PennState

CSE 543: Computer Security

Module: Cryptography

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

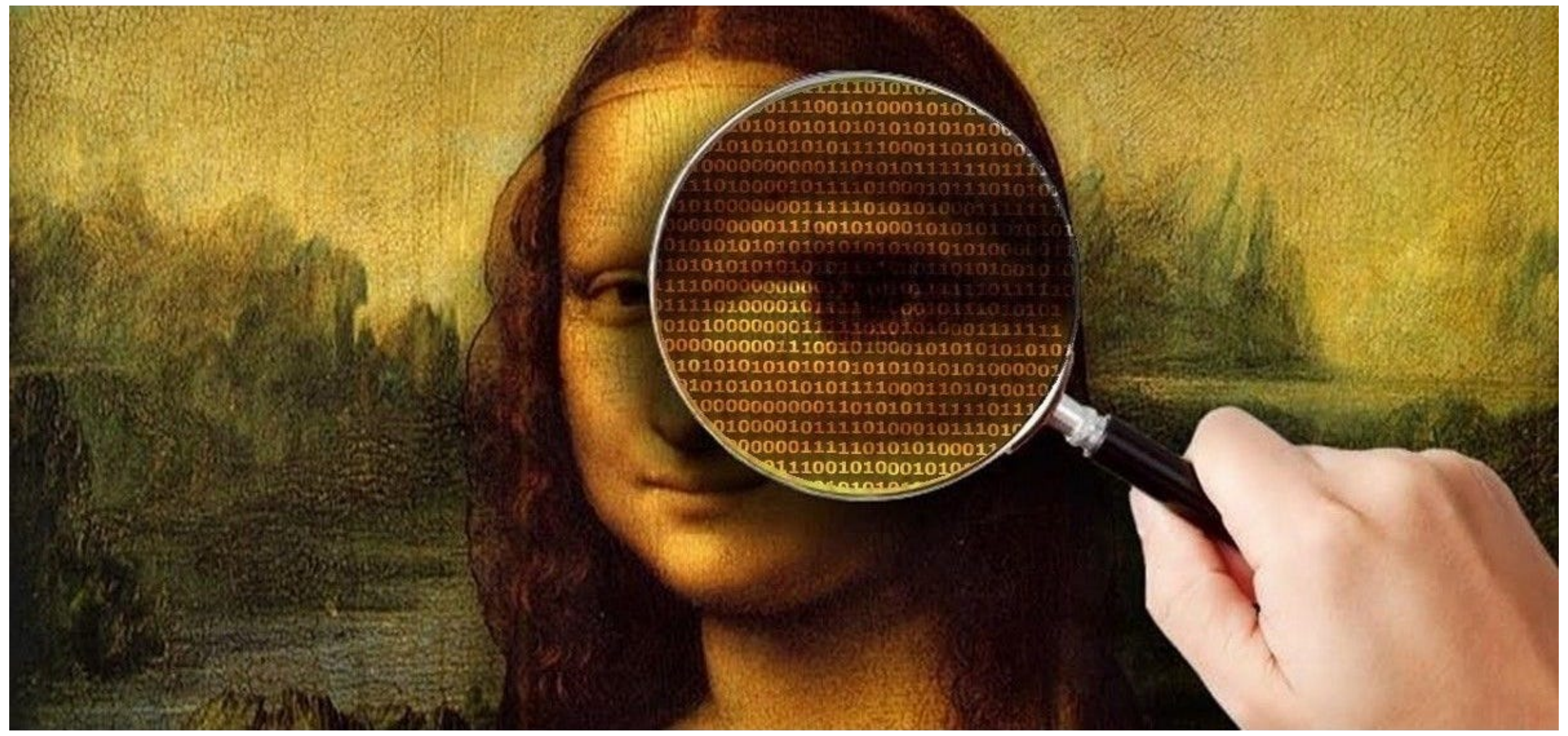
Goals of Cryptography

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

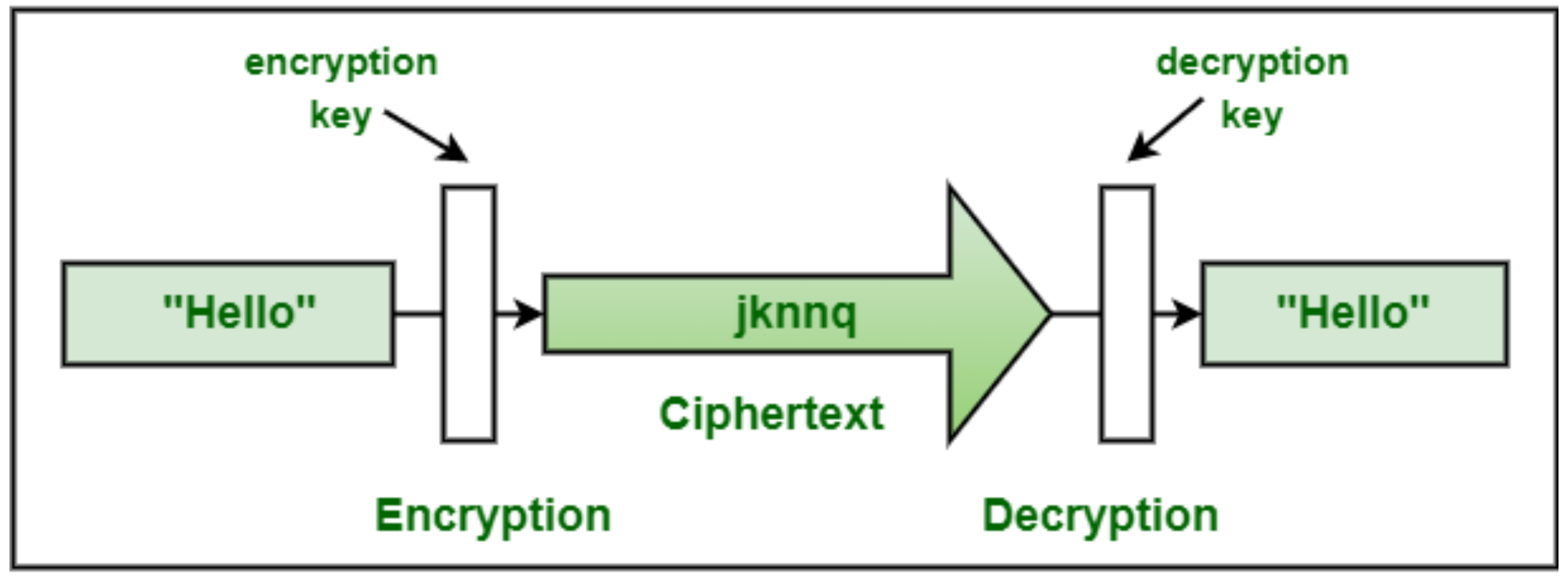
- What does secure communication mean?
  - confidentiality (privacy, secrecy)
    - only the intended recipient can see the communication
  - integrity (authenticity)
    - the communication is generated by the alleged sender
- What does insecure medium mean?
  - Two possibilities:
    - Passive attacker: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

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

Encryption algorithm

PennState

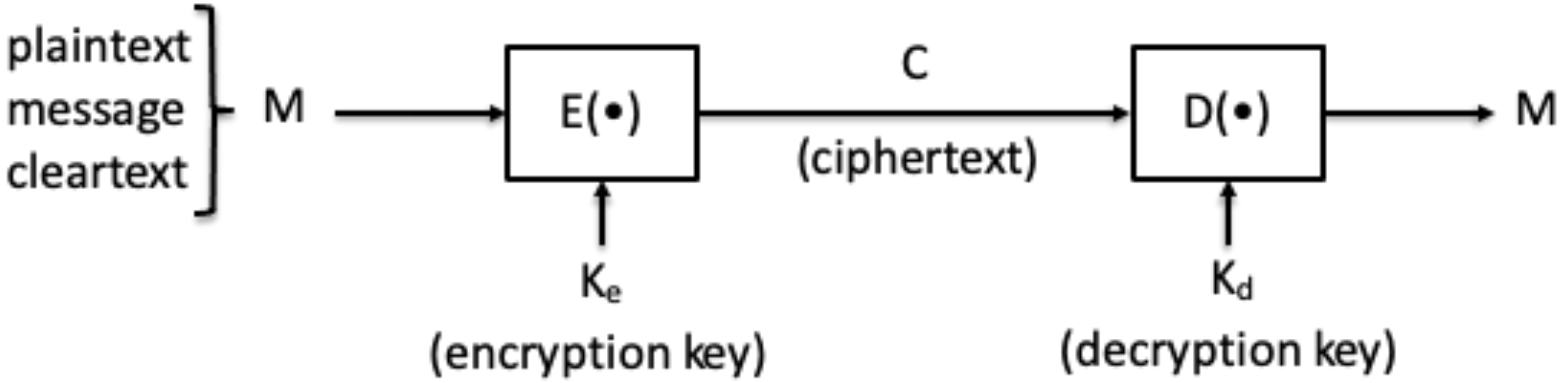
- Algorithm used to make content unreadable by all but the intended receivers

E(plaintext, key) = ciphertext

D(ciphertext, key) = plaintext

- *Algorithm is public, key is private*

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?

Hardness

- Inputs
  - Plaintext P
  - Ciphertext C
  - Encryption key k_e
  - Decryption key k_d

$$D(E(P, k_e), k_d) = P$$

- Computing P from C is hard, P from C with k_d is easy
  - for all Ps with more than negligible probability
  - This is known as a TRAPDOOR function
    - $y = f_k(x)$ is easy, but $x = f_k^{-1}(y)$ infeasible if y is known and k is unknown.
  - Devil is in the details ....

Example: Caesar Cipher

- Substitution 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

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

1	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

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

A	B	C	D	E	F	G	H	I	J	K	L	M
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
C	M	T	E	F	H	P	U	D	X	N	Z	L

N	O	P	Q	R	S	T	U	V	W	X	Y	Z
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
O	A	J	R	Y	I	G	W	V	B	S	Q	K

Strength of General Substitution

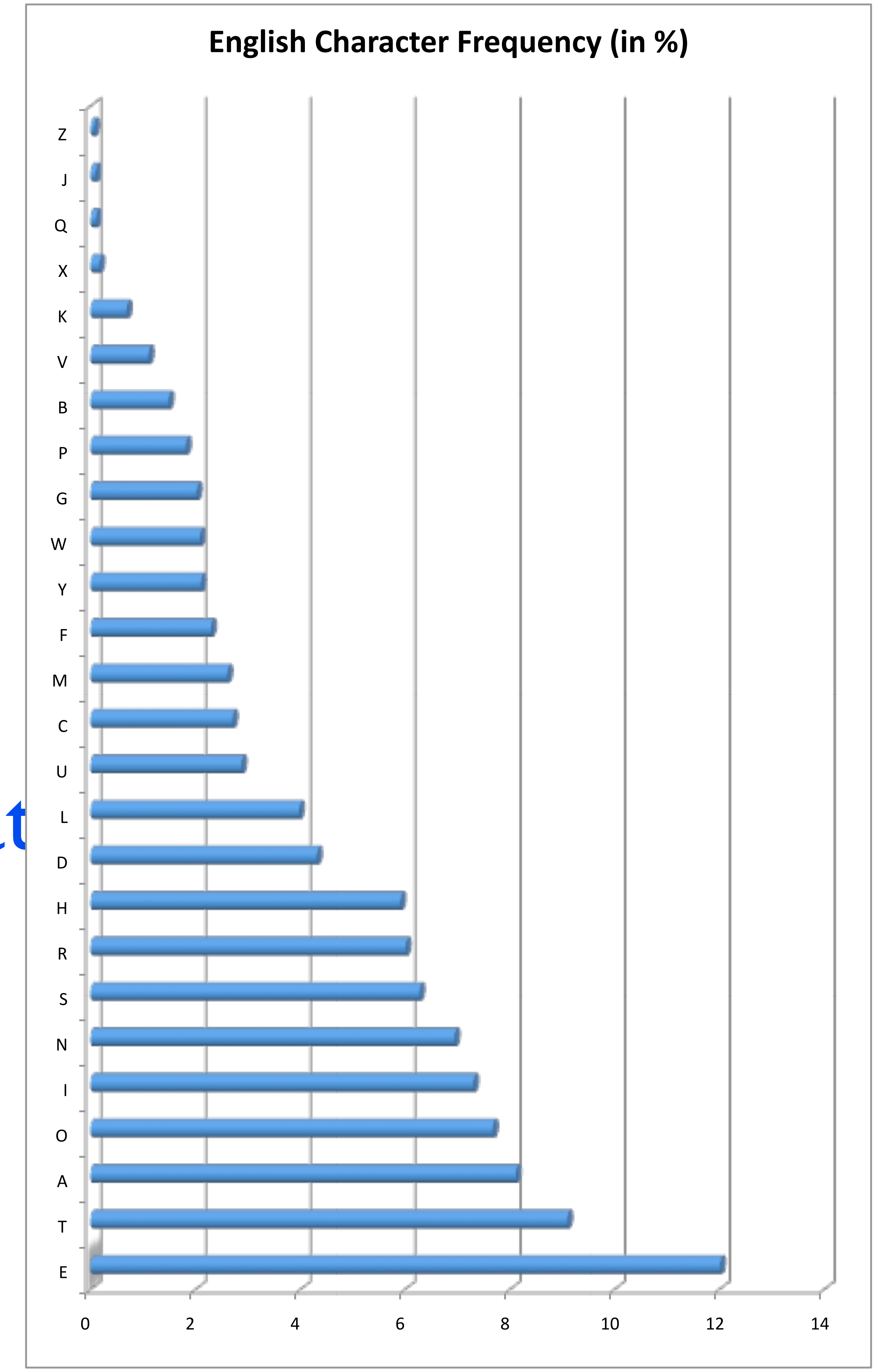
- Exhaustive search is difficult
  - key space size is $26! \approx 4 \times 10^{26}$

- Dominates the art of secret writing throughout the first millennium A.D.

- Thought to be unbreakable by many back then

Why are substitution ciphers breakable?

- Substitution ciphers are breakable because they don't hide the underlying frequency of characters.
- 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 vulnerable to frequency analysis at
- 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?

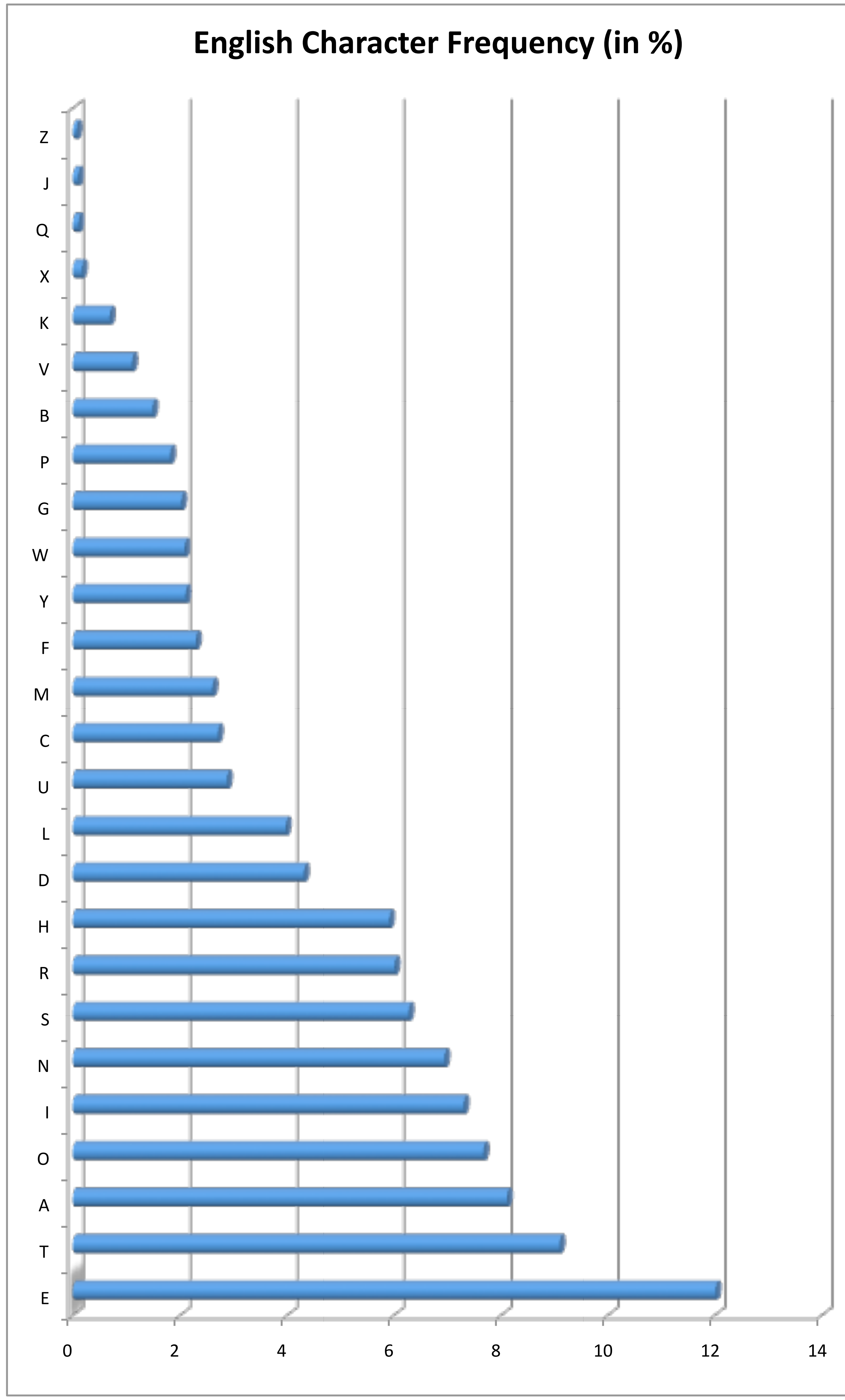


Using frequency ..

