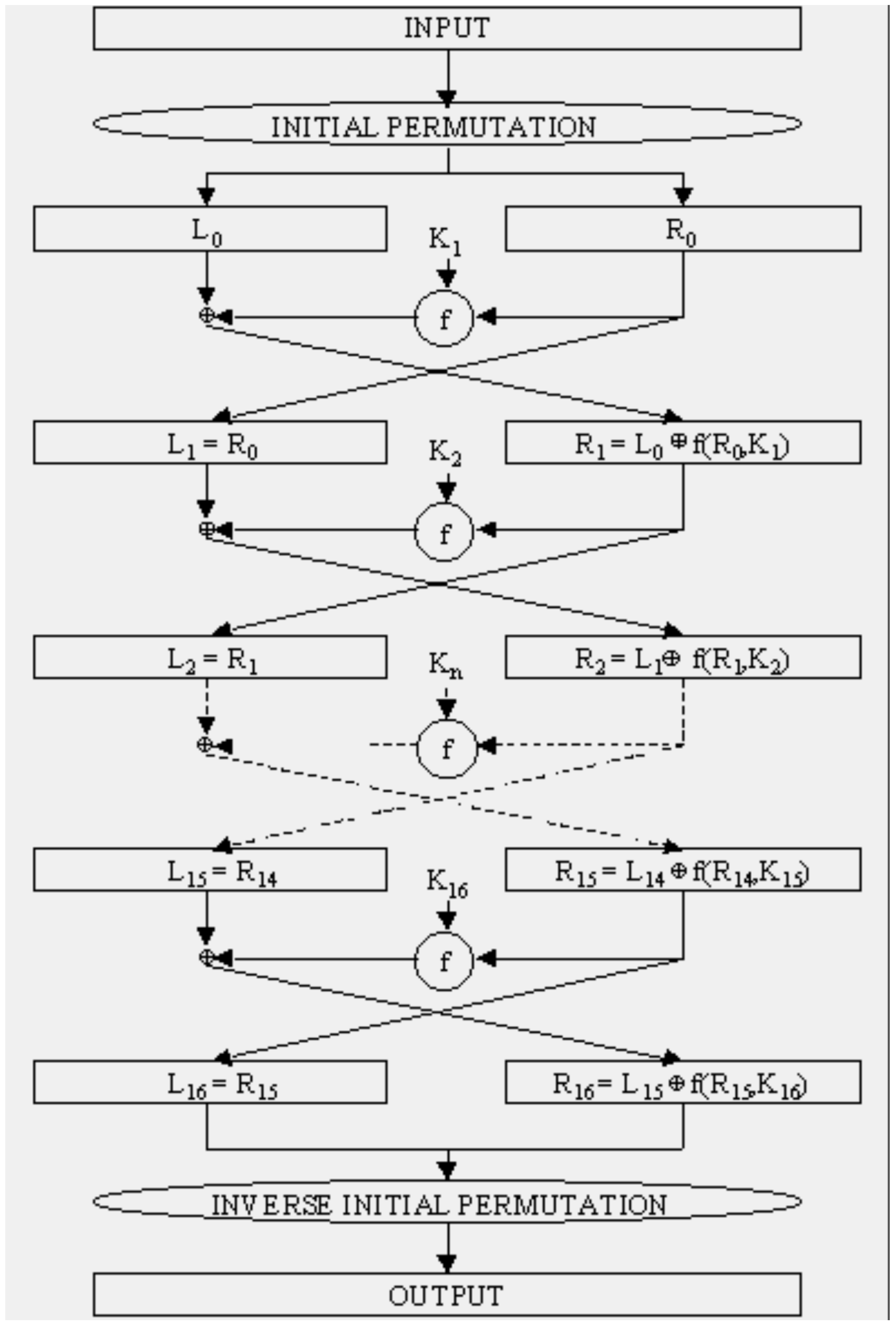
- Break input into smaller chunks
- Apply substitution on smaller chunks and permutation on output of the substitution
- Achieves Shannon’s properties of confusion and diffusion
  - Confusion: Relation between ciphertext and key as complex as possible
  - Diffusion: Relation between ciphertext and plaintext as complex as possible
- Multiple rounds
- Plaintext easily recovered



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

Data Encryption Standard (DES)

- Introduced by the US NBS (now NIST) in 1972
- Signaled the beginning of the modern area of cryptography
- Block cipher
  - Fixed sized input
- 8-byte input and a 8-byte key (56-bits+8 parity bits)
- Multiple rounds of substitution, initial and final permutation



INPUT
INITIAL PERMUTATION
L_0
K_1
R_0
f
$L_1 = R_0$
K_2
$R_1 = L_0 \oplus f(R_0, K_1)$
$L_2 = R_1$
K_n
$R_2 = L_1 \oplus f(R_1, K_2)$
$L_{15} = R_{14}$
K_{16}
$R_{15} = L_{14} \oplus f(R_{14}, K_{15})$
$L_{16} = R_{15}$
$R_{16} = L_{15} \oplus f(R_{15}, K_{16})$
INVERSE INITIAL PERMUTATION
OUTPUT

Fiestal Cipher

Encryption

Split the plaintext block into two equal pieces, (L_0, R_0)

For each round $i = 0, 1, \ldots, n$, compute

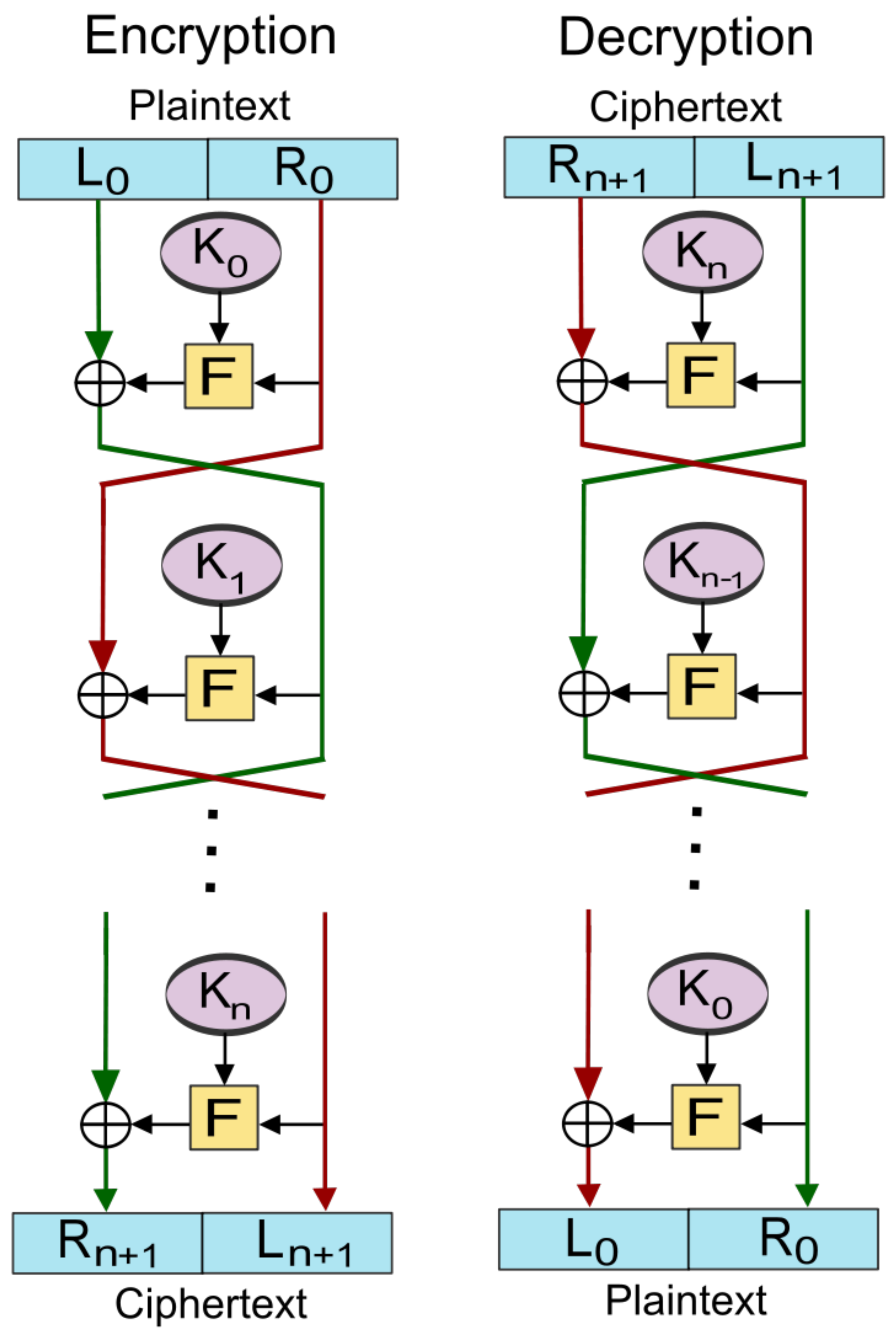
$$L_{i+1} = R_i$$
$$R_{i+1} = L_i \oplus \mathrm{F}(R_i, K_i).$$

Then the ciphertext is (R_{n+1}, L_{n+1}).