- 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 ..

PENNSTATE 1855

- 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.

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

Vigenère 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

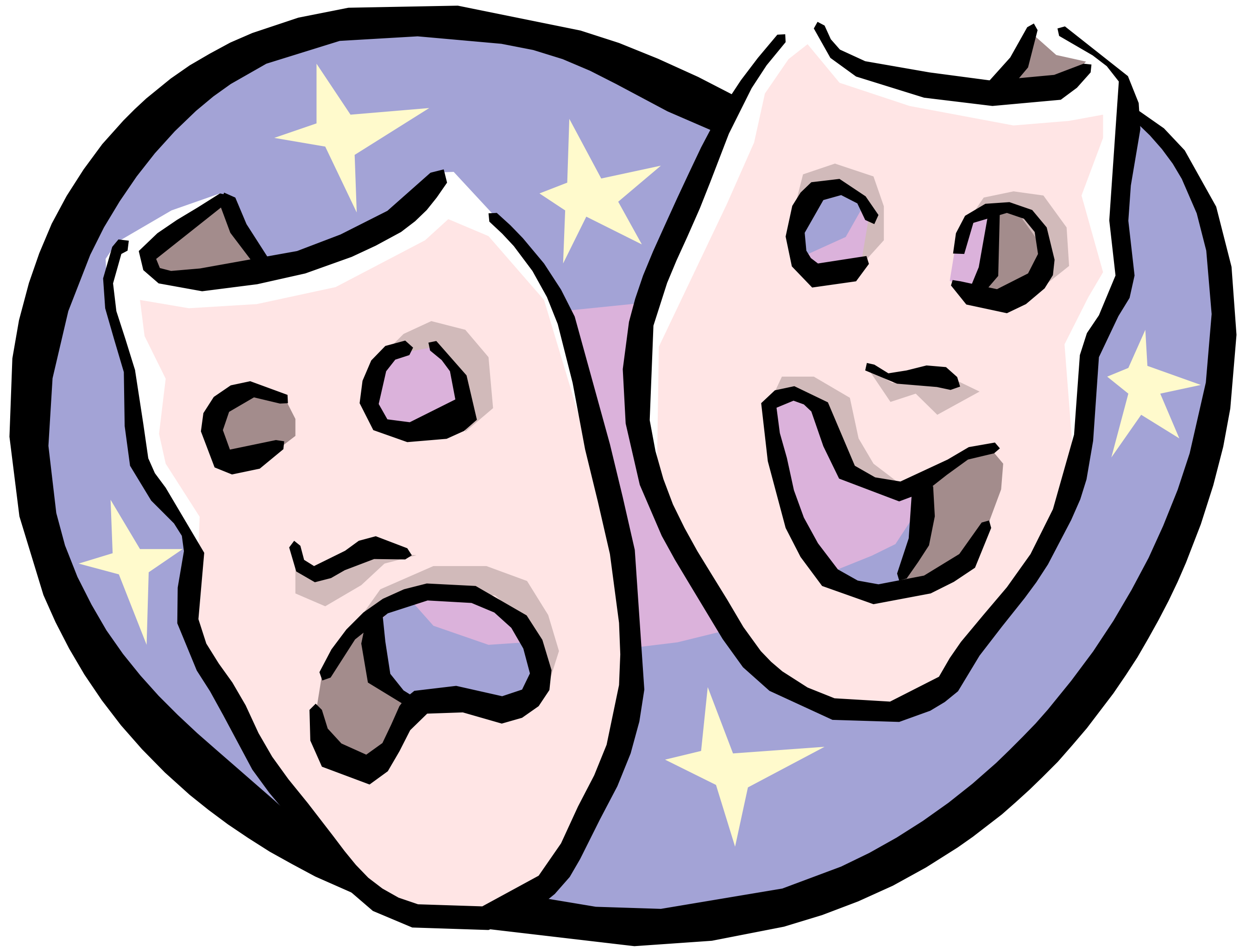
C=2 (starting the index at 0)
L=11
N=13

Z is a real number

Vigenère Cipher

- Vigenere masks the frequency with which a character appears in a language: one letter in the ciphertext corresponds to multiple letters in the plaintext. Makes the use of frequency analysis more difficult.

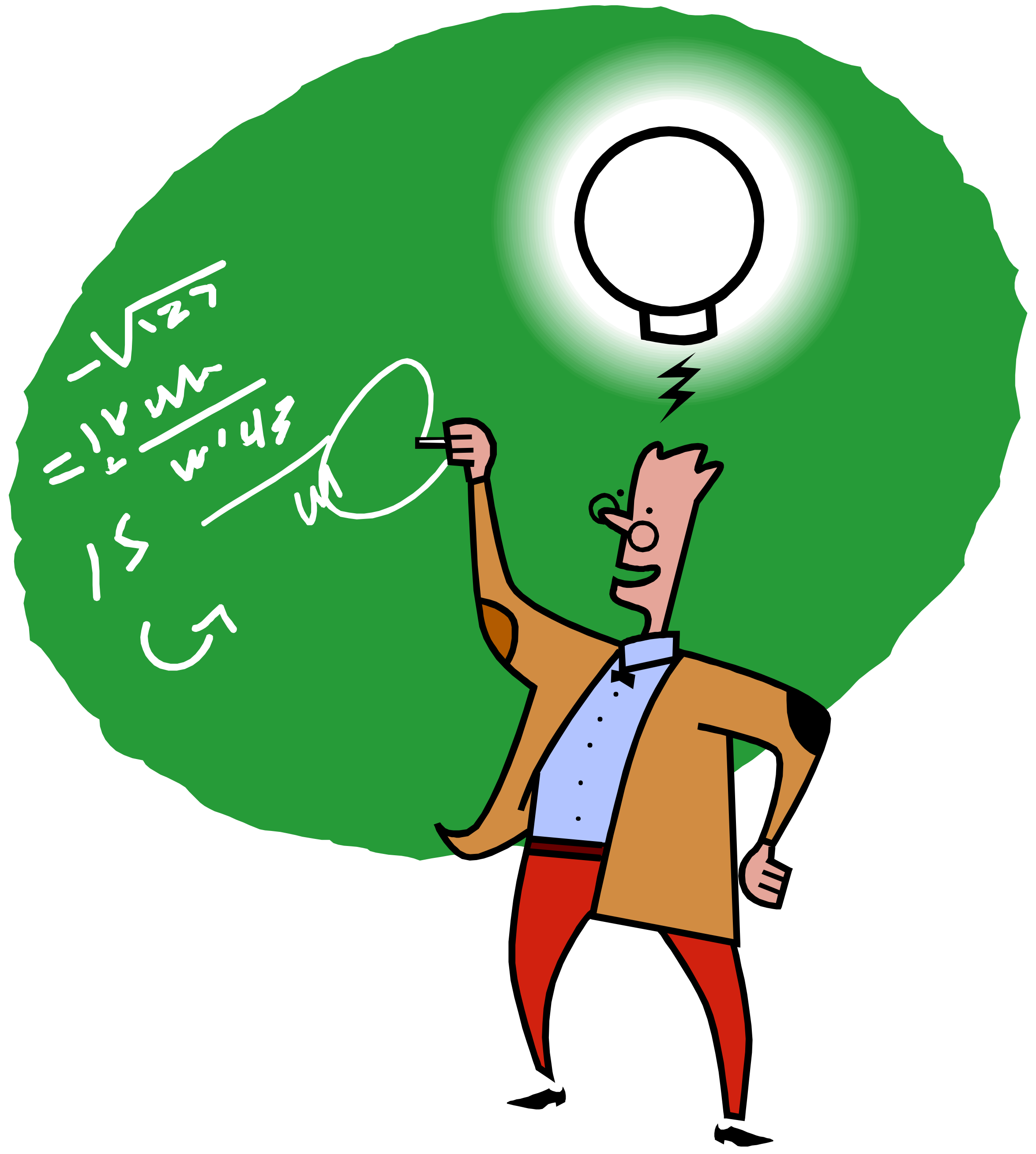
- Any message encrypted by a Vigenere cipher is a collection of as many shift ciphers as there are letters in the key.



Vigenere Cipher: Cryptanalysis

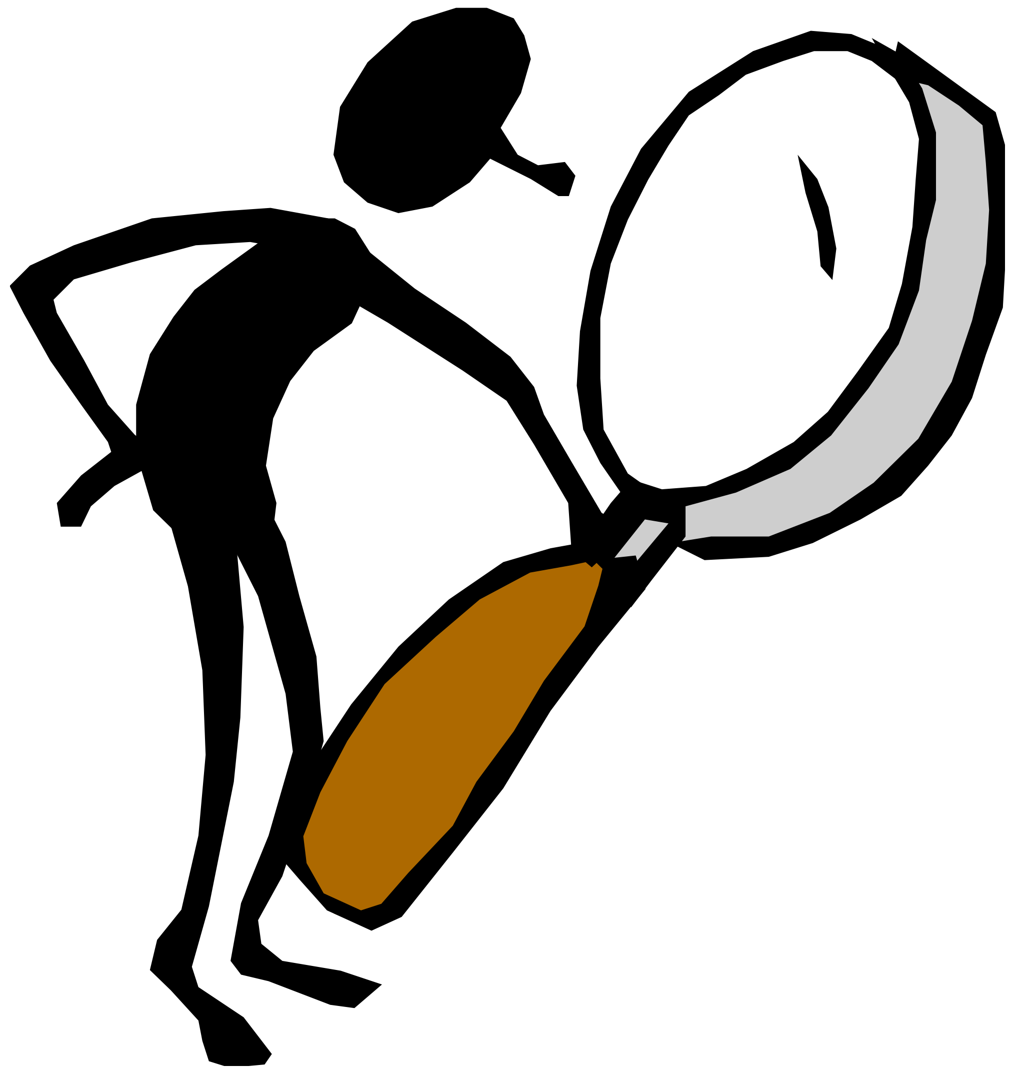
PennState

- Find the length of the key.
  - Kasisky test
  - Index of coincidence (we won't cover here)
- Divide the message into that many shift cipher encryptions.
- Use frequency analysis to solve the resulting shift ciphers.
  - How?



Kasisky Test for Finding Key Length

- Observation: two identical segments of plaintext, will be encrypted to the same ciphertext, if they occur in the text at a distance D such that D is a multiple of m, the key length.
- Algorithm:
  - Search for pairs of identical segments of length at least 3
  - Record distances between the two segments: D1, D2, …
  - m divides gcd(D1, D2, …)



Example of the Kasisky Test

- Key `K I N G K I N G K I N G K I N G K I N G K I N G`
- PT `t h e s u n a n d t h e m a n i n t h e m o o n`
- CT `D P R Y E V N T N B U K W I A O X B U K W W B T`

- Repeating patterns (strings of length 3 or more) in ciphertext are likely due to repeating plaintext strings encrypted under repeating key strings; thus the location difference should be multiples of key lengths.

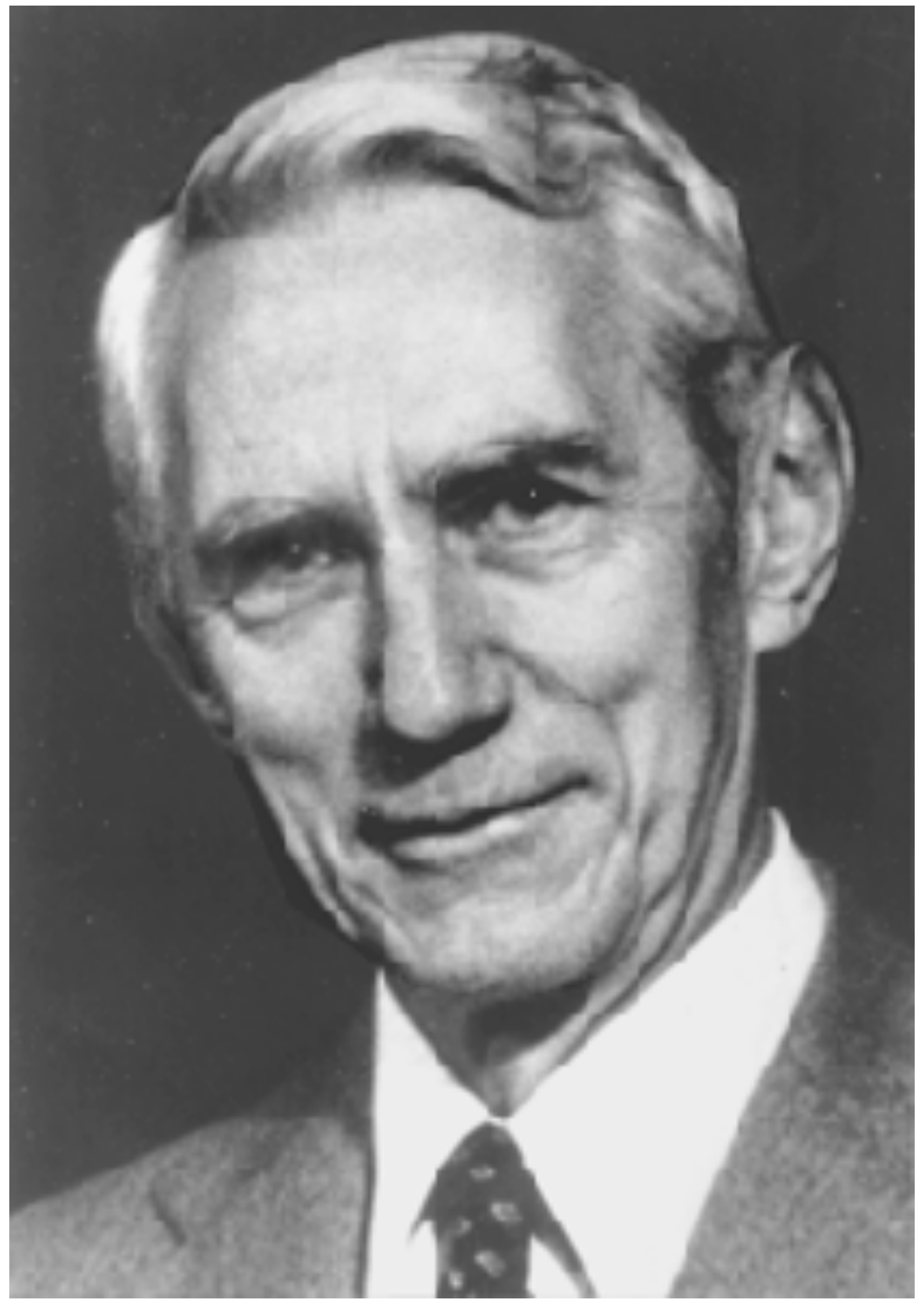
Example of the Kasisky Test

• Key	K I N G K I N G K I N G K I N G K I N G K I N G
• PT	t h e s u n a n d t h e m a n i n t h e m o o n
• CT	D P R Y E V N T N B U K W I A O X B U K W W B T

- Repeating patterns (strings of length 3 or more) in ciphertext are likely due to repeating plaintext strings encrypted under repeating key strings; thus the location difference should be multiples of key lengths.

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,…,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 $Y = (y_1\ y_2\ \ldots\ y_n)$
- $e_k(X) = (x_1+k_1\ \ x_2+k_2\ \ldots\ x_n+k_n) \bmod m$
- $d_k(Y) = (y_1-k_1\ \ y_2-k_2\ \ldots\ y_n-k_n) \bmod m$

The one-time pad (OTP)

- Assume you have a secret bit string k 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.

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?

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

Pr [PT = m] is what the adversary believes the probability that the plaintext is m, before seeing the ciphertext
Pr [PT = m | CT=c] is what the adversary believes after seeing that the ciphertext is c
Pr [PT=m | CT=c] = Pr [PT = m] means that after knowing that the ciphertext is C0, the adversary's belief does not change.

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).
  - $E_{key}[M]$ = [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
- Need to ensure that IV never repeats! How?

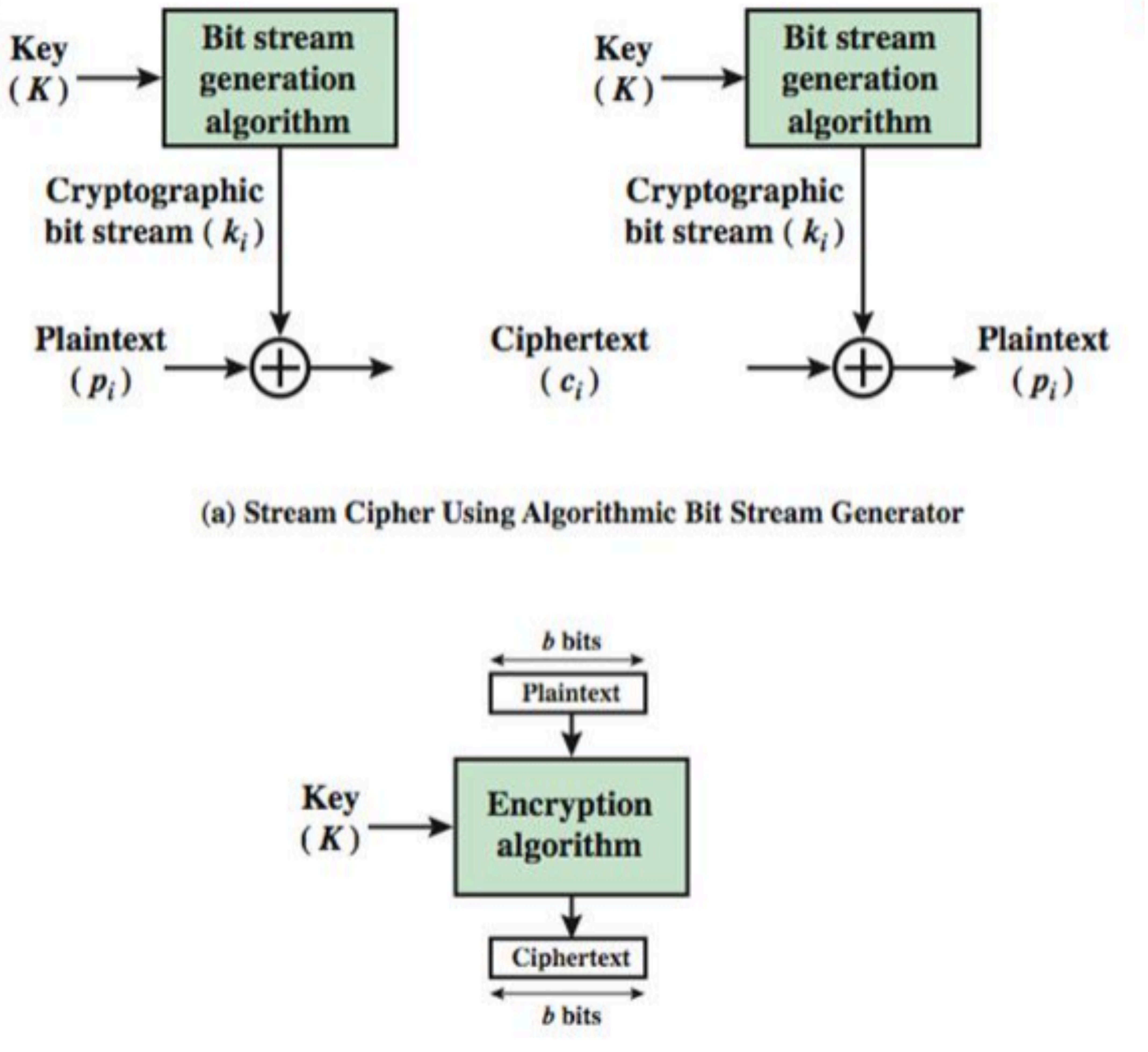
Shared key cryptography

- 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



(a) Stream Cipher Using Algorithmic Bit Stream Generator

(b) Block Cipher

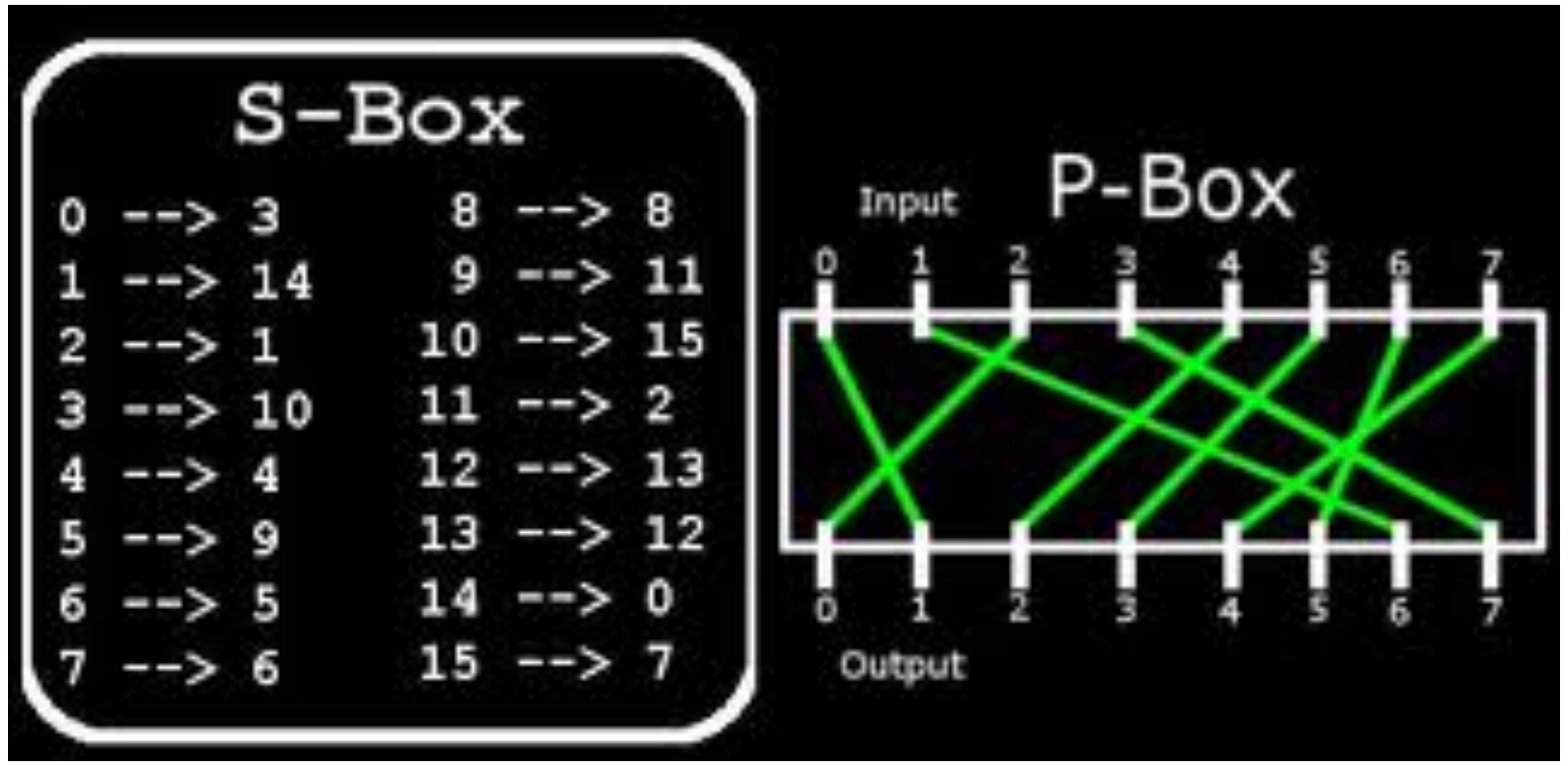
Block Cipher

- 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.