Decryption

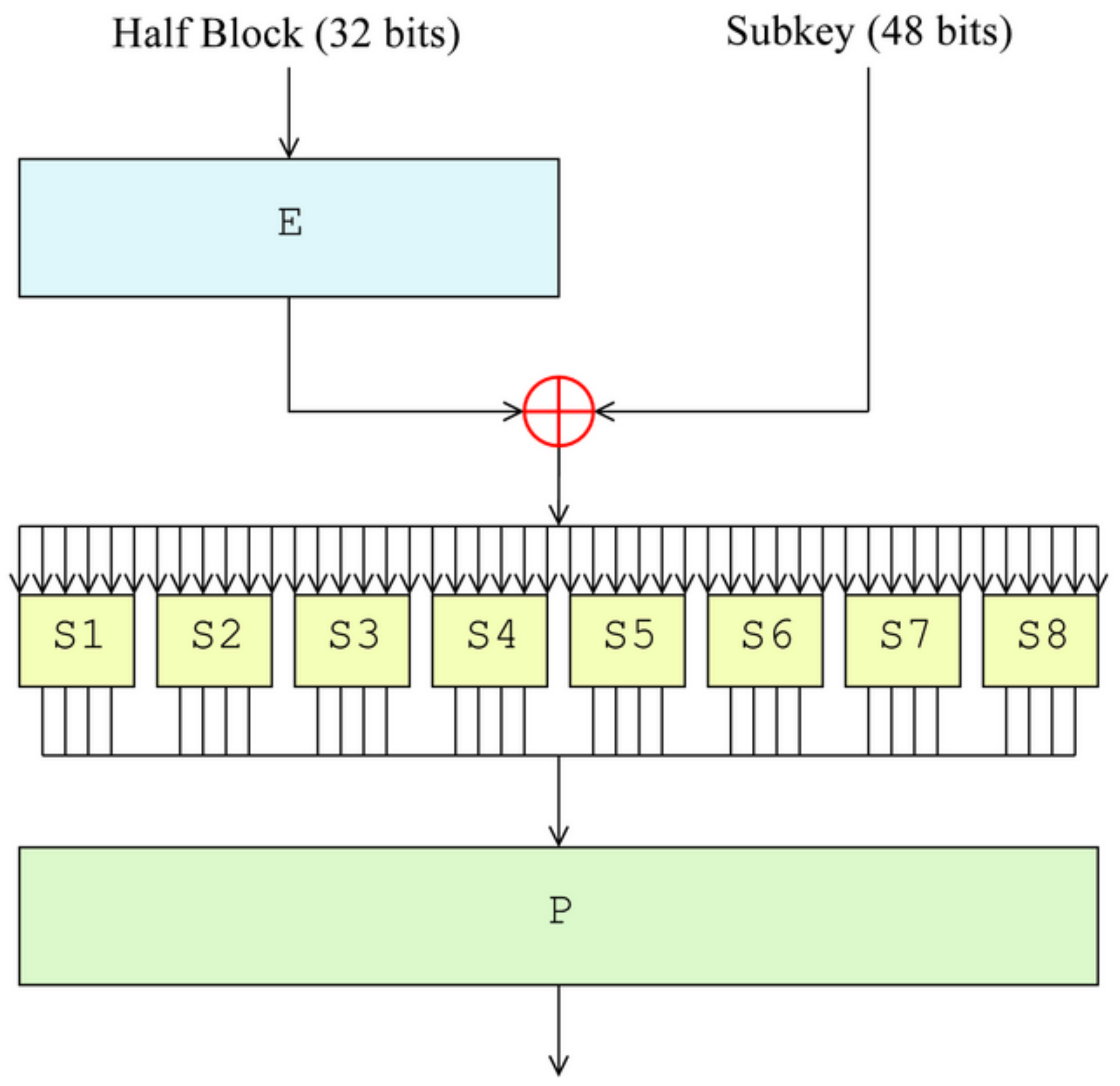
$$R_i = L_{i+1}$$
$$L_i = R_{i+1} \oplus \mathrm{F}(L_{i+1}, K_i).$$

Then (L_0, R_0) is the plaintext again.



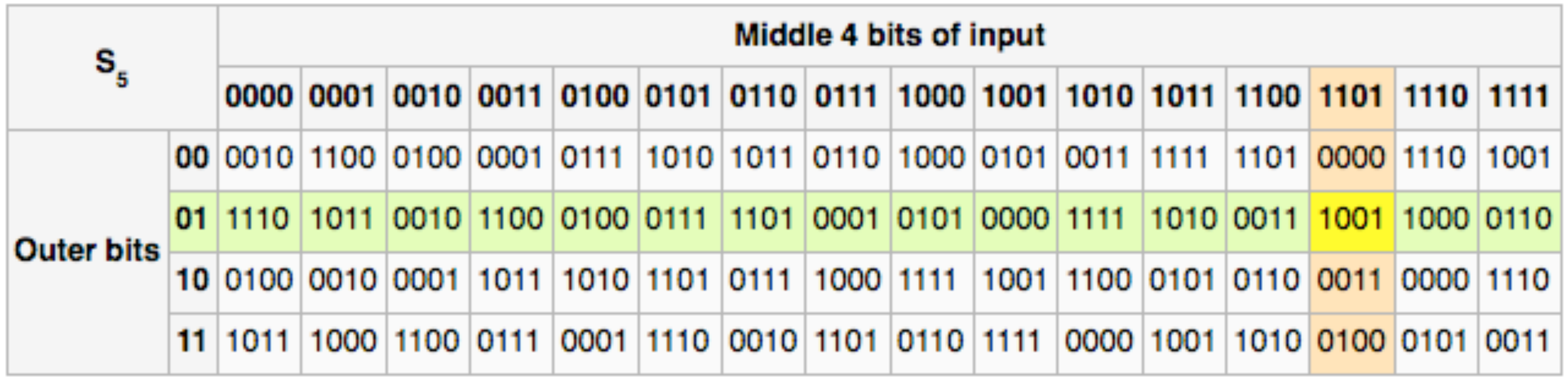
Data Encryption Standard (DES)

- Function F details
- E: Expansion from 32-bits to 48-bits via permutation
- XOR: with the round's subkey, which is also 48-bits
- Si: Substitution from 6-bit value to 4-bit value depending on S-box
- P: Permutation which spreads each S-box output across for



Substitution Box (S-box)

- A substitution box (or S-box) is used to obscure the relationship between the key and the ciphertext
  - Shannon's property of confusion: the relationship between key and ciphertext is as complex as possible.
  - In DES S-boxes are carefully chosen to resist cryptanalysis.
  - Thus, that is where part of the security comes from.



S_5		Middle 4 bits of input															
		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
Outer bits	00	0010	1100	0100	0001	0111	1010	1011	0110	1000	0101	0011	1111	1101	0000	1110	1001
	01	1110	1011	0010	1100	0100	0111	1101	0001	0101	0000	1111	1010	0011	1001	1000	0110
	10	0100	0010	0001	1011	1010	1101	0111	1000	1111	1001	1100	0101	0110	0011	0000	1110
	11	1011	1000	1100	0111	0001	1110	0010	1101	0110	1111	0000	1001	1010	0100	0101	0011

Example: Given a 6-bit input, the 4-bit output is found by selecting the row using the outer two bits, and the column using the inner four bits. For example, an input "011011" has outer bits "01" and inner bits "1101"; the corresponding output would be "1001".

Permutations Box (P-box)

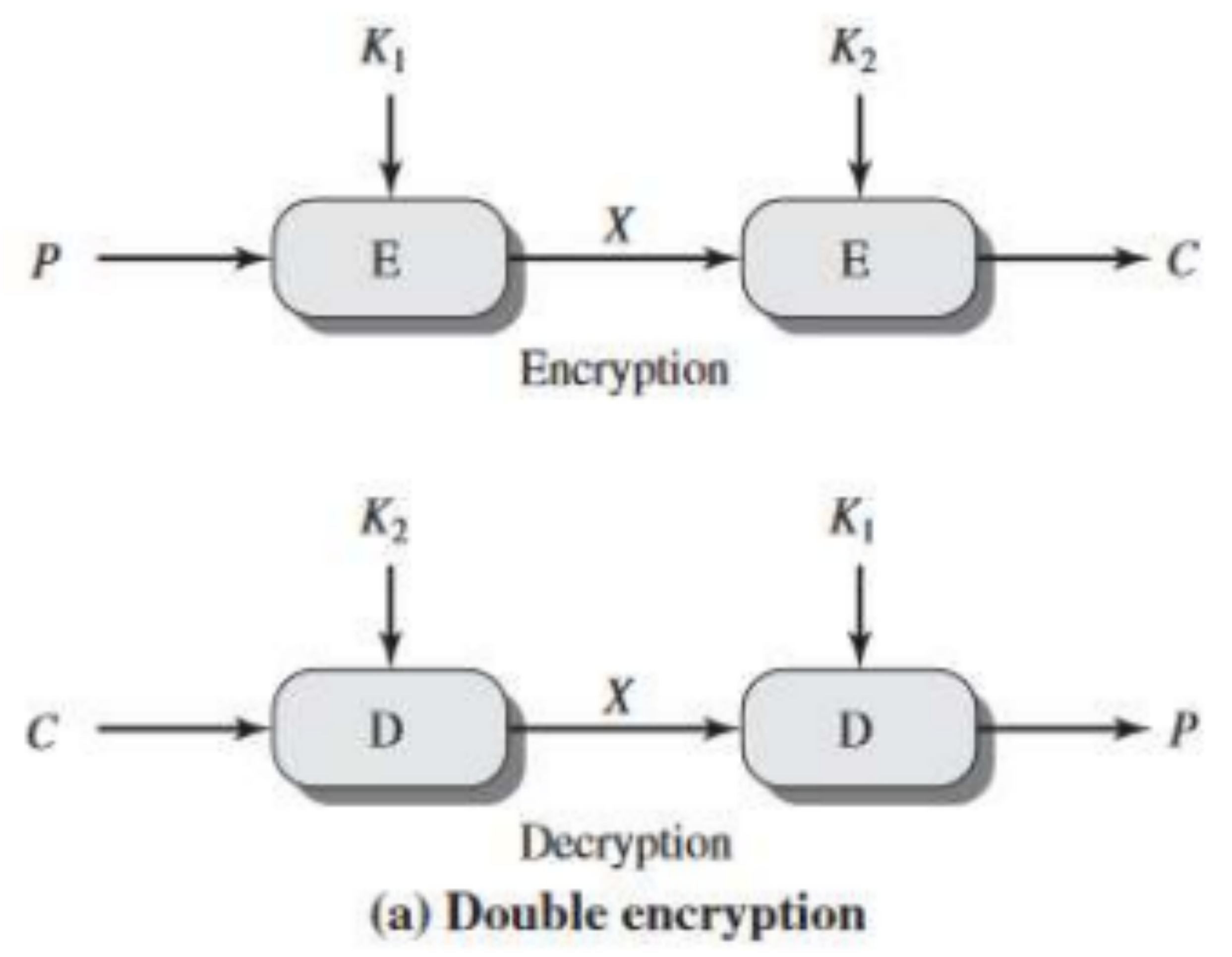
- A permutations box (or P-box) is used to obscure the relationship between the plaintext and the ciphertext
  - Shannon's property of diffusion: the relationship between plaintext and ciphertext is as complex as possible.
  - DES uses a combination of diffusion and confusion to resist cryptanalysis
  -

Cryptanalysis of DES

- DES has an effective 56-bit key length
  - Wiener: $1,000,000 - 3.5 hours (never built)
  - July 17, 1998, the EFF DES Cracker, which was built for less than $250,000 < 3 days
  - January 19, 1999, Distributed.Net (w/EFF), 22 hours and 15 minutes (over many machines)
  - We all assume that NSA and agencies like it around the world can crack (recover key) DES in milliseconds