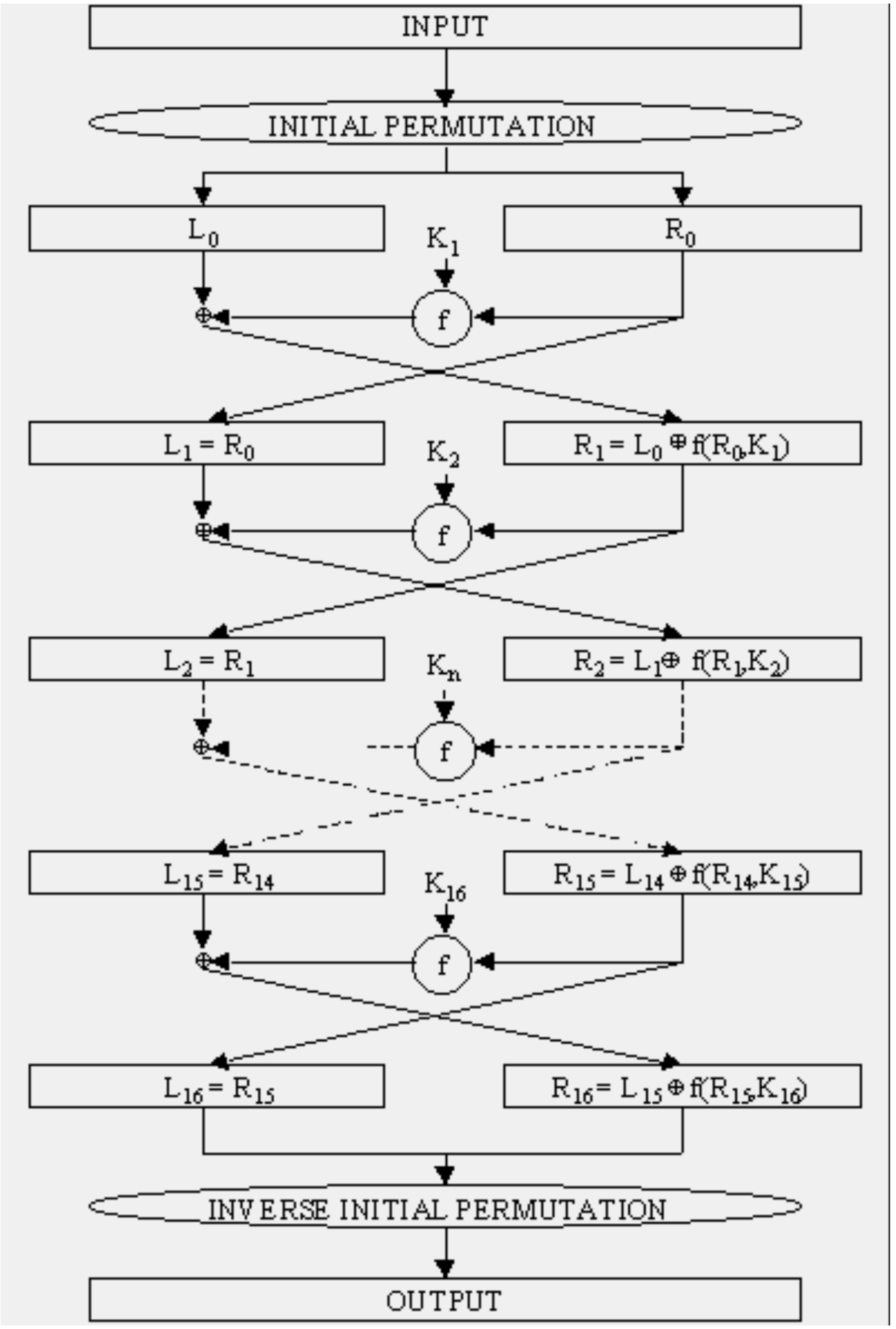
Generic Block Encryption

- 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



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



Fiestal Cipher

Encryption

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

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

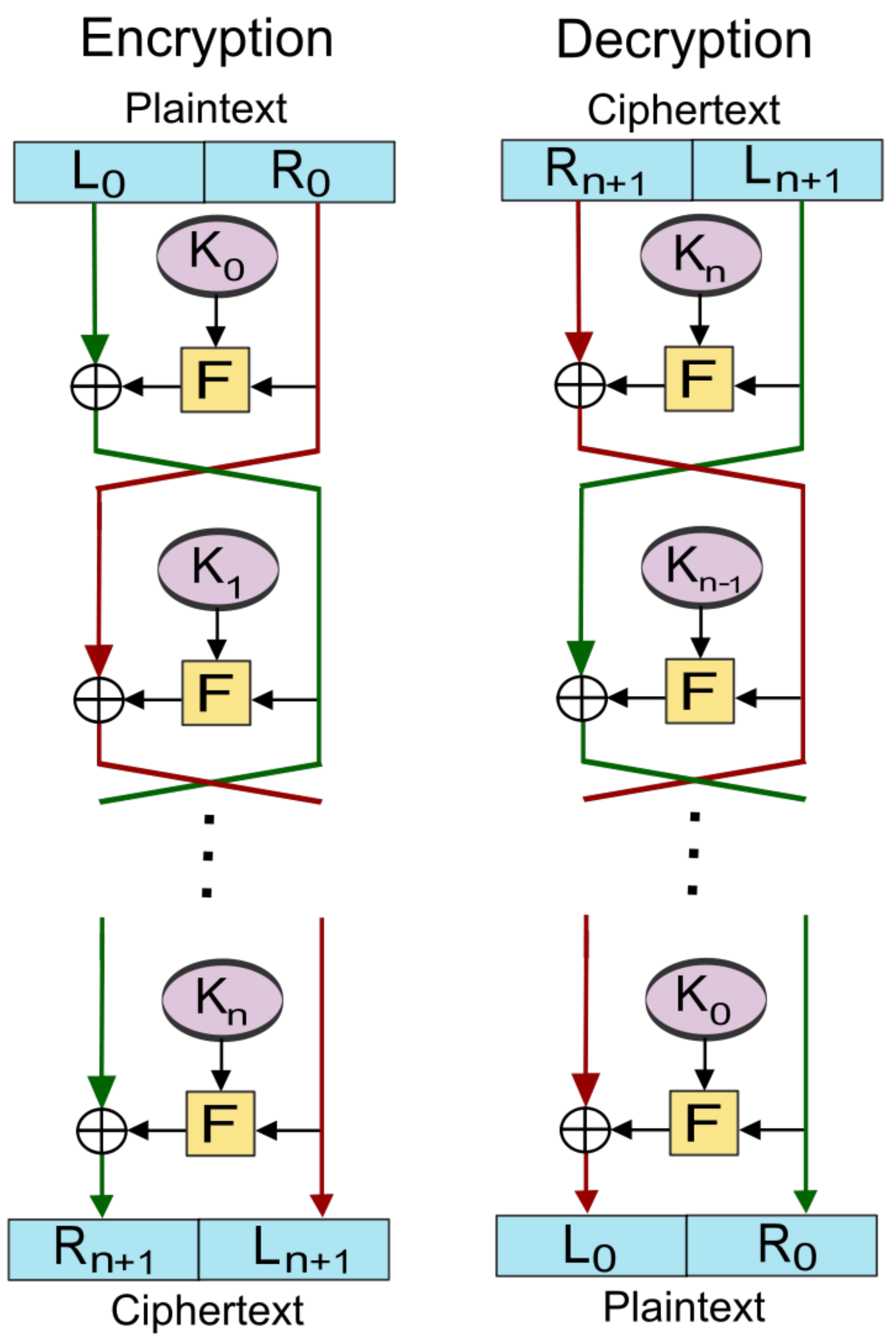
$$L_{i+1} = R_i$$
$$R_{i+1} = L_i \oplus \mathrm{F}(R_i, K_i).$$

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

Decryption

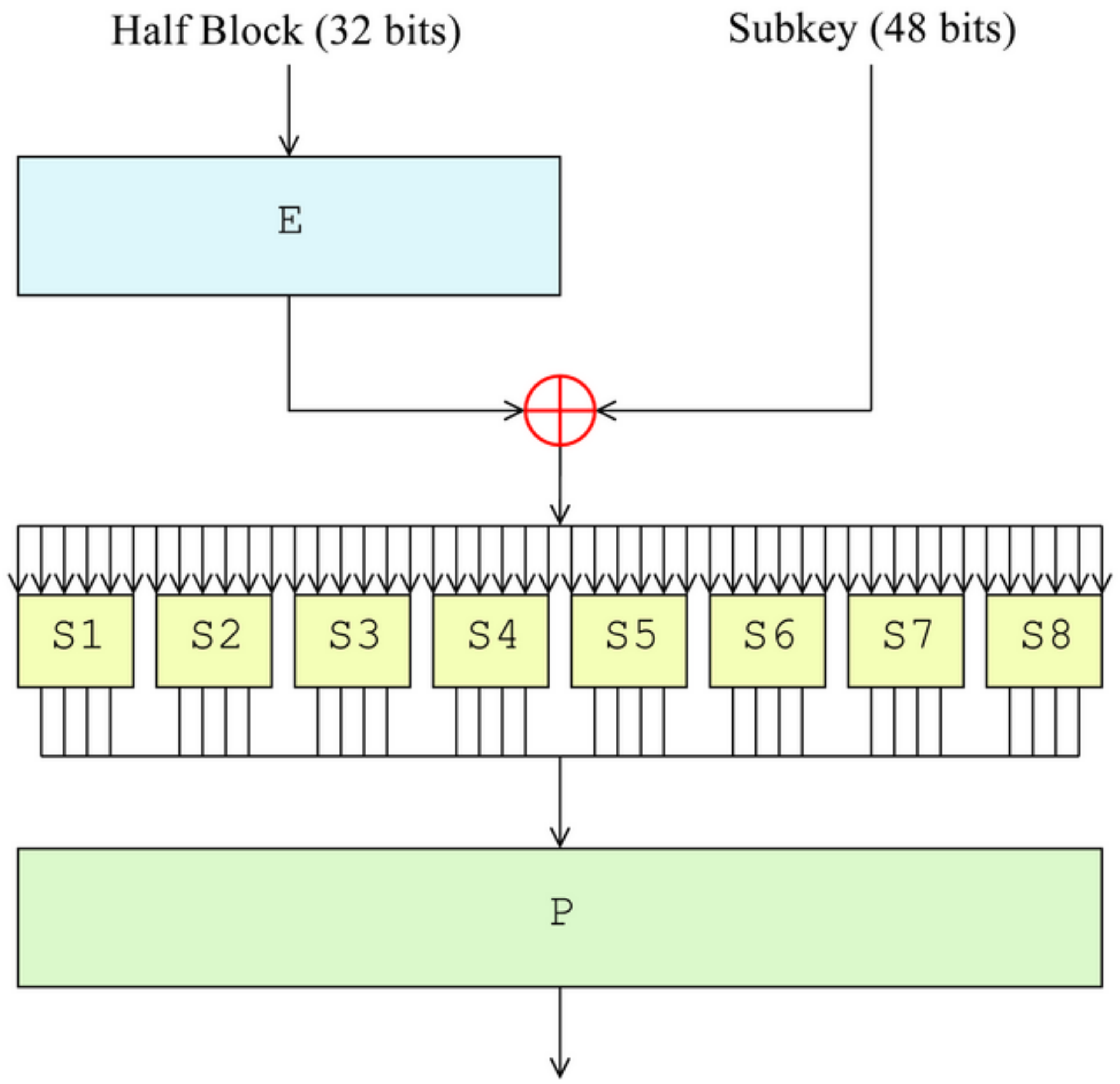
$$R_i = L_{i+1}$$
$$L_i = R_{i+1} \oplus \mathrm{F}(L_{i+1}, K_i).$$

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
- S_i: 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)

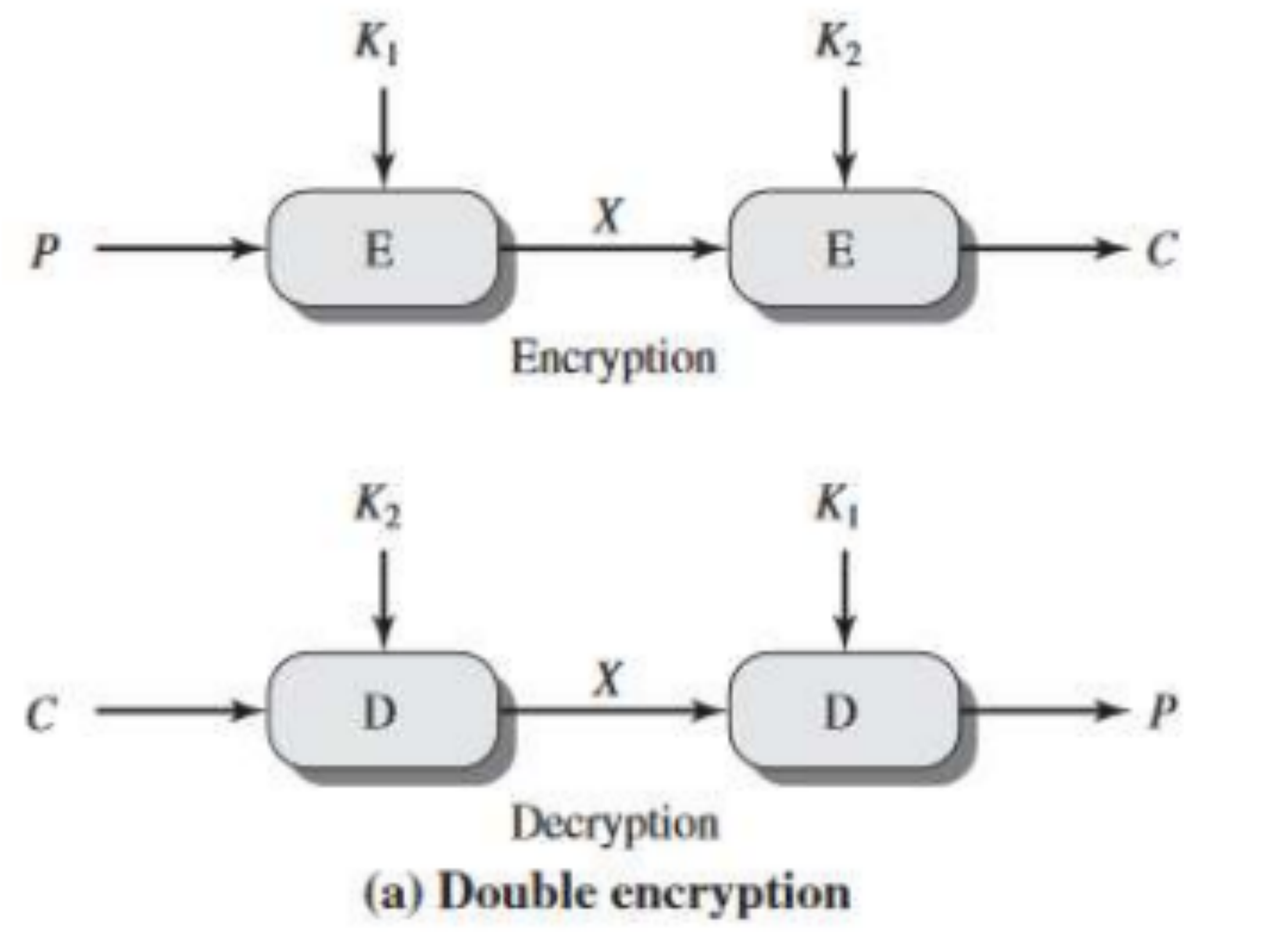
- A permutations box (or P-box) is used to obscure the relationship between the plaintext and the ciphertext
  - Shannon's property of diffusion: the relationship between plaintext and ciphertext is as complex as possible.
  - DES uses a combination of diffusion and confusion to resist cryptanalysis

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

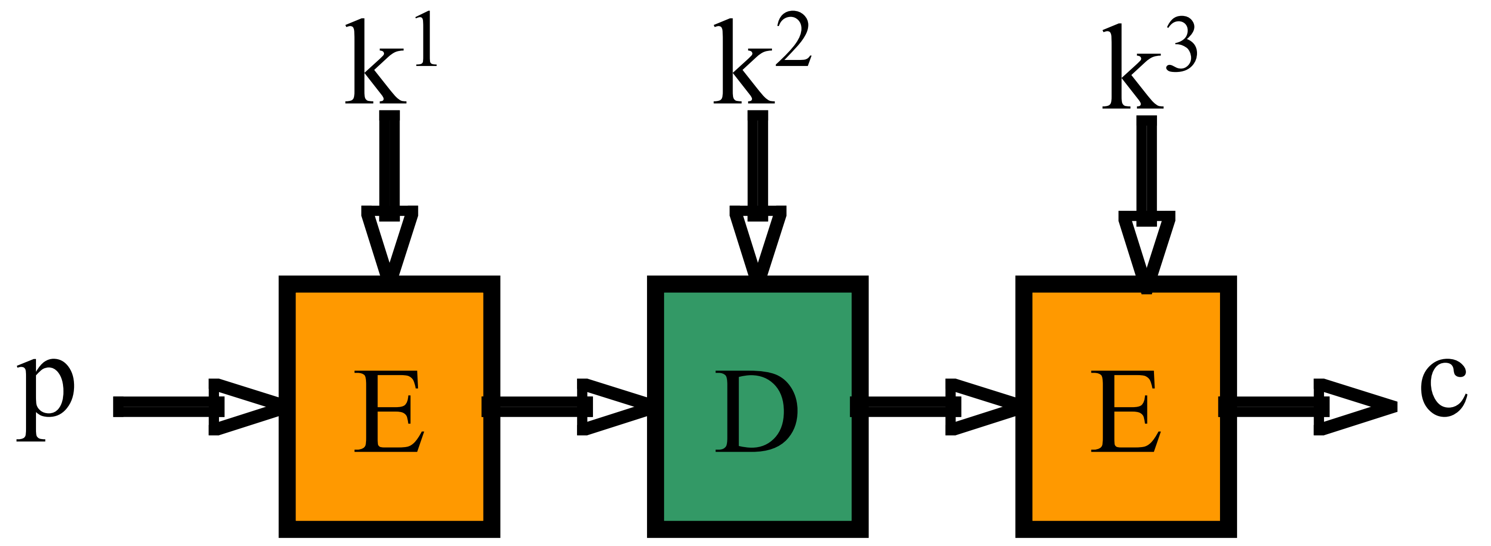
- Double DES (two keys = 112-bits)
  - Meet-in-the-Middle Attack
    - P->X (56 bits)
    - X->C (56 bits)
  - Complexity of cryptanalysis
    - 2^56 + 2^56 (worst case)



Double-DES

- Triple DES (three keys ~=112-bits)
  - keys k_1, k_2, k_3

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



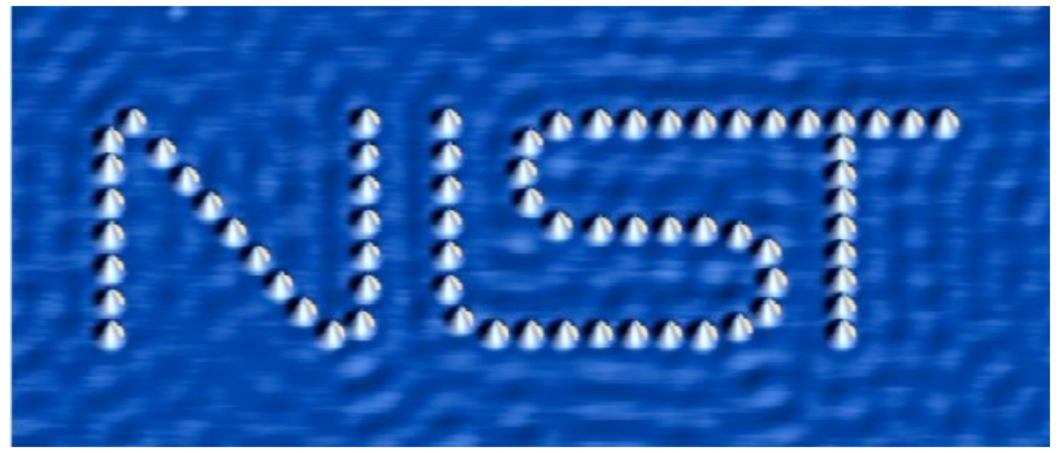
Key size and algorithm strength

- 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")