Variants of DES

- Double DES (two keys = 112-bits)
  - Meet-in-the-Middle Attack
    - P->X (56 bits)
    - X->C (56 bits)
  - Complexity of cryptanalysis
    - 2^56 + 2^56 (worst case)

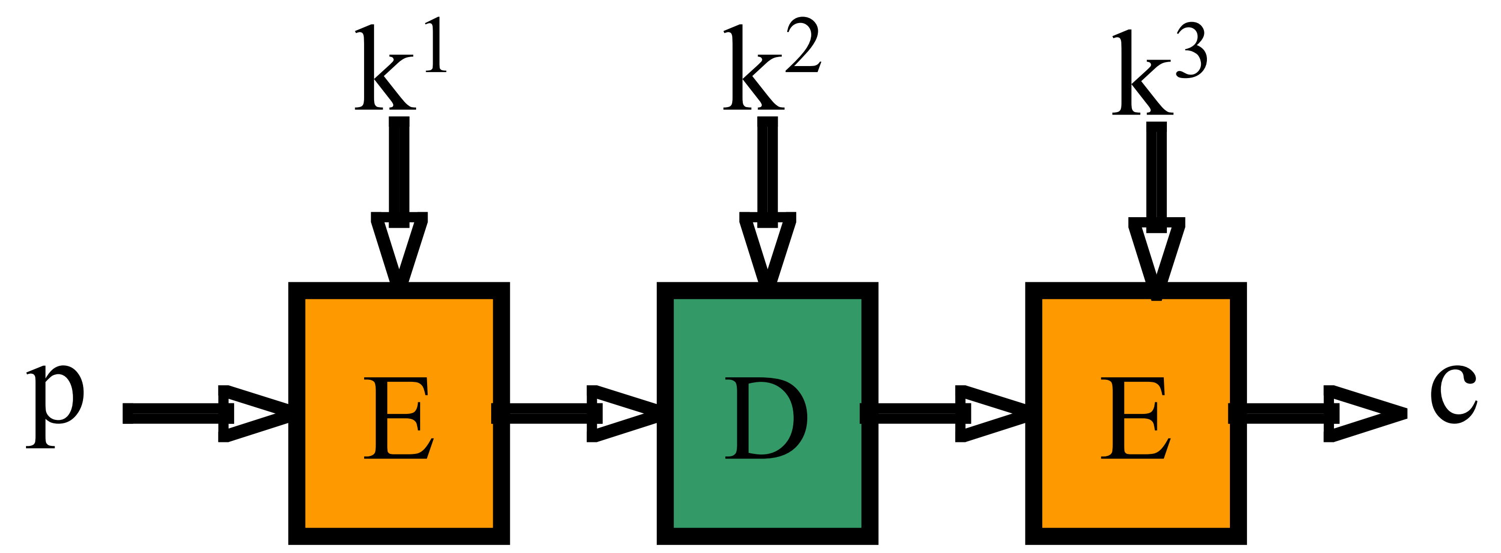


Double-DES

- Triple DES (three keys ~=112-bits)
  - keys

$$k_1, k_2, k_3$$

$$C = E(D(E(p, k_1), k_2, k_3)$$



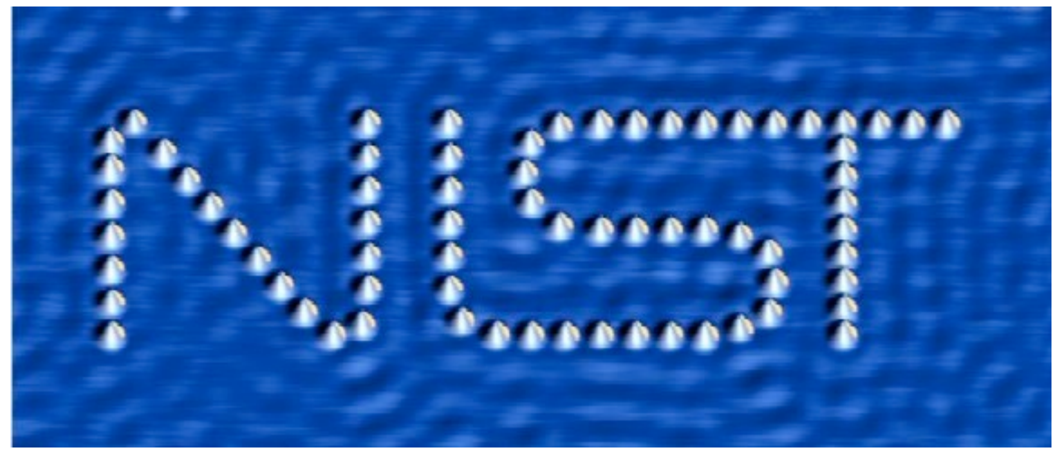
Key size and algorithm strength

- Key size is an oft-cited measure of the strength of an algorithm, but is strength strongly correlated (or perfectly correlated with key length)?
  - Say we have two algorithms, A and B with key sizes of 128 and 160 bits (the common measure)
  - Is A "less secure" than B?
  - What if A=B (for variable key-length algorithms)?

Implication: references to key length in advertisements are often meaningless.

Advanced Encryption Standard (AES)

- International NIST bakeoff between cryptographers
  - Rijndael (pronounced "Rhine-dall")



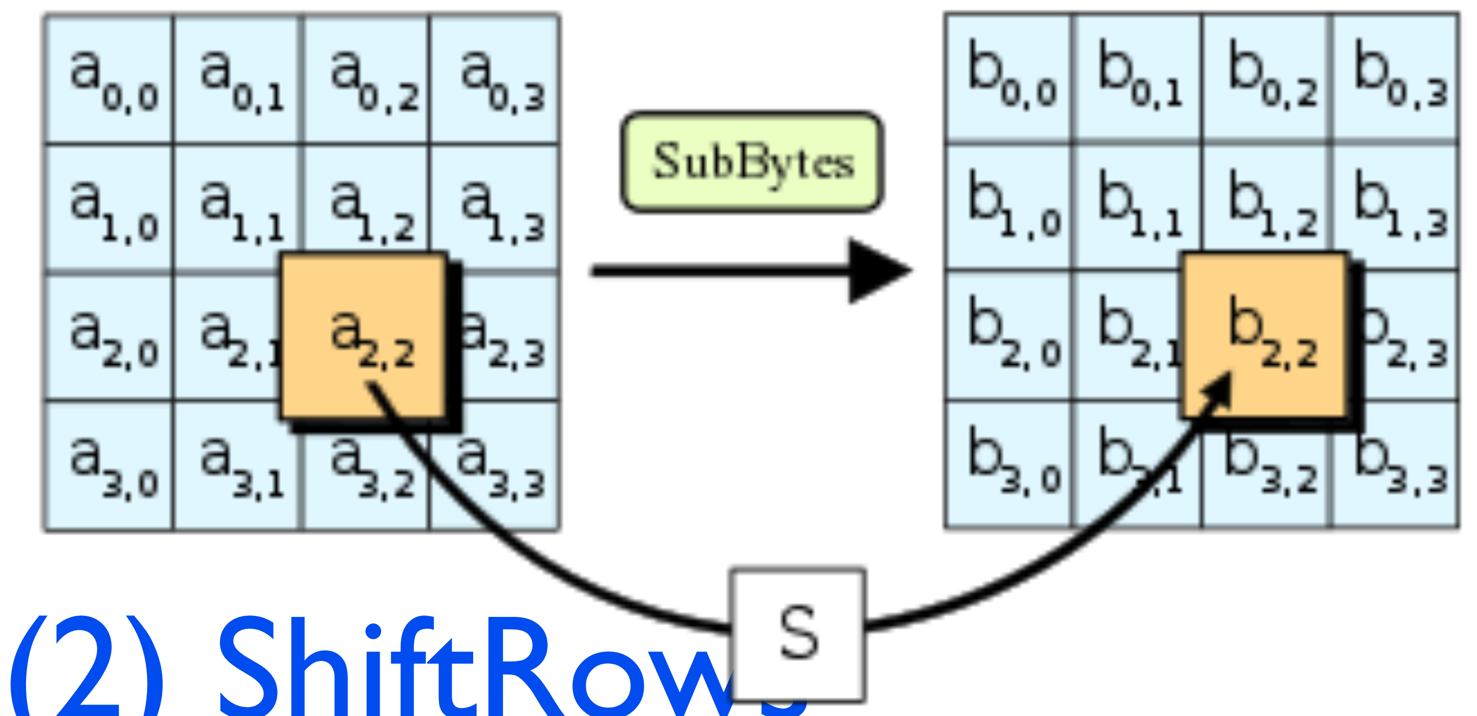
- Replacement for DES/accepted symmetric key cipher
  - Substitution-permutation network, not a Feistel network
  - Block size: 128 bits
  - Variable key lengths: 128, 192, or 256 bits
  - Fast implementation in hardware and software
  - Small code and memory footprint: No known exploitable algorithmic weaknesses
  - Implementation may be vulnerable to timing attacks
  - Intel has AES-NI, CPU-based implementation for AES
-

Advanced Encryption Standard (AES)