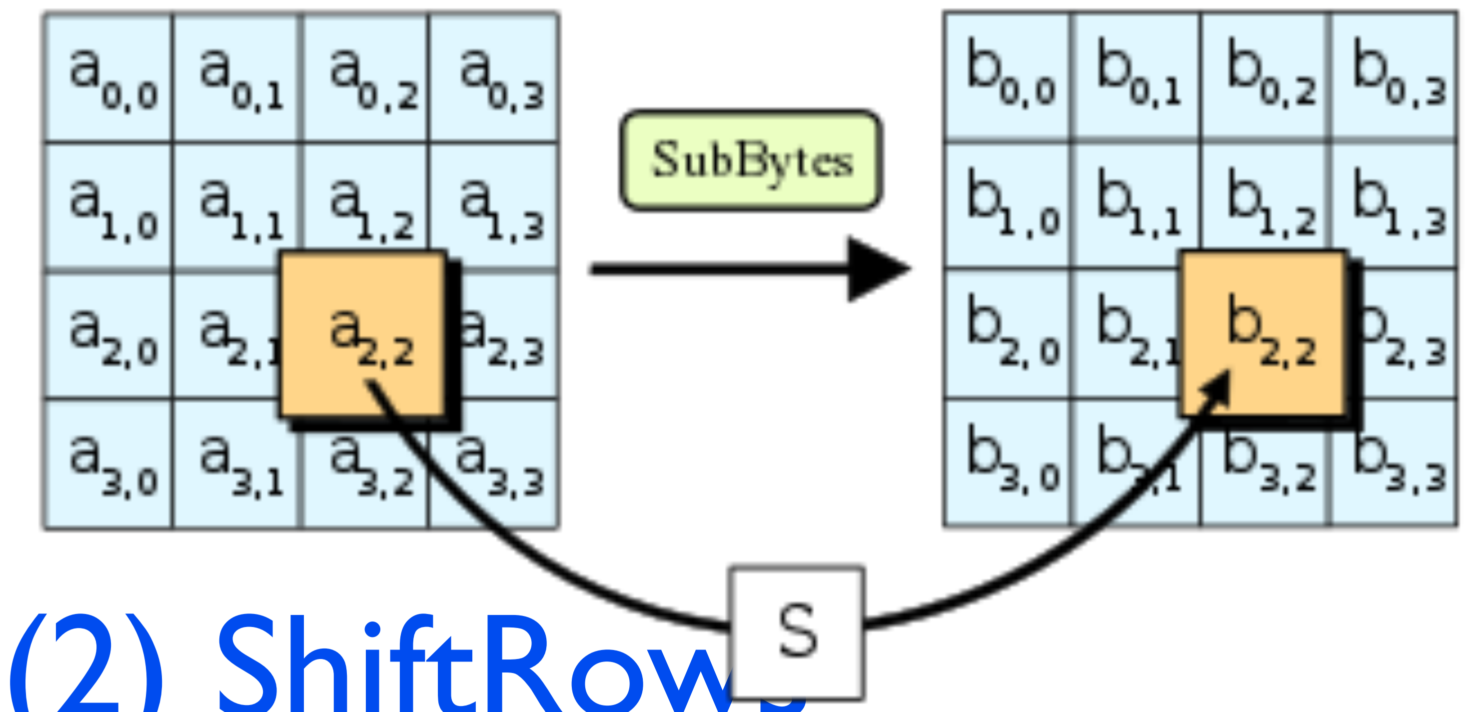
- Replacement for DES/accepted symmetric key cipher
  - Substitution-permutation network, not a Feistel network
  - Block size: 128 bits
  - Variable key lengths: 128, 192, or 256 bits
  - Fast implementation in hardware and software
  - Small code and memory footprint: No known exploitable algorithmic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

Advanced Encryption Standard (AES)

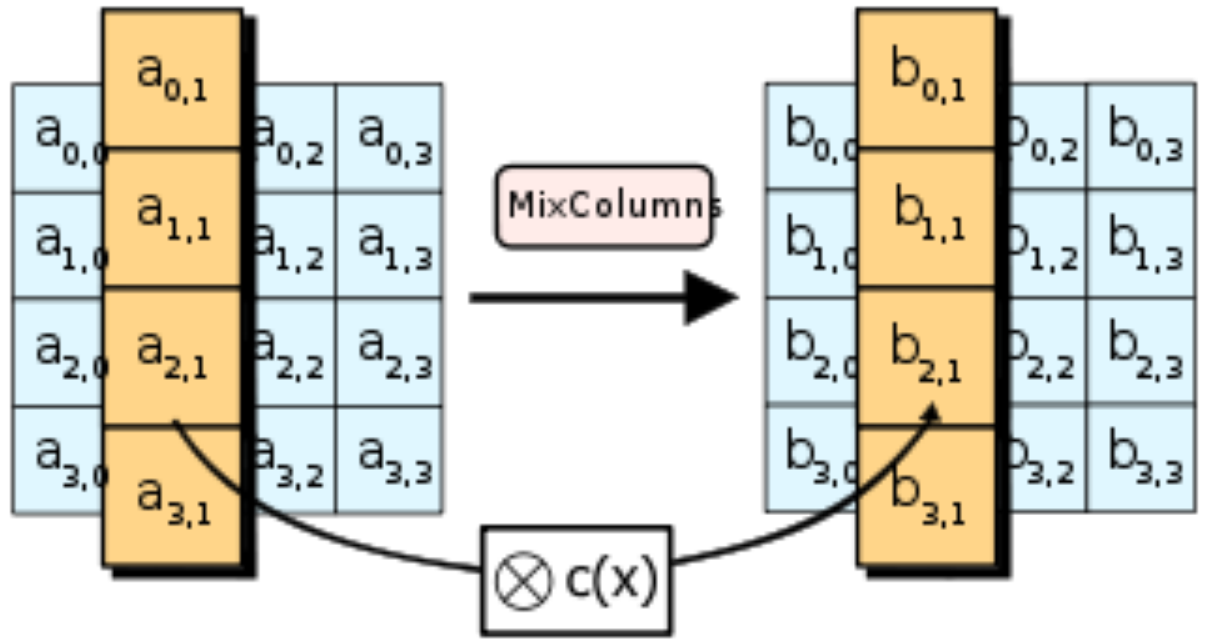
- Replace 3DES basically
- With something fast and flexible
- And secure against attacks for a while into the future
- Takes a block of the plaintext and the key as inputs and applies several alternating "rounds" or "layers" of substitution boxes (S-boxes) and permutation boxes (P-boxes) to produce the 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)

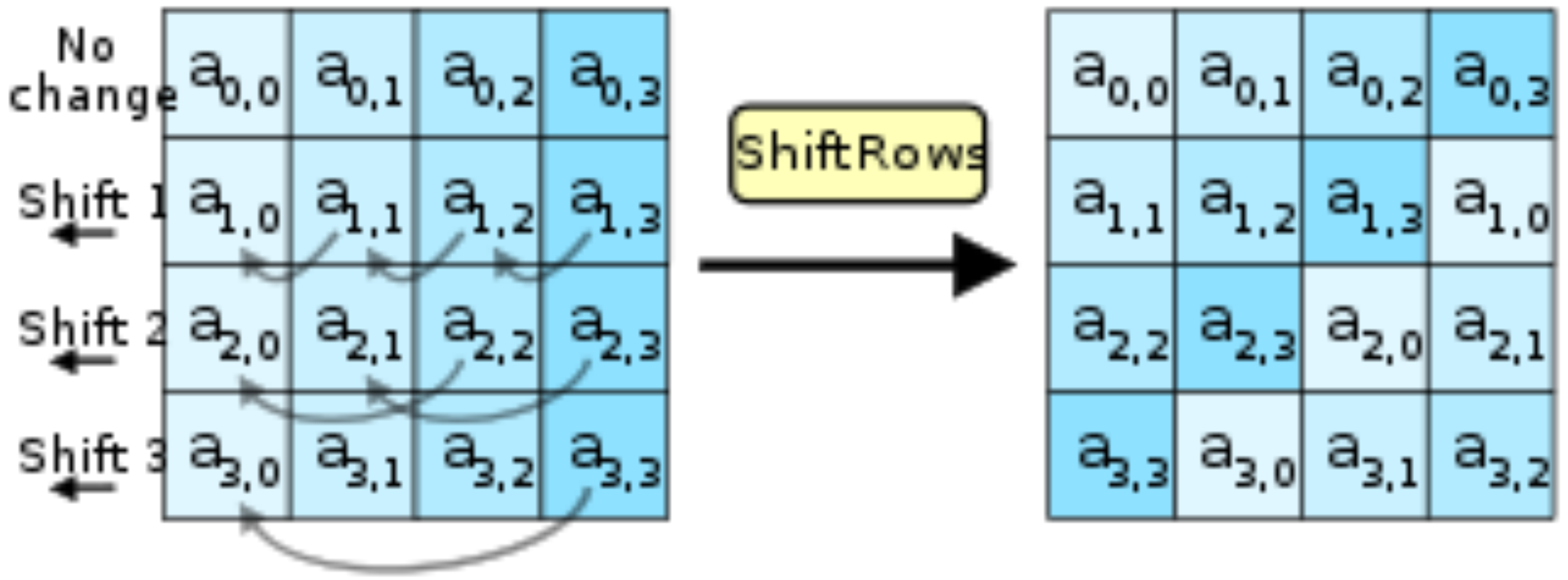
- Magic step - Round Operations
- (1) SubBytes



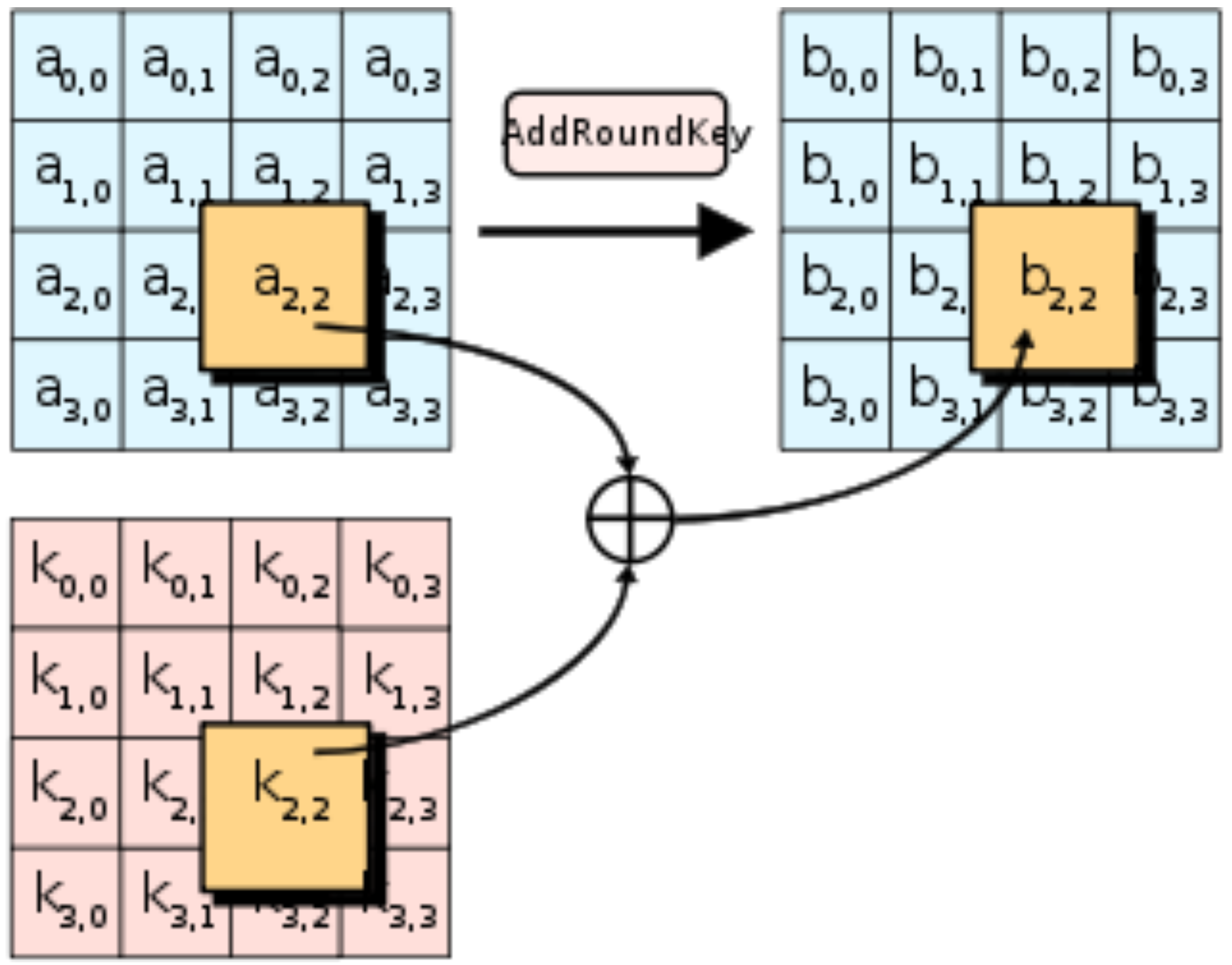
(3) MixColumns



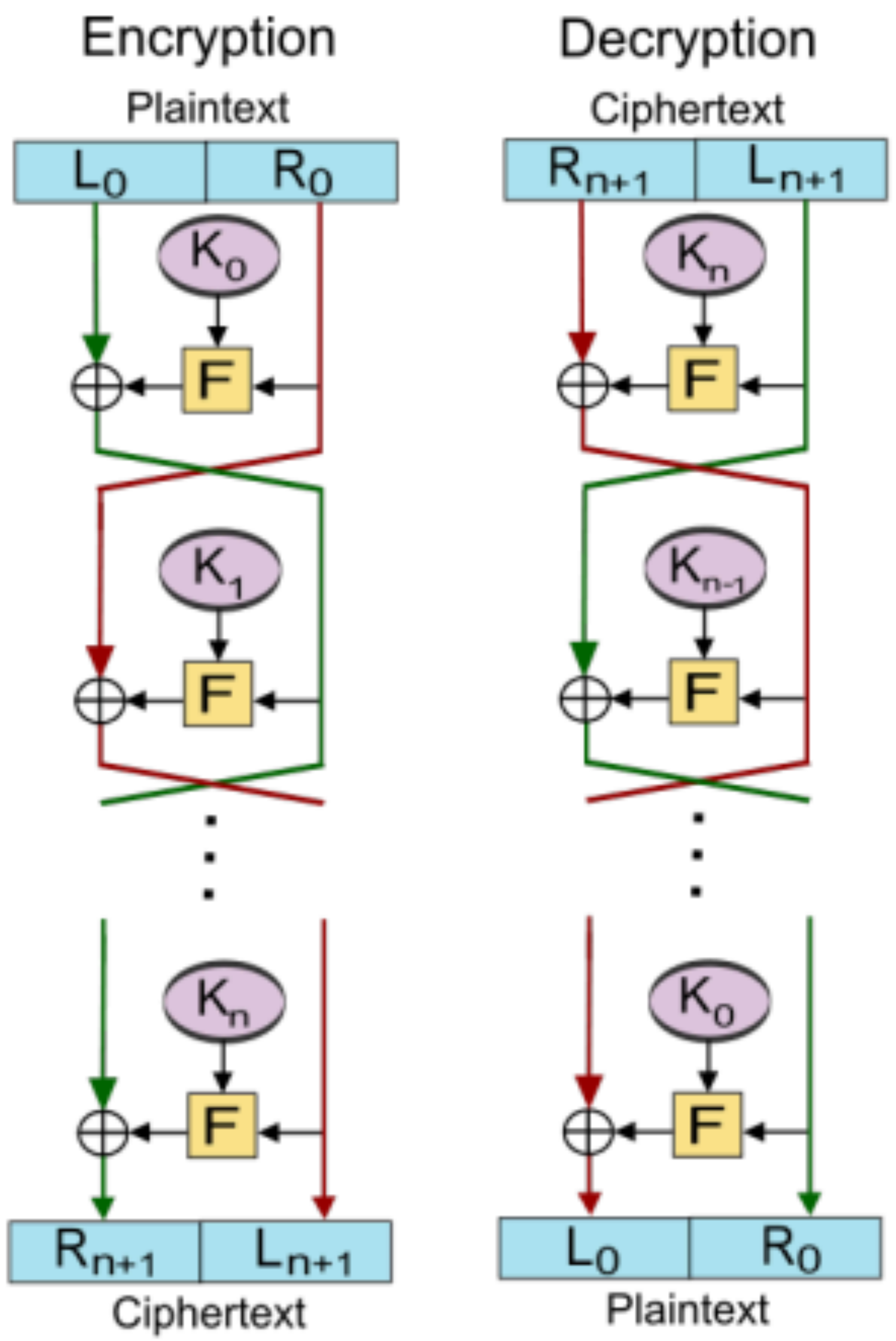
- (2) ShiftRows

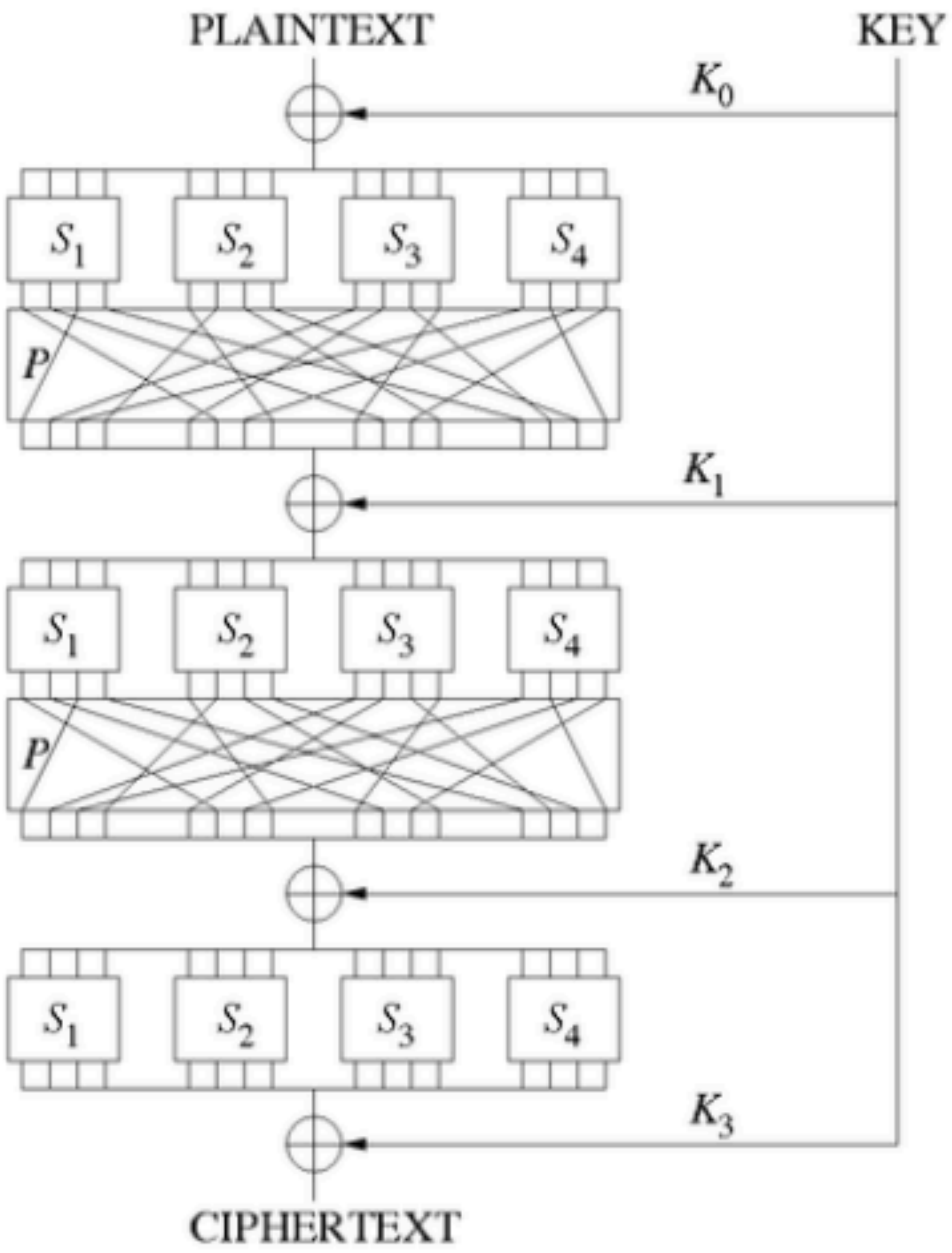


(4) AddRoundKey



Fiestal vs. SP Network





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

- 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
  - An encryption scheme is called indistinguishable under chosen plaintext attack (short IND-CPA) if an attacker cannot distinguish the encryptions of two messages of his choice.
  - A cipher is semantically secure if knowledge of the ciphertext and the length of the original message, does not reveal any additional information on the original message that can be feasibly extracted.

Symmetric Ciphers and Attacks

- How do we use symmetric key ciphers and prevent known-plaintext attacks?
  - What was the problem?

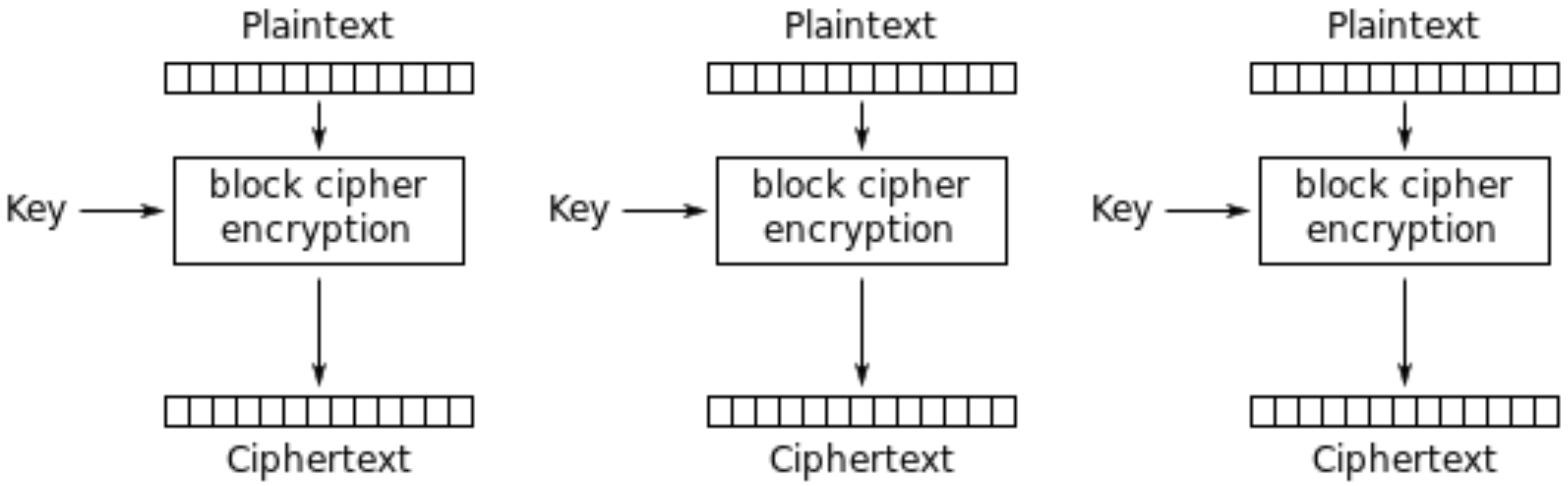
Symmetric Ciphers and Attacks

- 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?

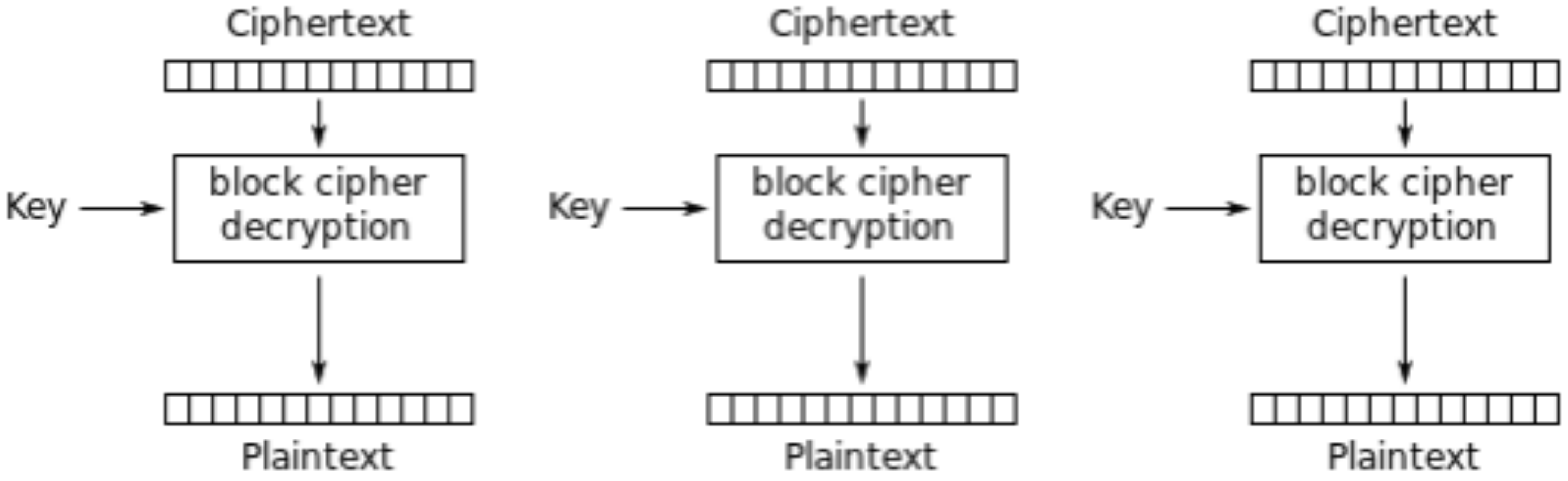
Symmetric Ciphers and Attacks

- 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)

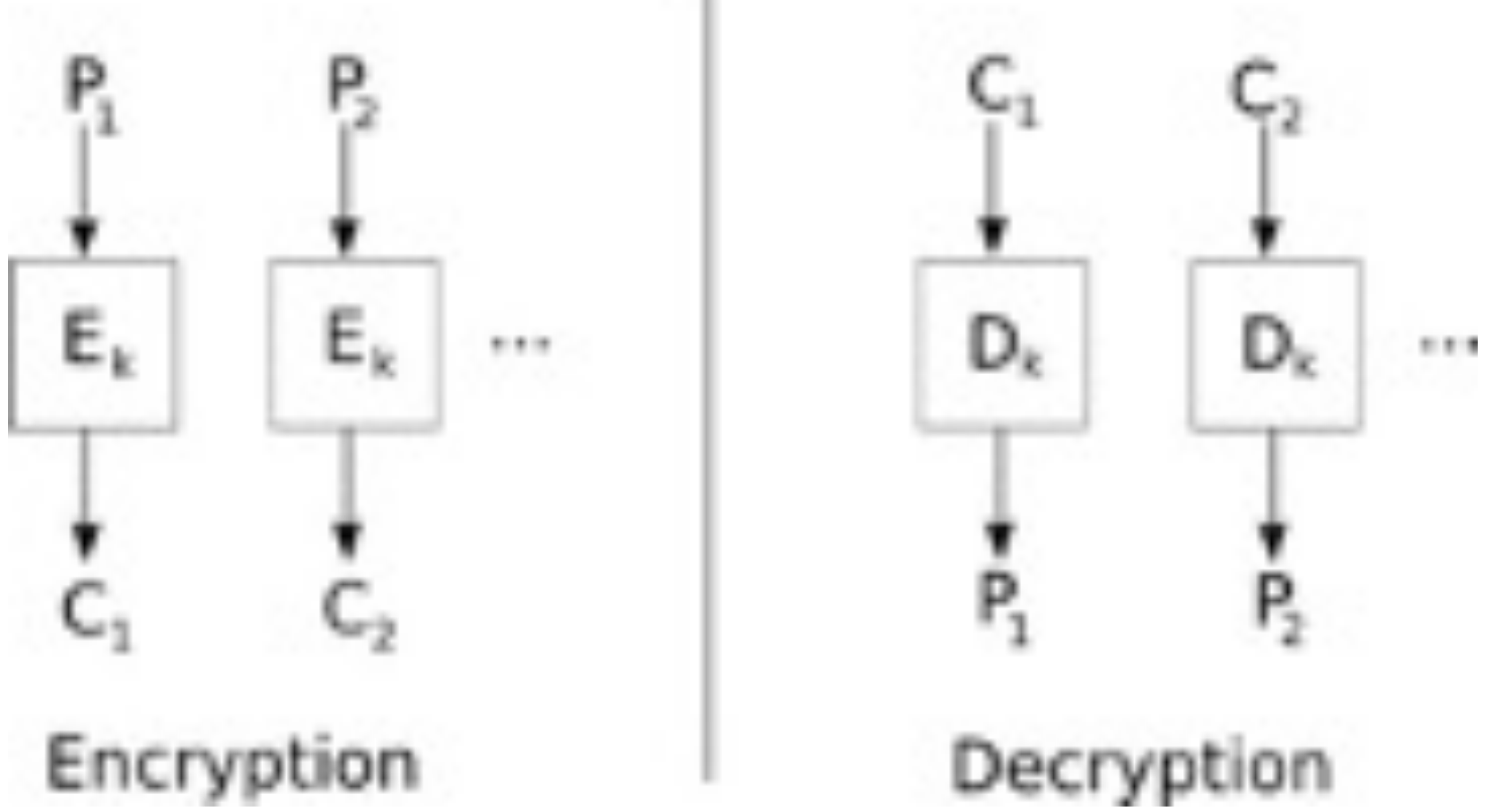
Electronic Code Book (ECB)

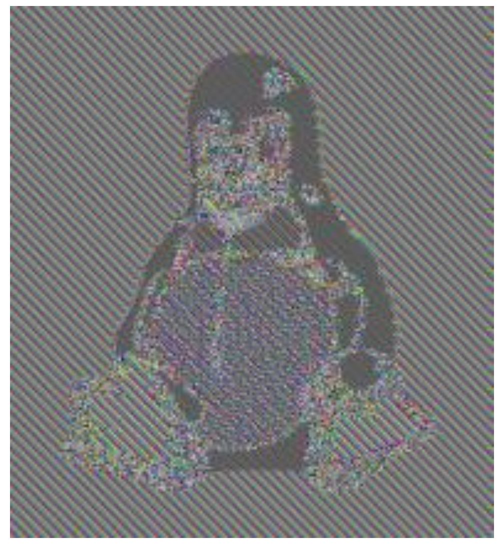


Electronic Codebook (ECB) mode encryption



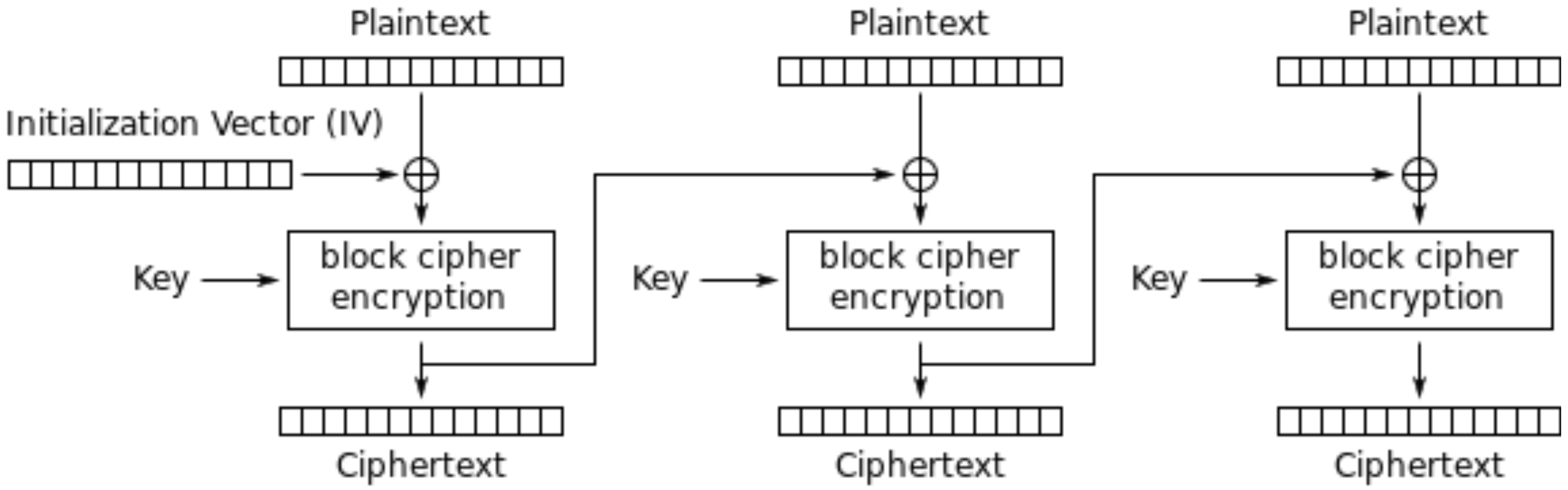
Electronic Codebook (ECB) mode decryption