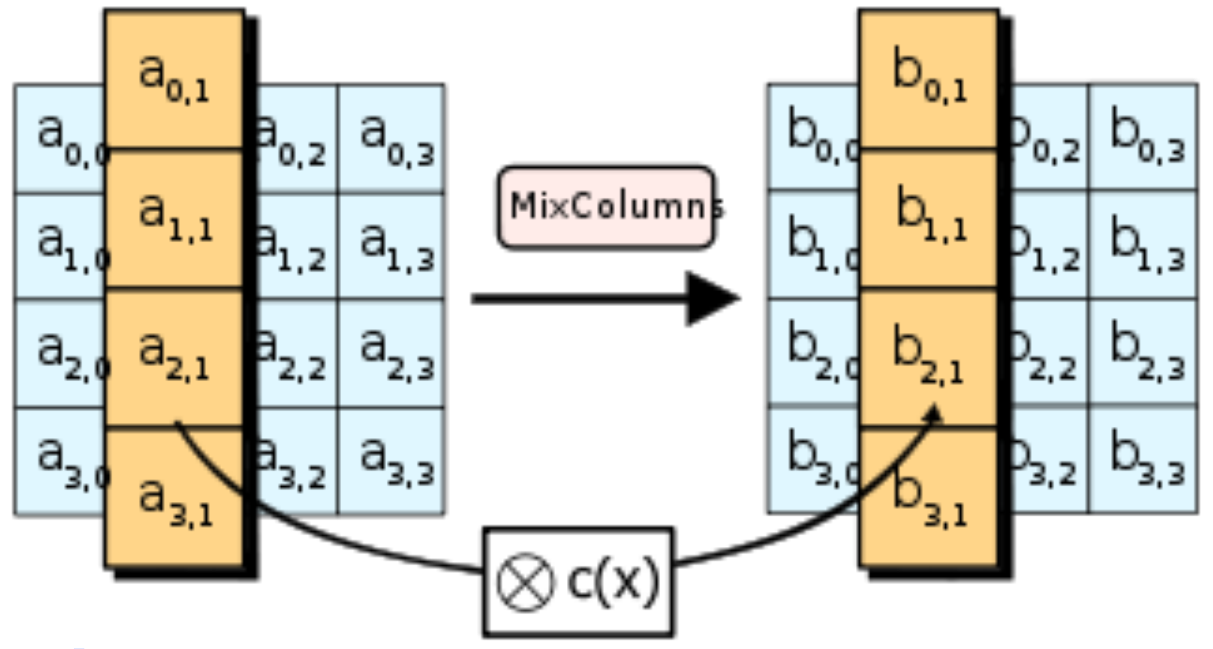
- Replace 3DES basically
- With something fast and flexible
- And secure against attacks for a while into the future
- Takes a block of the plaintext and the key as inputs and applies several alternating "rounds" or "layers" of substitution boxes (S-boxes) and permutation boxes (P-boxes) to produce the ciphertext block
- Basic Steps
  - Key expansion - derive keys for each round
  - Initial key addition - combine block with round key via XOR
  - Perform round operation (9, 11, 13 times) - magic here
  - Final round - similar to round operation except does not use the "MixColumn" operation

Advanced Encryption Standard (AES)

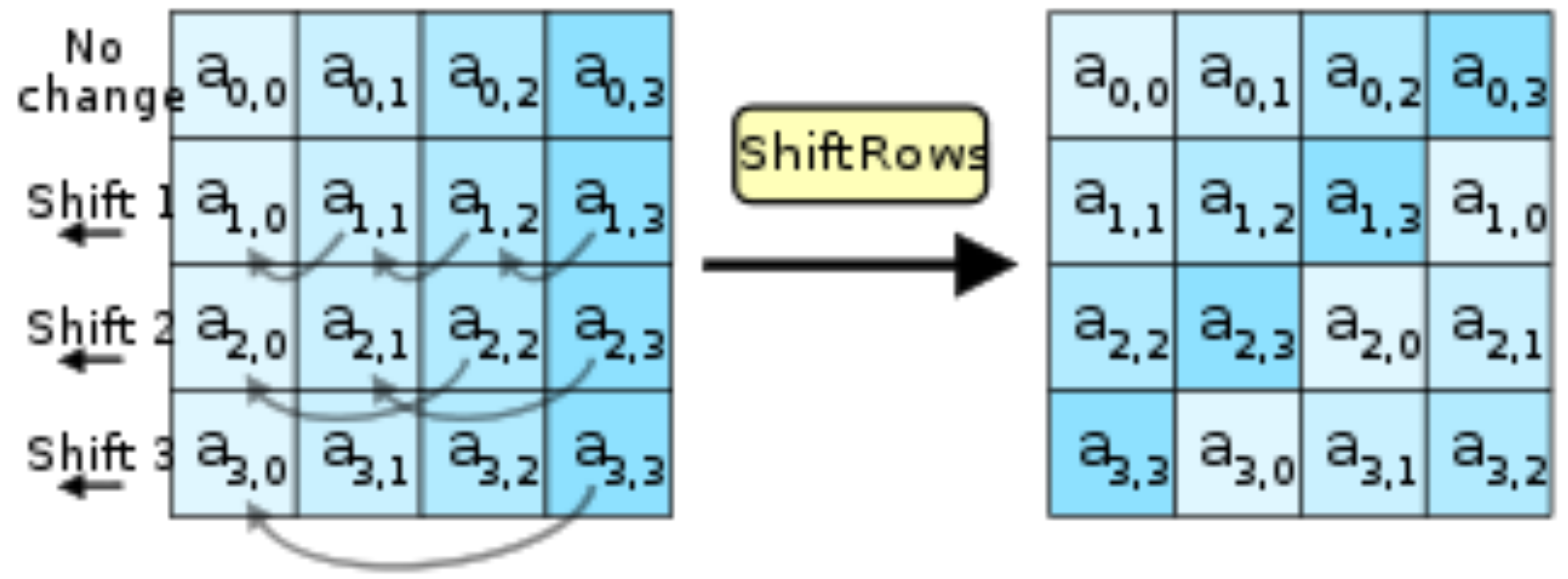
- Magic step - Round Operations
- (1) Substitute Bytes