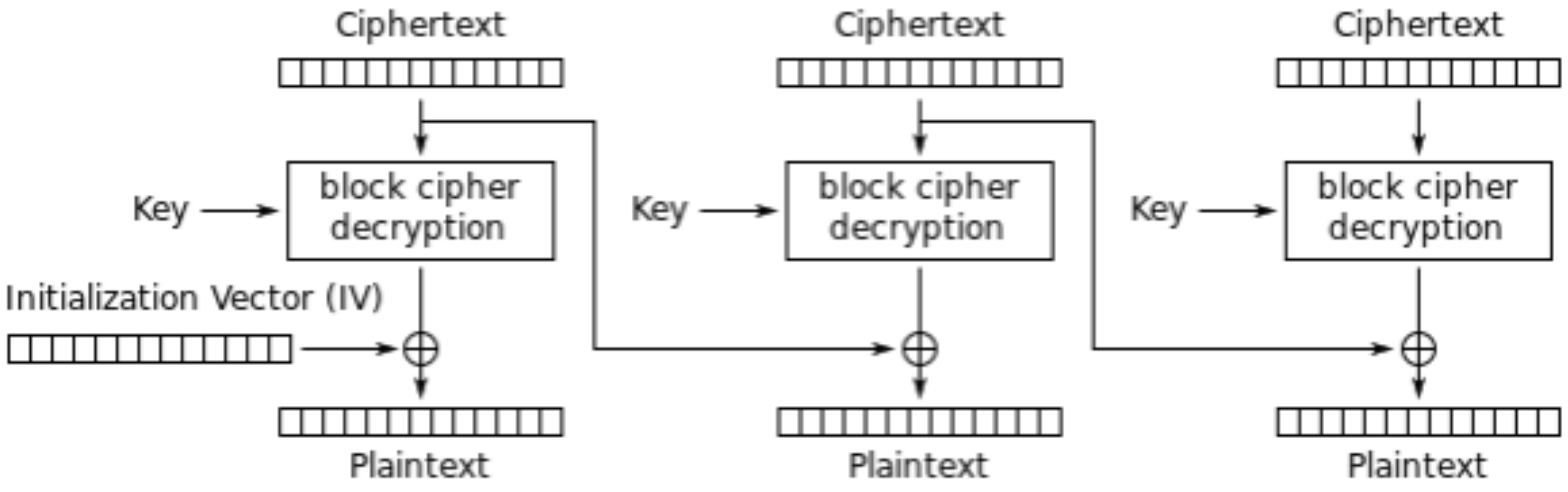


Cipher Block Chaining (CBC)

- 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



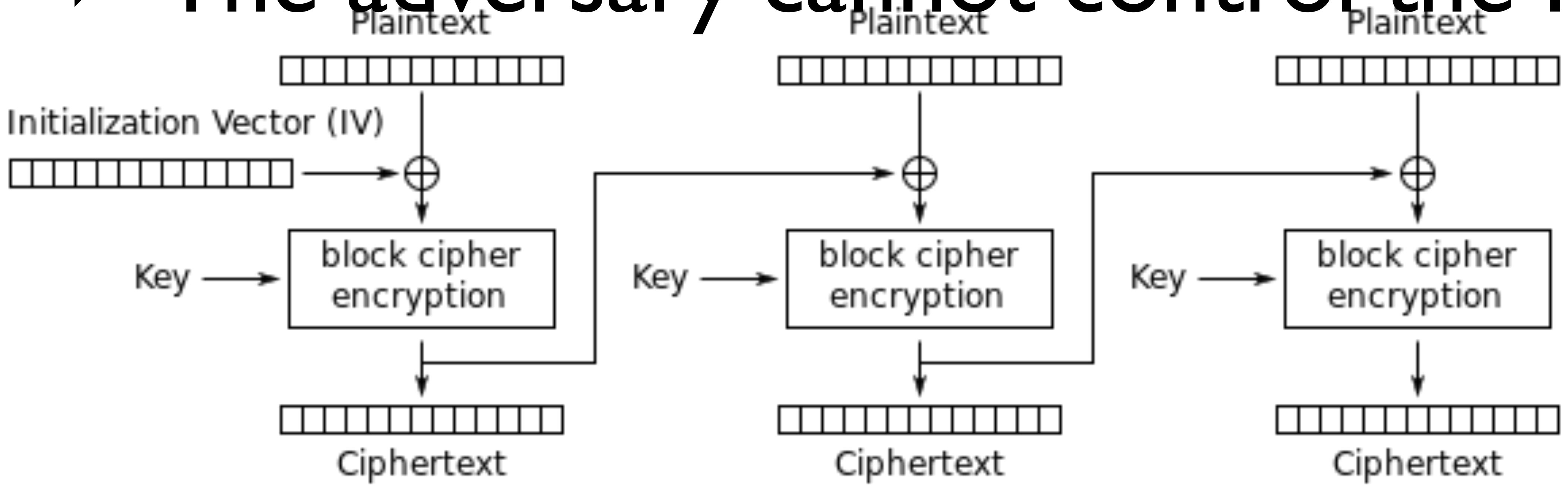
Cipher Block Chaining (CBC) mode encryption



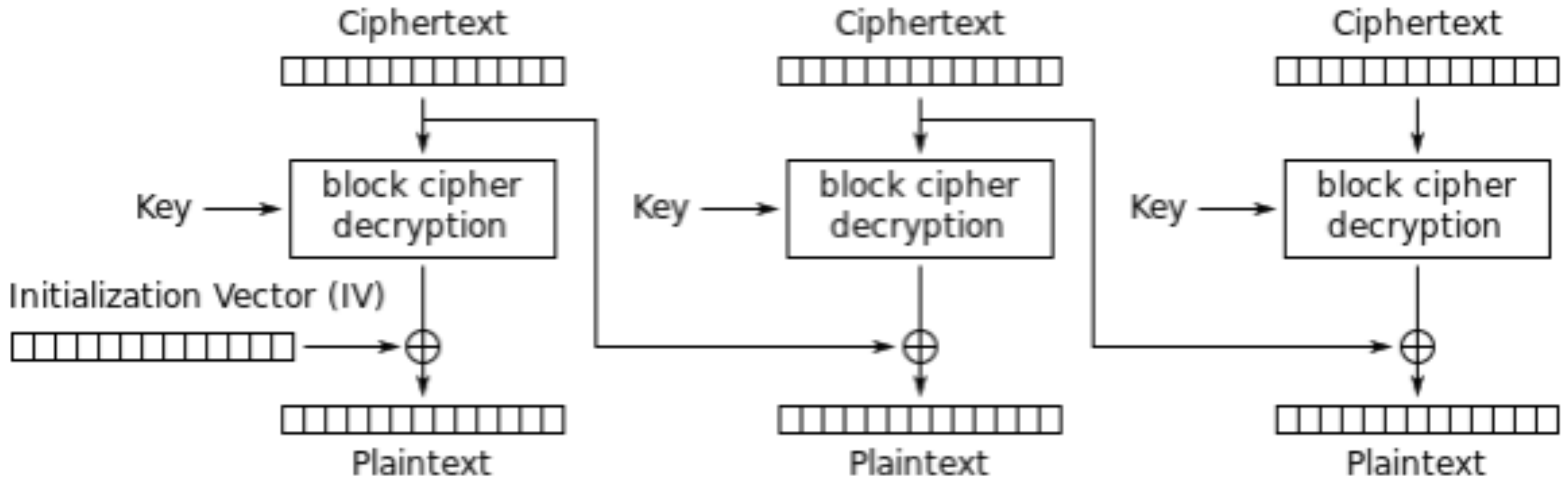
Cipher Block Chaining (CBC) mode decryption

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)
  - The adversary cannot control the IV



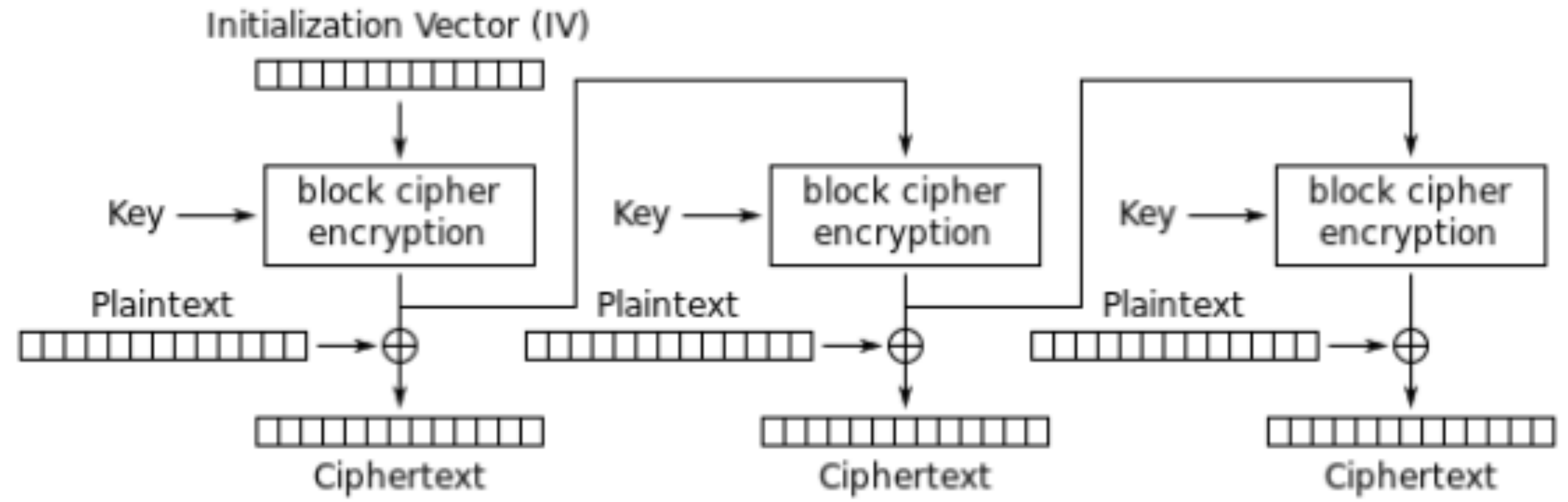
Cipher Block Chaining (CBC) mode encryption



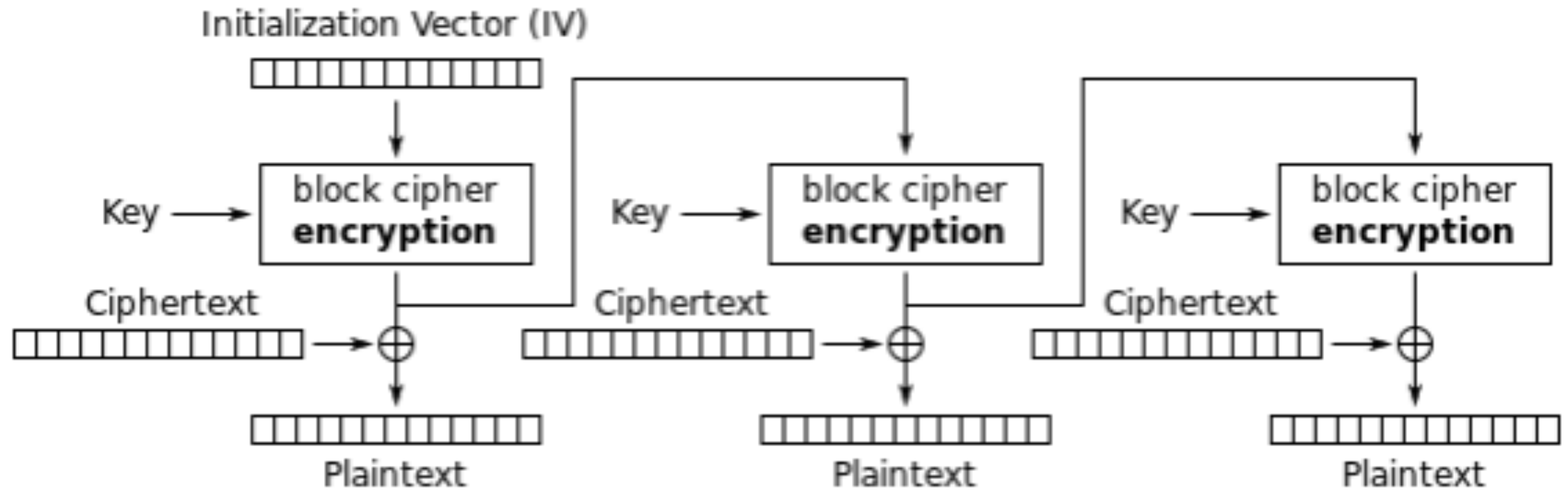
Cipher Block Chaining (CBC) mode decryption

Output Feedback (OFB)

- Turns a block cipher into a synchronous stream cipher



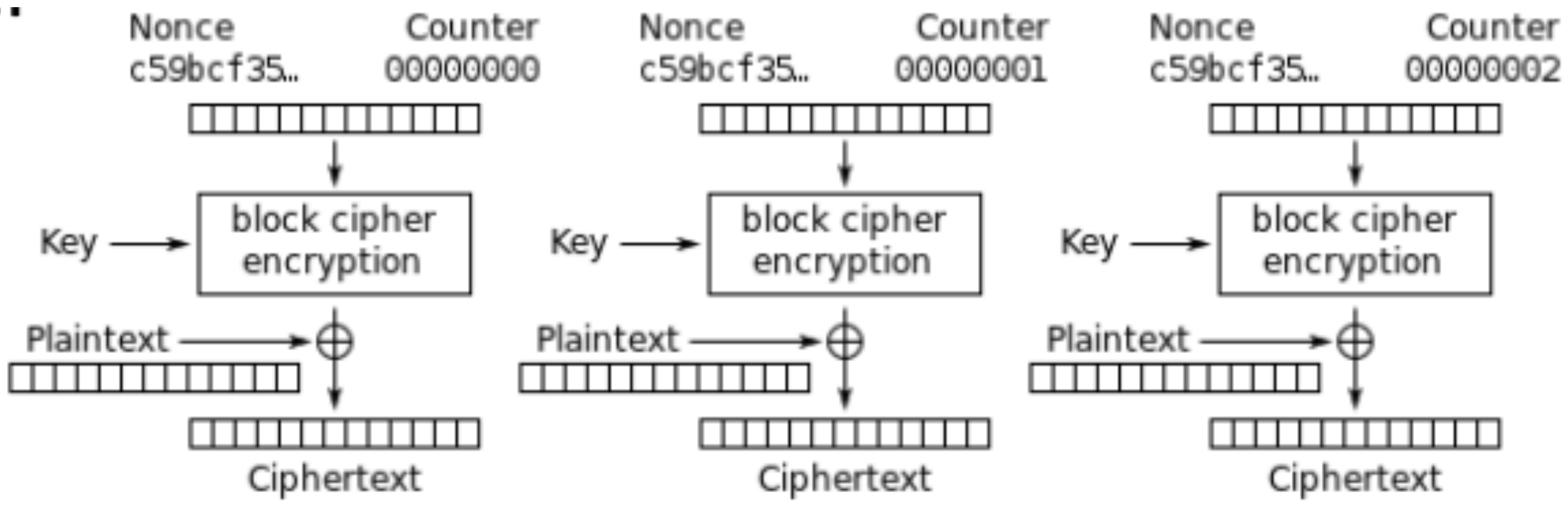
Output Feedback (OFB) mode encryption



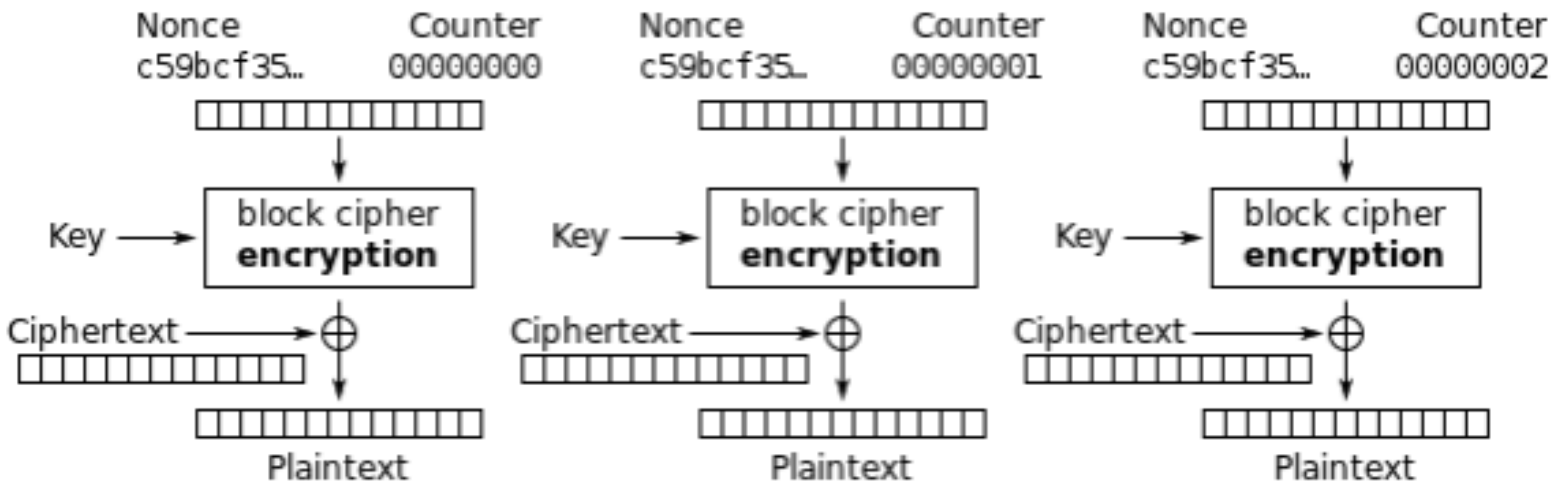
Output Feedback (OFB) mode decryption

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 xor k

c2= m2 xor k

c1 xor c2 = m1 xor m2

Building systems with cryptography

- Use quality libraries
  - E.g., OpenSSL, Libgcrypt, Cryptlib, BouncyCastle (Java, (
  - Find out what cryptographers think of a package before
- 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