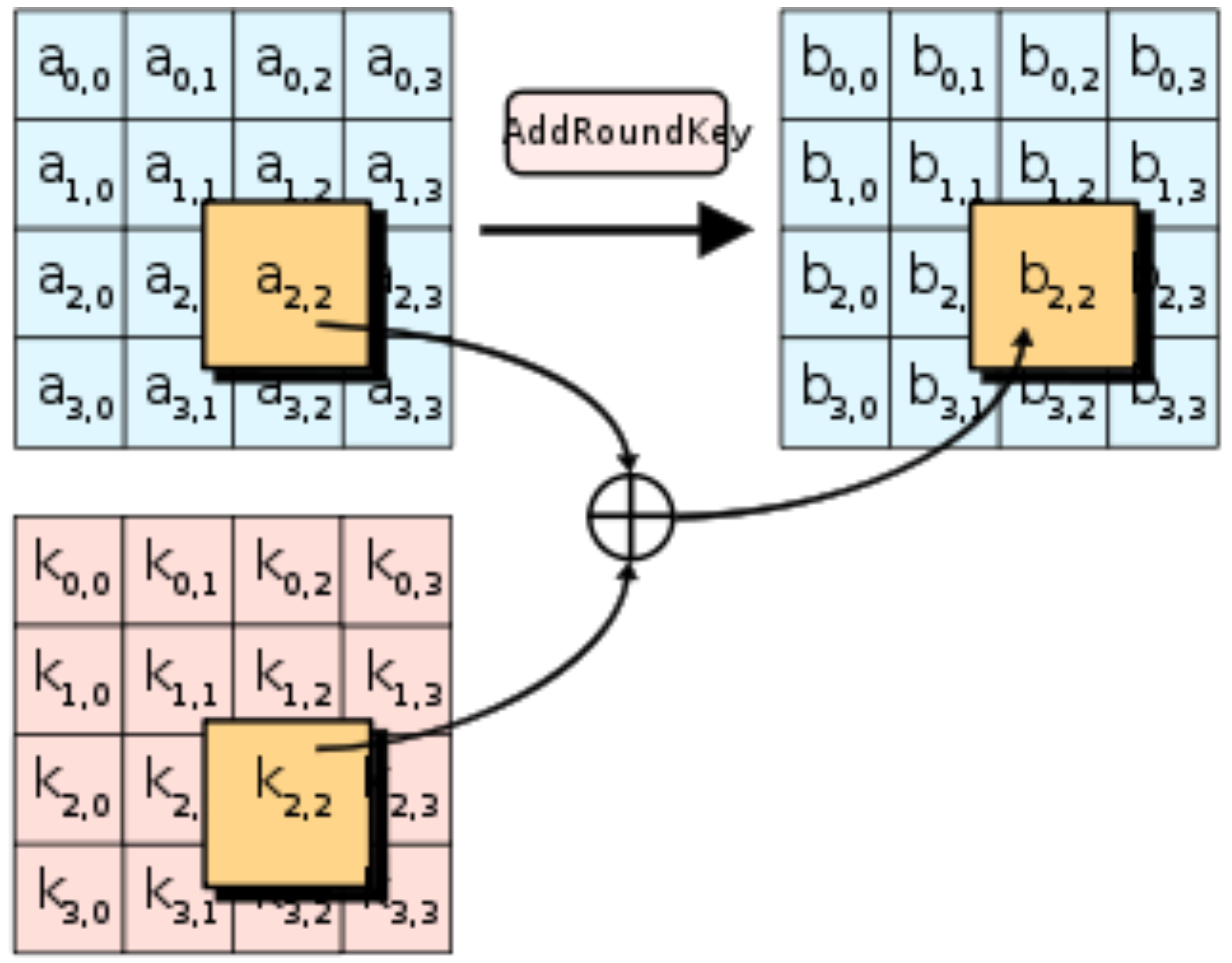
(3) MixColumns



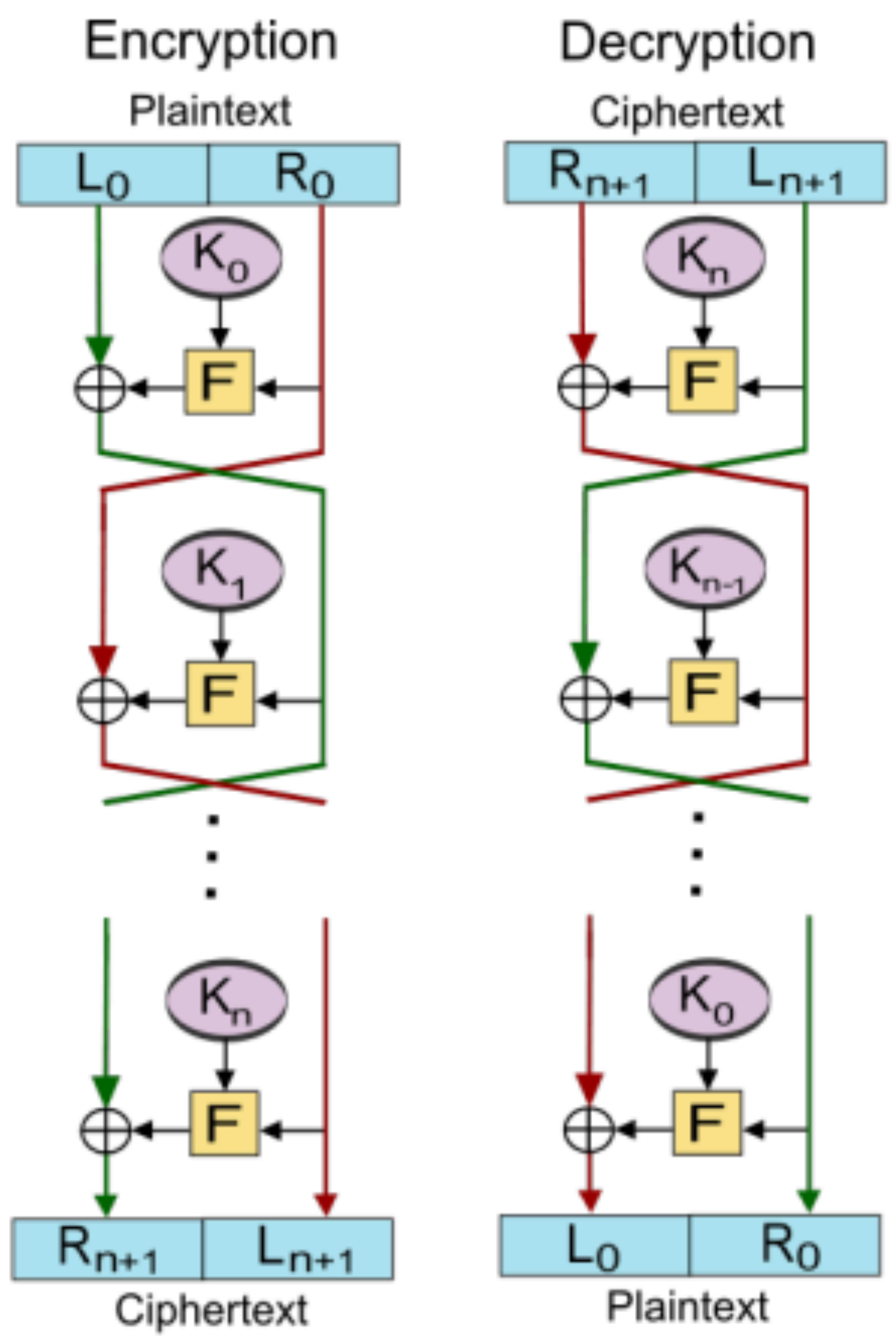
- (2) ShiftRows

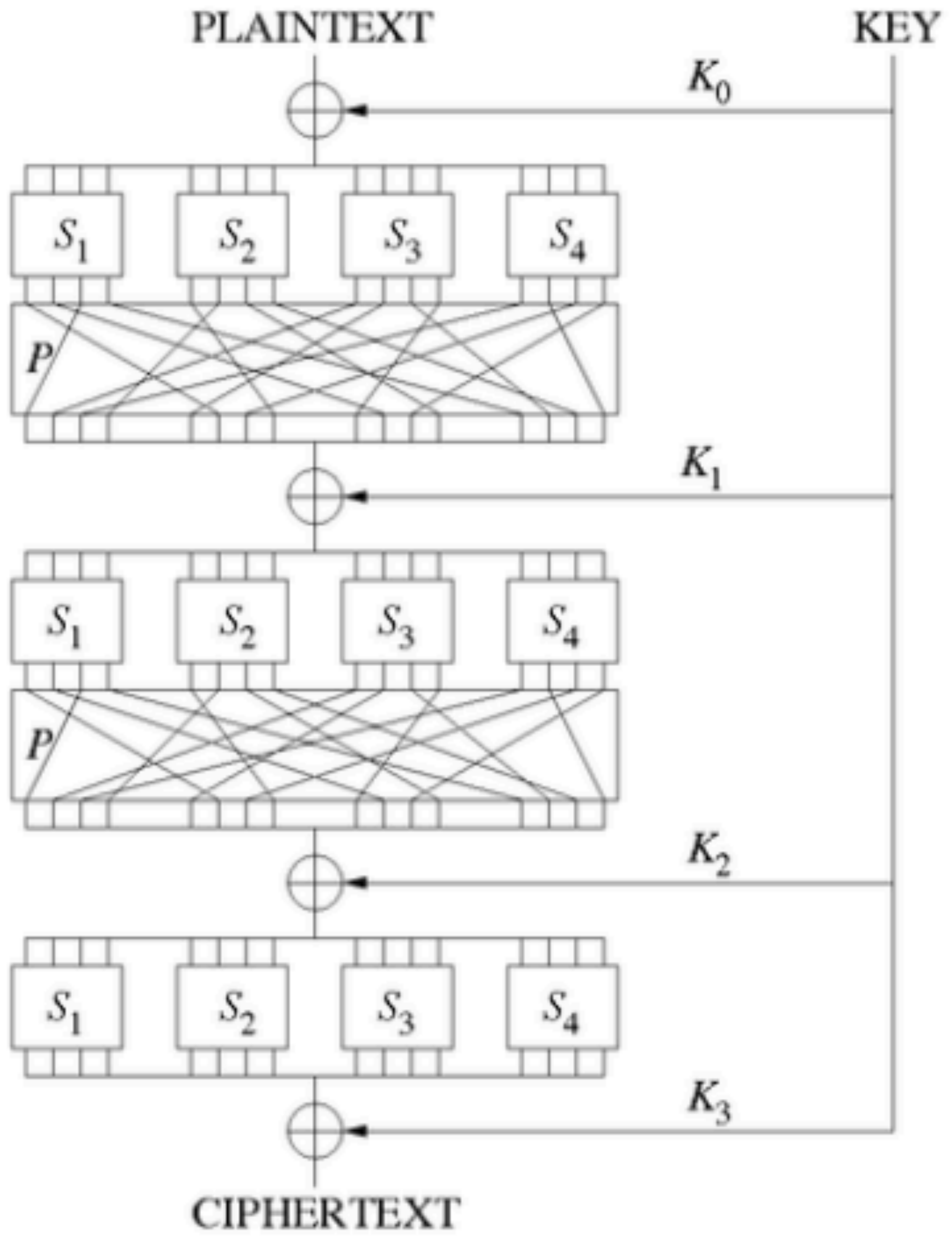


(4) AddRoundKey



Fiestal vs. SP Network





AES Implementation Aspects

- AES can be implemented very efficiently on an 8-bit processor
- AddRoundKey is a bytewise XOR operation
- ShiftRows is a simple byte-shifting operation
- SubBytes operates at the byte level and only requires a table of 256 bytes
- MixColumns requires matrix multiplication in the field $GF(2^8)$, whichmeans all operations are carried out on bytes
- Designers believe this very efficient implementation was a key factor in its selection as the AES cipher

Attacking a Cipher

- The attack mounted will depend on what information is available to the adversary
  - Ciphertext-only attack: adversary only has the ciphertext available and wants to determine the plaintext
  - Known-plaintext attack: adversary learns one or more pairs of ciphertext/plaintext encrypted under the same key, tries to determine plaintext from a different ciphertext
  - Chosen-plaintext attack: adversary can obtain the encryption of any plaintext, tries to determine the plaintext for a different ciphertext
  - Chosen-ciphertext attack: adversary can obtain the plaintext of any ciphertext except the one the adversary wants to decrypt

Known-Plaintext Attack

- Known-plaintext attack: adversary learns one or more pairs of ciphertext/ plaintext encrypted under the same key, tries to determine plaintext based on a different ciphertext
  - Suppose that the adversary knows common messages
    - "Calling all cars"
  - When these messages are encrypted the adversary may use them to extract the key material
    - "Xwggdib wgg xwmn"
- As a result, we will see that cryptographers designed cryptographic "modes" to prevent such detection

Need for Encryption Mode

PennState

- A block cipher encrypts only one block
- Needs a way to extend it to encrypt an arbitrarily long message
- Want to ensure that if the block cipher is secure, then the encryption is secure
- Aims at providing Semantic Security (IND-CPA) assuming that the underlying block ciphers are strong

Block Cipher

- Block ciphers are fantastic tools for encrypting n-bit length messages
  - What if I want to encrypt more than n bits?

- If too short: *pad the input using an established standard*
  - Padding is meaningless data added to the end of a message to round out to n bits
  - Example: Always add a "1", Then as many "0"s as necessary

- If too long: *break the message up into n-bit blocks and pass parts of the message* to the cipher block by block (pad last block if necessary)
  - Several ways of doing this – each is called a "block cipher mode"

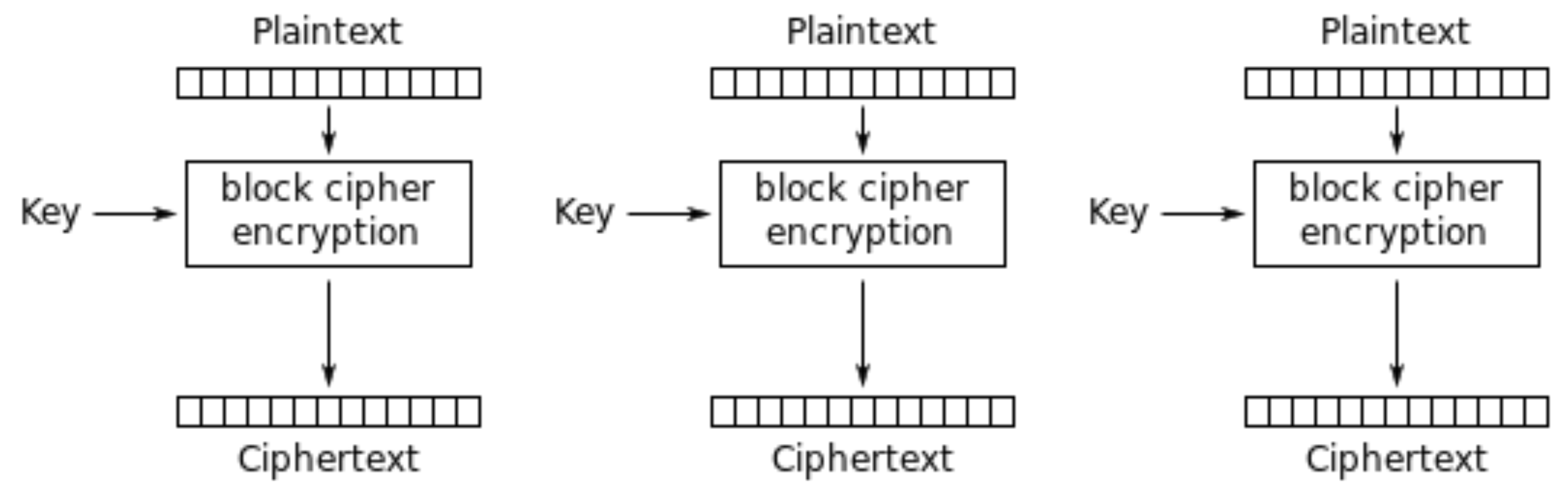
Symmetric Ciphers and Attacks

- Problem: Same plaintext encrypts to same cipher text
  - E(d, k) = c for each d and k
- Why does this happen?
- What can you do?

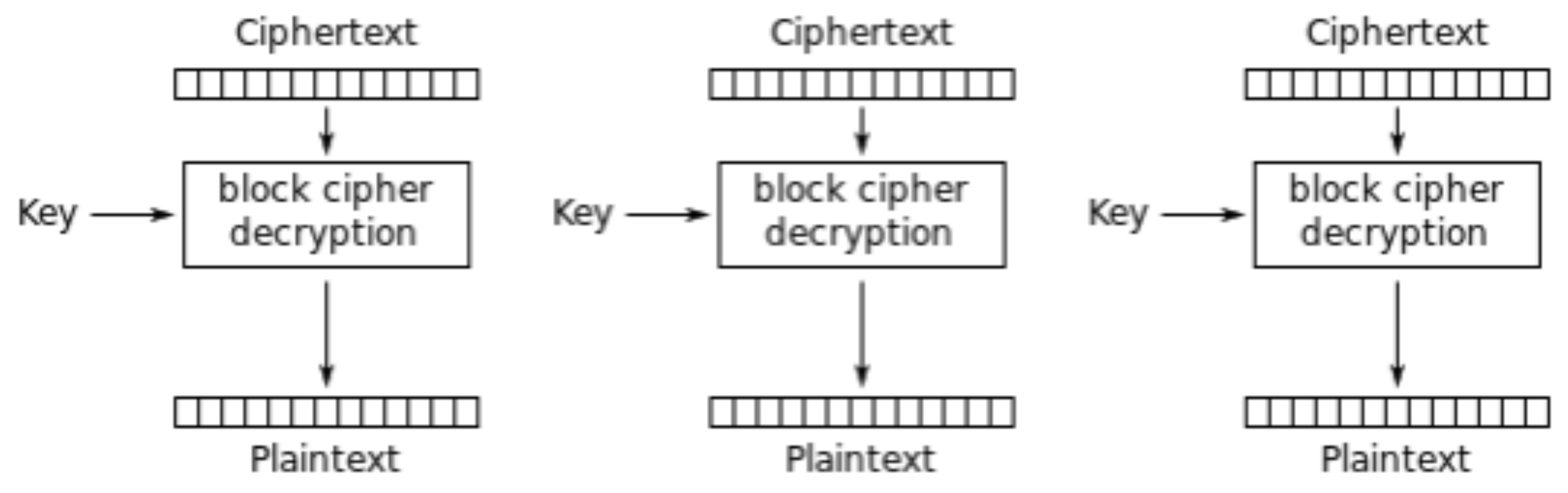
Symmetric Ciphers and Attacks

- Add a salt to the encryption process (like for passwords)
  - Initialization vector
  - Propagate using ciphertext for subsequent blocks
- Cipher modes
  - ECB (Electronic Code Book)
  - CBC (Cipher Block Chaining)
  - OFB (Output FeedBack)
  - CTR (Counter Mode)

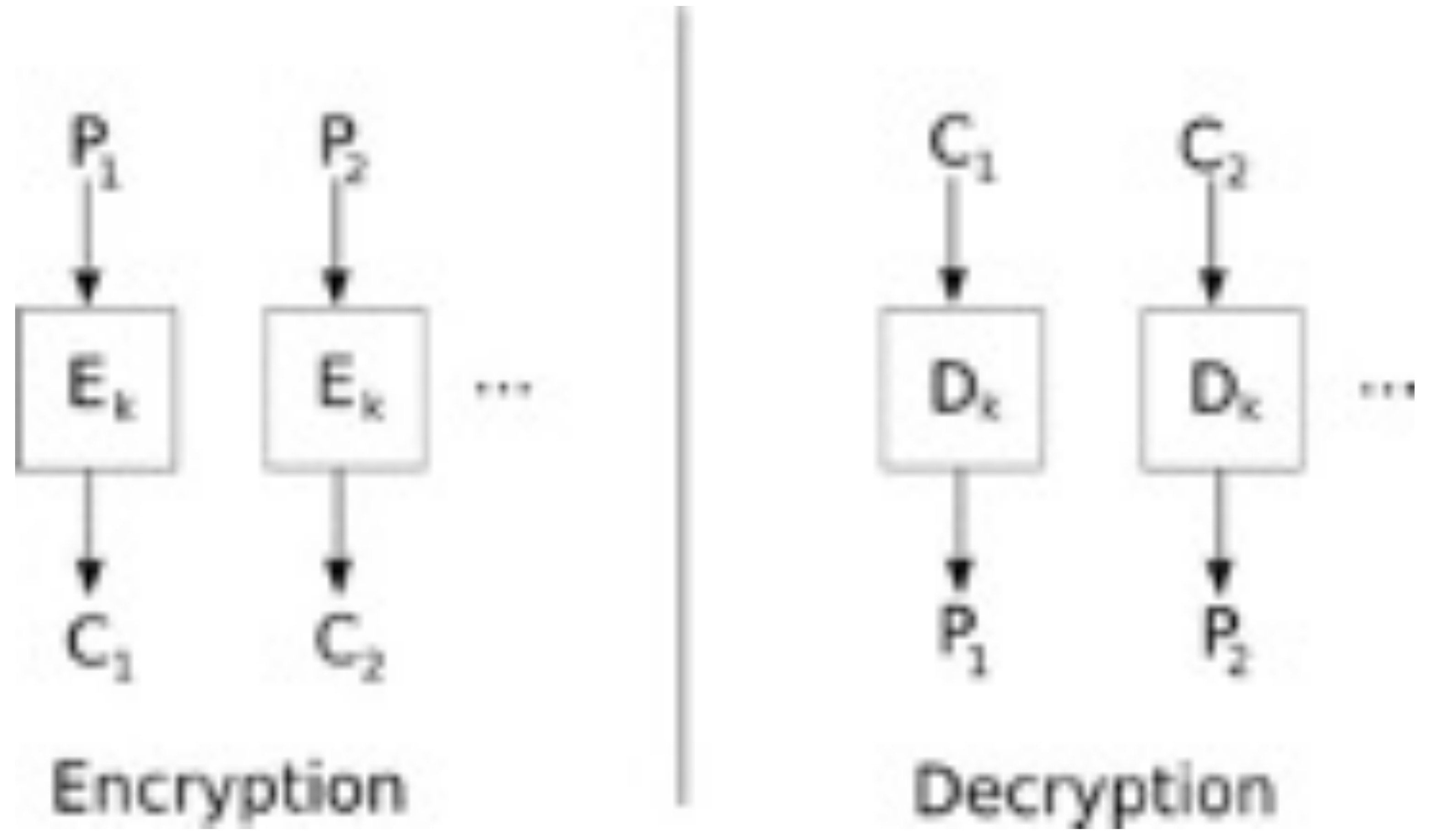
Electronic Code Book (ECB)

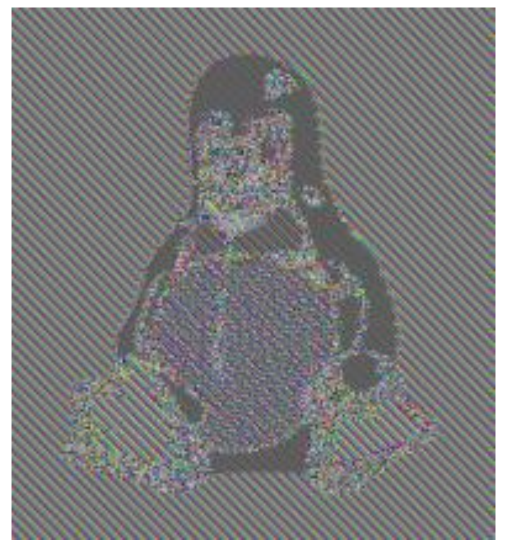


Electronic Codebook (ECB) mode encryption



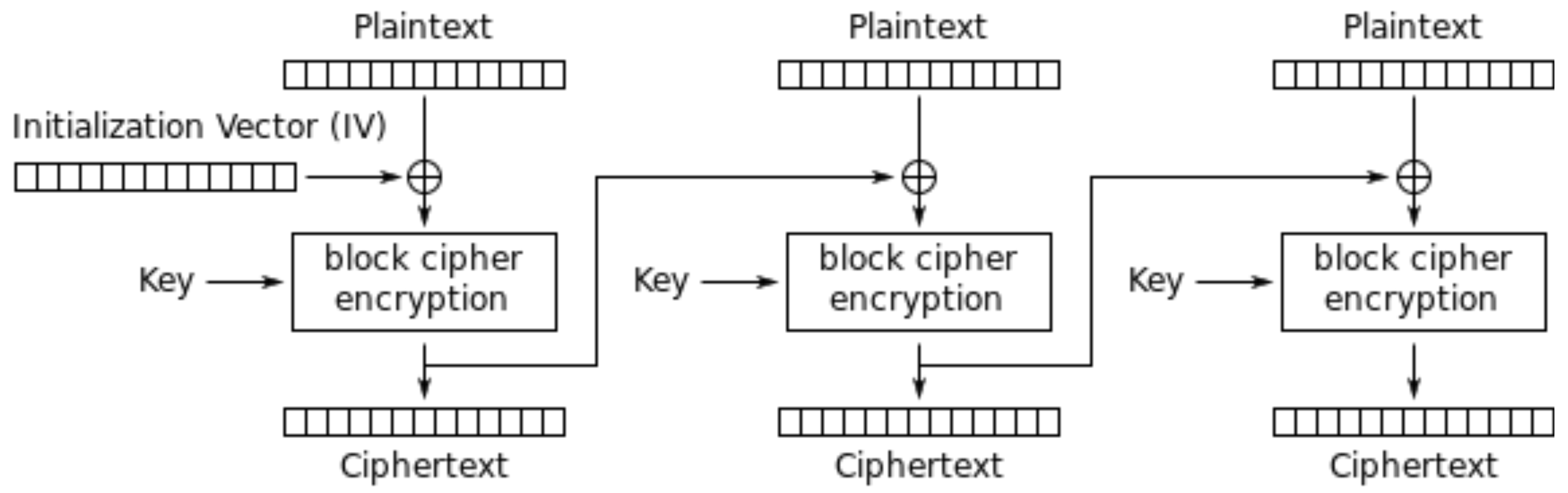
Electronic Codebook (ECB) mode decryption



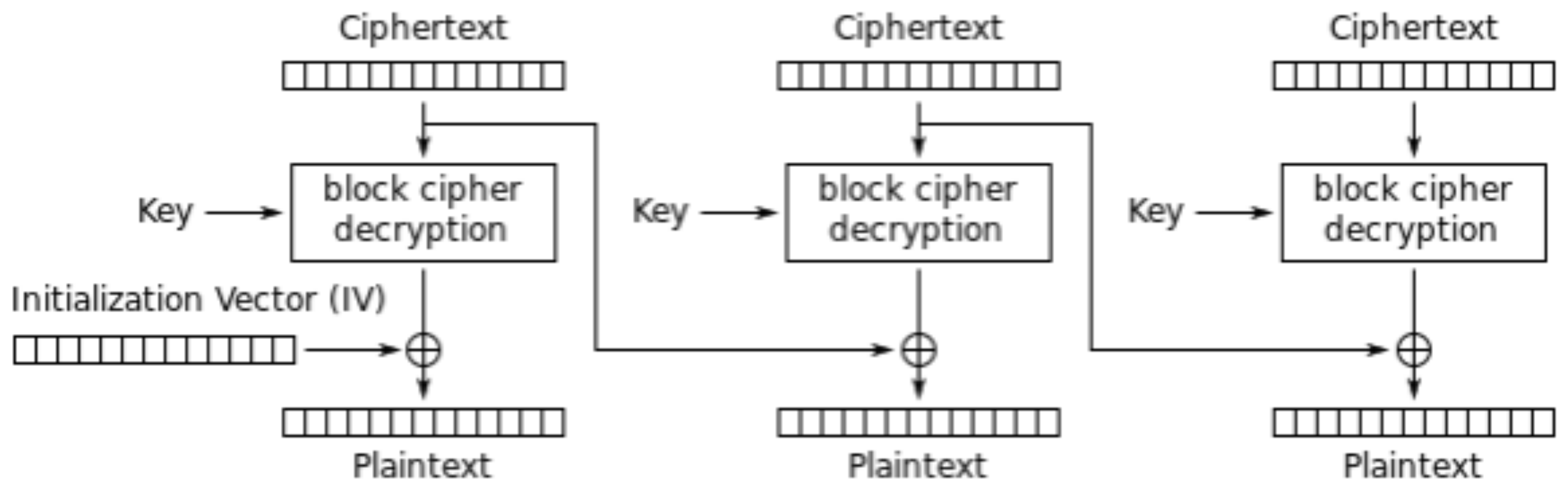


Cipher Block Chaining (CBC)

- Adds in a *random* initialization vector (IV) to randomize ciphertext appearance.
  - IV isn't secret! It can be sent in the clear
- Chains together by XOR-ing the previous ciphertext with the current plaintext block



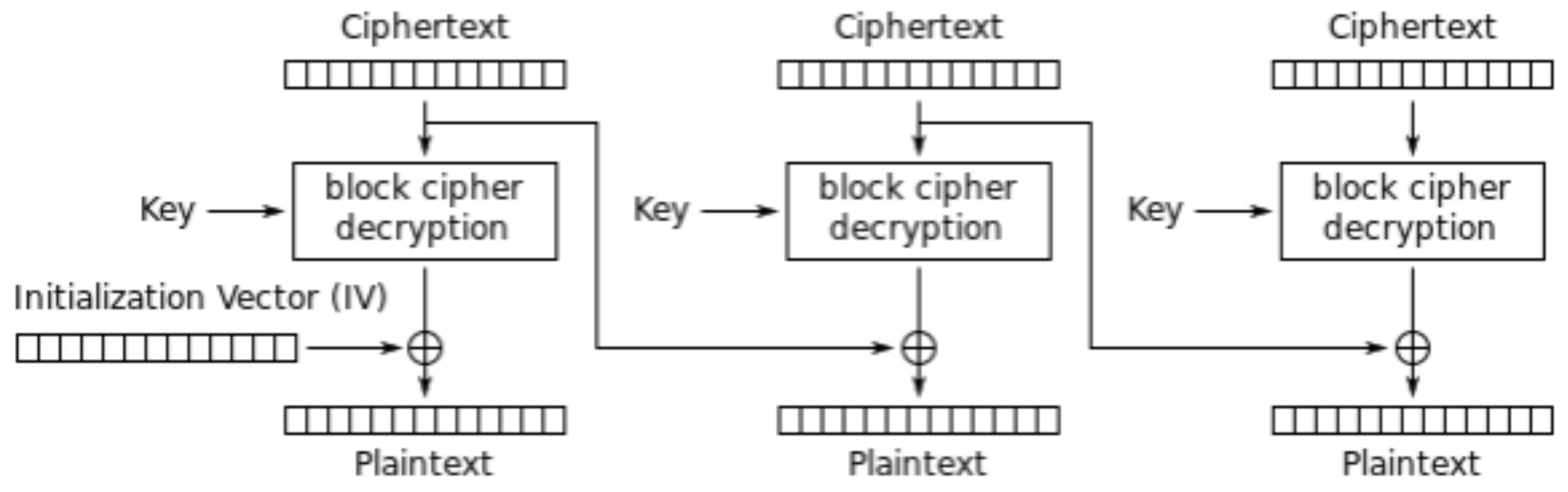
Cipher Block Chaining (CBC) mode encryption



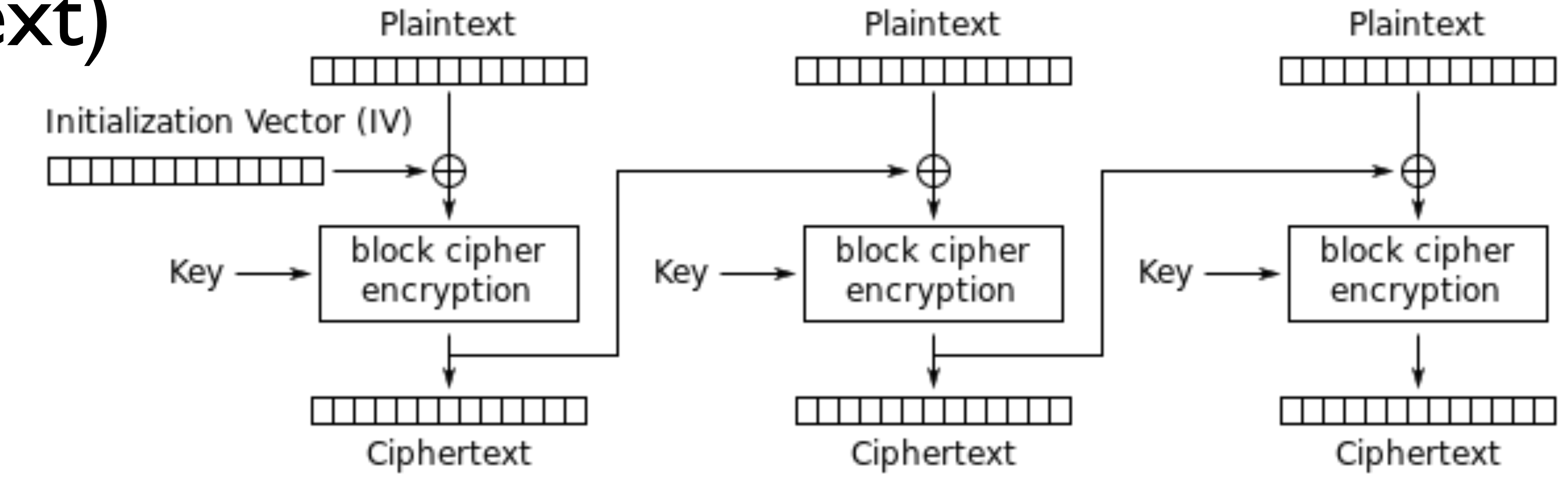
Cipher Block Chaining (CBC) mode decryption

CBC

- Randomized encryption: repeated text gets mapped to different encrypted data.
  - can be proven to provide IND-CPA assuming that the block cipher is secure (i.e., it is a Pseudo Random Permutation (PRP)) and that IV's are randomly chosen and the IV space is large enough (at least 64 bits)
- Each ciphertext block depends on all preceding plaintext blocks.
- Usage: chooses random IV and protects the integrity of IV
  - The IV is not secret (it is part of ciphertext)



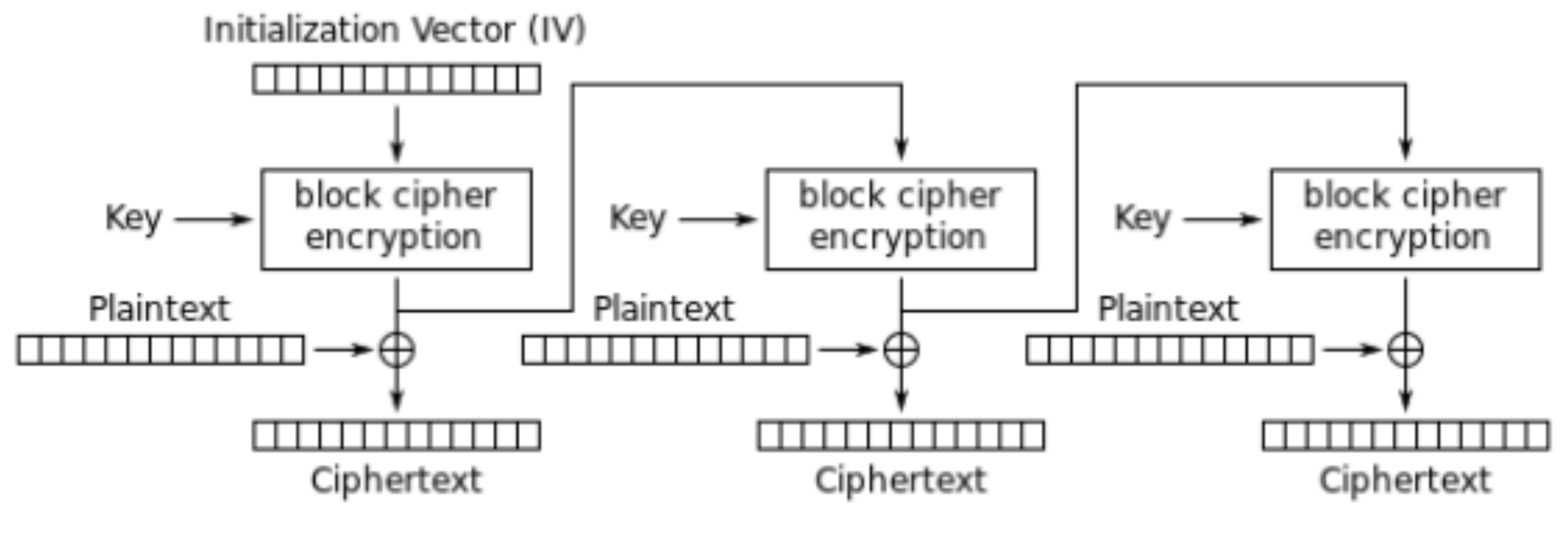
Cipher Block Chaining (CBC) mode decryption



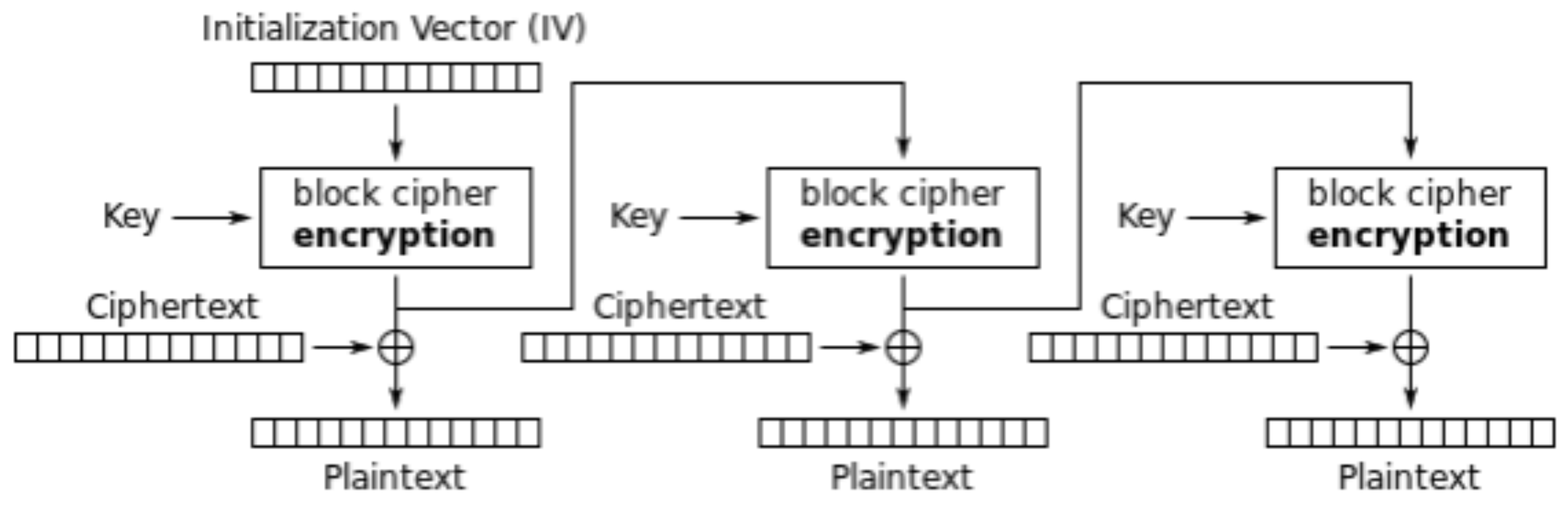
Cipher Block Chaining (CBC) mode encryption

Output Feedback (OFB)

- Turns a block cipher into a synchronous stream cipher



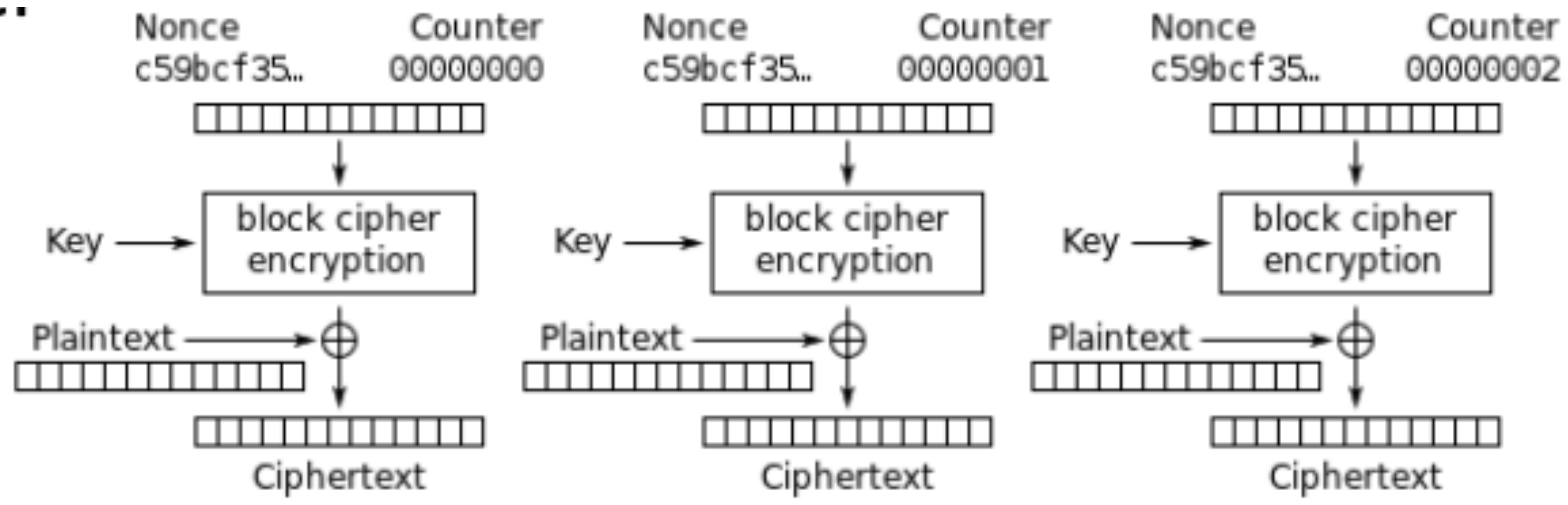
Output Feedback (OFB) mode encryption



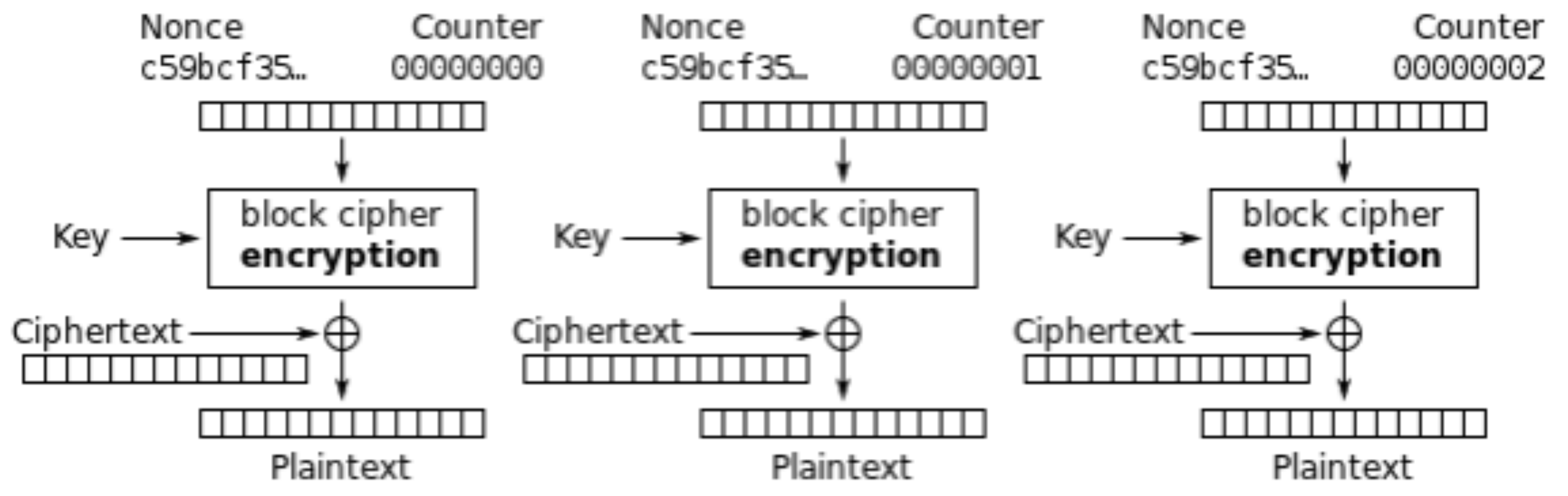
Output Feedback (OFB) mode decryption

Counter Mode (CTR)

- Turns a block cipher into a stream cipher where keystream blocks are created by encrypting successive values of a "counter"
- Properties of CTR
  - Gives a stream cipher from a block cipher
  - Randomized encryption:
    - when starting counter is chosen randomly
  - Random Access: encryption and decryption of a block can be done in random order, very useful for hard-disk encryption.
    - E.g., when one block changes, re-encryption only needs to encrypt that block. In CBC, all later blocks also need to change.



Counter (CTR) mode encryption



Counter (CTR) mode decryption

$$c1 = m1 \text{ xor } k$$

$$c2 = m2 \text{ xor } k$$

$$c1 \text{ xor } c2 = m1 \text{ xor } m2$$

- *Information leakage*: Does it reveal info about the plaintext blocks?
- *Ciphertext manipulation*: Can an attacker modify ciphertext block(s) in a way that will produce a predictable/desired change in the decrypted plaintext block(s)?
  - Note: assume the structure of the plaintext is known, e.g., first block is employee #1 salary, second block is employee #2 salary, etc.
- *Parallel/Sequential*: Can blocks of plaintext (ciphertext) be encrypted (decrypted) in parallel?
- *Error Propagation*: If there is an error in a plaintext (ciphertext) block, will there be an encryption (decryption) error in more than one ciphertext (plaintext) block?

Answer The Following

	ECB	CBC	OFB	CTR
Information Leakage	?	?	?	?
Ciphertext Manipulation	?	?	?	?
Parallel	?	?	?	?
Error Propagation	?	?	?	?
Random Access	?	?	?	?

	ECB	CBC	OFB	CTR
Information Leakage	Yes	No	No	No
Ciphertext Manipulation	Yes	Yes	Yes	Yes
Parallel	Yes	No (enc) Yes (dec)	Yes (enc) Yes (dec)	Yes
Error Propagation	No	No (enc) a little (dec)	No	No
Random Access	Yes	No	No	Yes

Building systems with cryptography

- Use quality libraries
  - E.g., OpenSSL, Libgcrypt, Cryptlib, BouncyCastle (Java, (
  - Find out what cryptographers think of a package befor
- Code review like crazy
- Educate yourself on how to use libraries
  - Caveats by original designer and programmer

Common issues that lead to pitfalls

- Generating randomness
- Storage of secret keys
- Virtual memory (pages secrets onto disk)
- Protocol interactions
- Poor user interface
- Poor choice of key length, prime length, using parameters from one algorithm in another