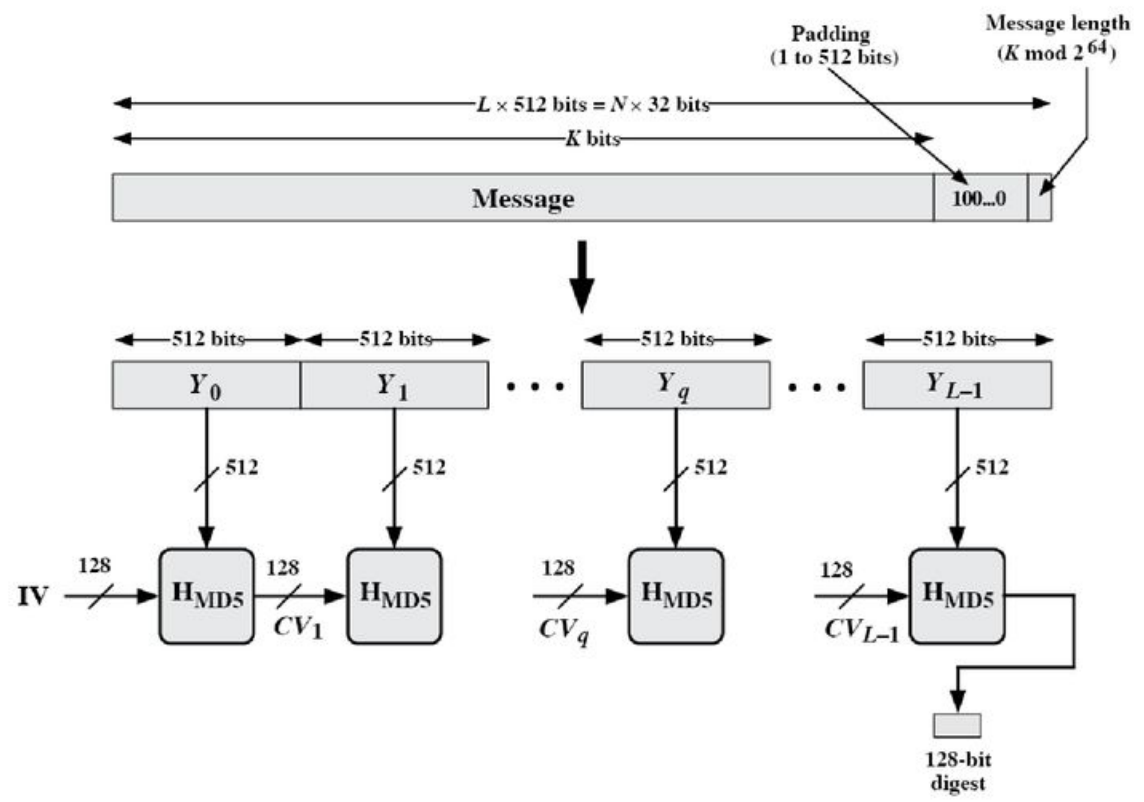
Hash Algorithms

- Hash algorithm
  - Compression of data into a hash value
  - E.g., h(d) = parity(d)
  - Such algorithms are generally useful in algorithms (speed/space optimization)
- … as used in cryptosystems
  - *One-way* - (computationally) hard to invert h() , i.e., compute $h^{-1}(y)$, where y=h(d)
    - Given y, it is hard to find d.
  - *Collision resistant* hard to find two data x_1 and x_2 such that $h(x_1)== h(x_2)$
    - weak: For any given block x1, it is computationally infeasible to find x1 != x2 with H(x1) = H(x2)
    - strong: It is computationally infeasible to find any pair (x, y) s.t. H(x1)=H(x2).
  - Q: What can you do with these constructs?

Hash Functions

- MD4, MD5
  - Substitution on complex functions in multiple passes
- SHA-1
  - 160-bit hash
  - “Complicated function”
- SHA-2, 2001
  - 256 to 512 bit hash (SHA-256)
- SHA-3, 2015
  - Keccak Algorithm
- Limited formal basis
  - Practical attacks on SHA-1, MD5

MD5



Message Digest Generation Using MD5 15

Using hashes as authenticators

- Consider the following scenario
  - Prof. Alice has not decided if she will cancel the next lecture.
  - When she does decide, she communicates to Bob the student through Mallory, her evil TA.
  - She does not care if Bob shows up to a cancelled class
  - She wants Bob to show for all classes held
-

Copyright 2005 ScienceFictionFantasyHorror.com. All rights reserved.

Hash chain

- Now, consider the case where Alice wants to do the same protocol, only for all 26 classes (the semester)
- Alice and Bob use the following protocol:

1. Alice invents a secret t
2. Alice gives Bob $h^{26}(t)$, where $h^{26}()$ is 26 repeated uses of $h()$.
3. If she cancels class on day d, she gives $h^{(26-d)}(t)$ to Mallory, e.g.,

   If cancels on day 1, she gives Mallory $h^{25}(t)$

   If cancels on day 2, she gives Mallory $h^{24}(t)$

   …….

   If cancels on day 25, she gives Mallory $h^{1}(t)$

   If cancels on day 26, she gives Mallory t
4. If does not cancel class, she does nothing

– If Bob receives the token t, he knows that Alice sent it

Hash Chain (cont.)

- Why is this protocol secure?
  - On day d, $h^{(26-d)}(t)$ acts as an authenticated value (authenticator) because Mallory could not create $h^{(26-d)}(t)$ without inverting $h^{(26-d-1)}(t)$ because for any $h^k(t)$ she has $h^j(t)$ where $26>j>k$
  - That is, Mallory potentially has access to the hash values for all days prior to today, but that provides no information on today's value, as they are all post-images of today's value
  - Note: Mallory can again convince Bob that class is occurring by not delivering $h^{(26-d)}(t)$
  - Chain of hash values are ordered authenticators
- Important that Bob got the original value $h^{26}(t)$ from Alice directly (was provably authentic)

A (simplified) sample token device

- A one-time password system that essentially uses a *hash chain* as authenticators.
  - For seed (S) and chain length (l), epoch length (x)
  - Tamperproof token encodes S in firmware

$$\mathrm{pw}_i = h^{l-i}(S)$$

  - Device display shows password for epoch i
  - Time synchronization allows authentication server to know what i is expected, and authenticate the user.
- *Note*: somebody can see your token display at some time but learn nothing useful for later periods.

Birthday Paradox

- Q: Why is the birthday paradox important to hash functions?

- **Birthday paradox :** the probability that two or more people in a group of 23 share the same birthday is >than 50%

Compute $P(A)$: probability that at least two people in the room have the same birthday.
Compute $P(A')$: the probability that no two people in the room have the same birthday.

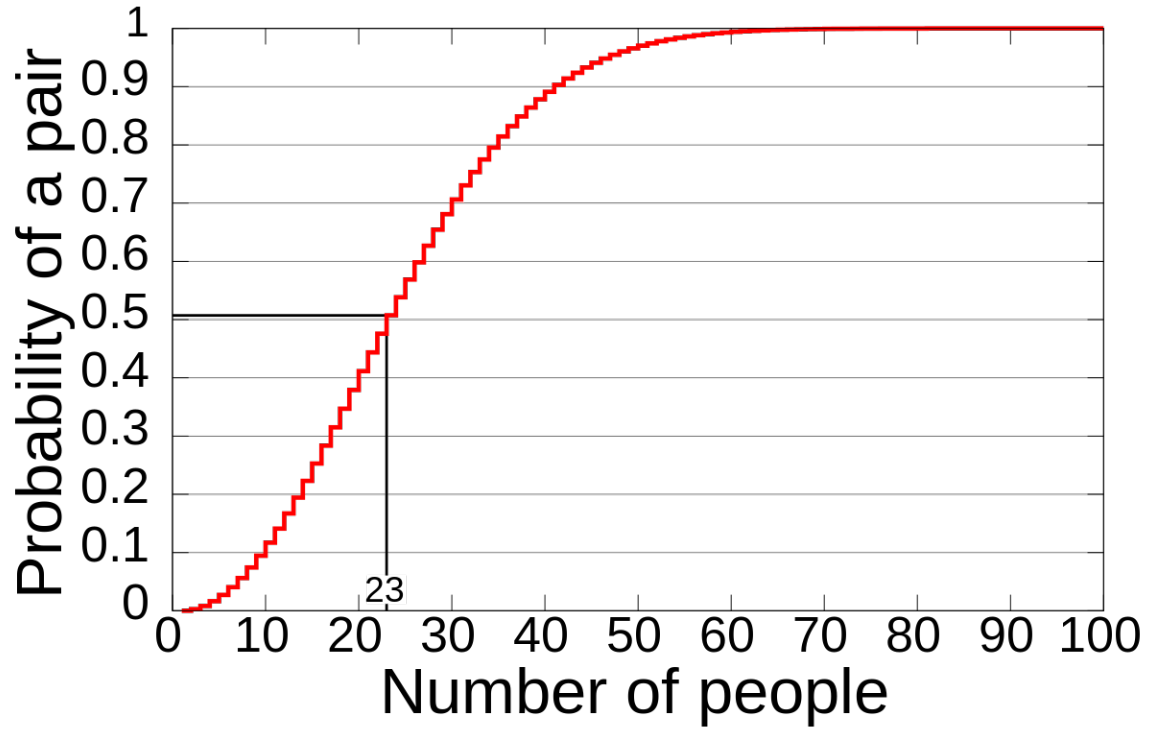
$$P(A') = \frac{365}{365} \times \frac{364}{365} \times \frac{363}{365} \times \frac{362}{365} \times \cdots \times \frac{343}{365}$$

The terms of equation (1) can be collected to arrive at:

$$P(A') = \left(\frac{1}{365}\right)^{23} \times (365 \times 364 \times 363 \times \cdots \times 343)$$

Evaluating equation (2) gives $P(A') \approx 0.492703$

Therefore, $P(A) \approx 1 - 0.492703 = 0.507297$ (50.7297%).



$$\bar{p}(n) = 1 \times \left(1 - \frac{1}{365}\right) \times \left(1 - \frac{2}{365}\right) \times \cdots \times \left(1 - \frac{n-1}{365}\right)$$

$$= \frac{365 \times 364 \times \cdots \times (365 - n + 1)}{365^n}$$

$$= \frac{365!}{365^n (365-n)!} = \frac{n! \cdot \binom{365}{n}}{365^n} = \frac{{}_{365}P_n}{365^n}$$

where ! is the factorial operator, $\binom{365}{n}$ is the binomial coefficient and ${}_kP_r$ denotes permutation.

Message Authentication Code

- MAC
  - Used in protocols to authenticate content, authenticates integrity for data d
  - To simplify, hash function h(), key k, data d

$$MAC(k,d) = h(k \,||\, d)$$

  - E.g., XOR the key with the data and hash the result

- Q: Why does this provide integrity?
  - Cannot produce MAC(k,d) unless you know k
  - If you could, then can *invert* h()
- Exercise for class: prove the previous statement

A simple proof

- Setup: algorithm X(d) that produces MAC(k,d) without k (assume d known).
  - k is a secret (in this instance a OTP)
  - d is known

$$X(d, k) = h(k \oplus d)$$

  - From Shannon: there is no information content in

$$k \oplus d$$

  - for X on k.
  - Thus, X() must know k to compute the X(d).
  - *A contradiction*.

A simple proof

- Setup: you know d and have an polynomial-time algorithm X(d) that produces MAC(k,d) without k (assume *d* is known).
- Suppose X() exists:

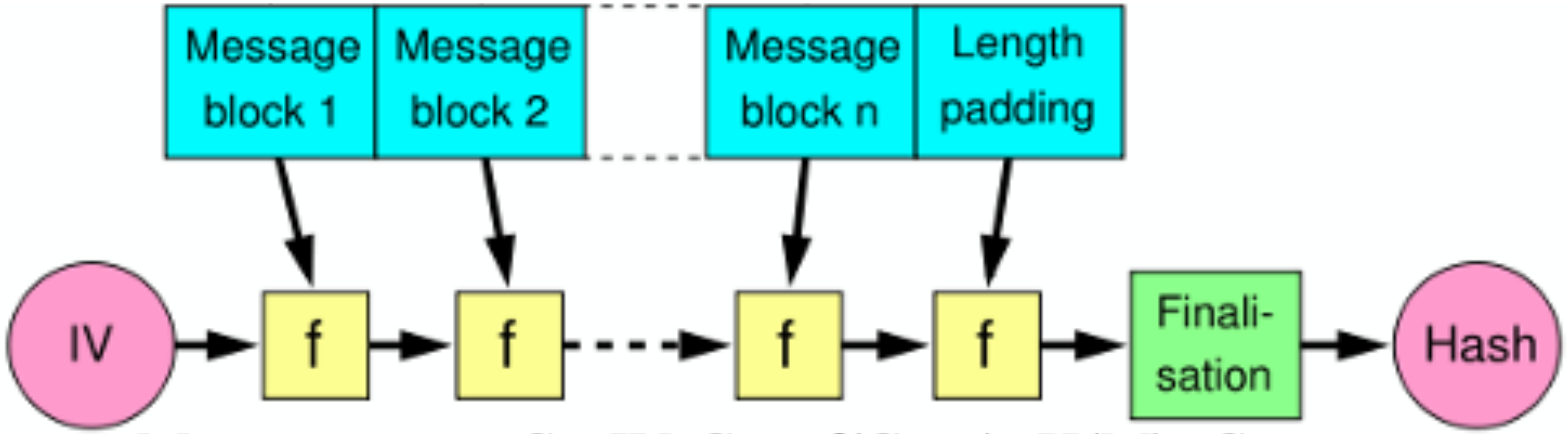
$$\mathrm{d} = 0$$
$$\text{then, } \mathrm{X(d)} = \mathrm{h(k} \oplus 0) = h(k)$$

- There are two possible explanations
  - k is constant (which it is not)
  - X(d) knows or receives k from input (which by definition it does not)
  - ... a contradiction.

Constructing MAC from Hash

- Message is divided into fixed-size blocks and padded
- Uses a compression function f, which takes a chaining variable (of size of hash output) and a message block, and outputs the next chaining variable
- Final chaining variable is the hash value

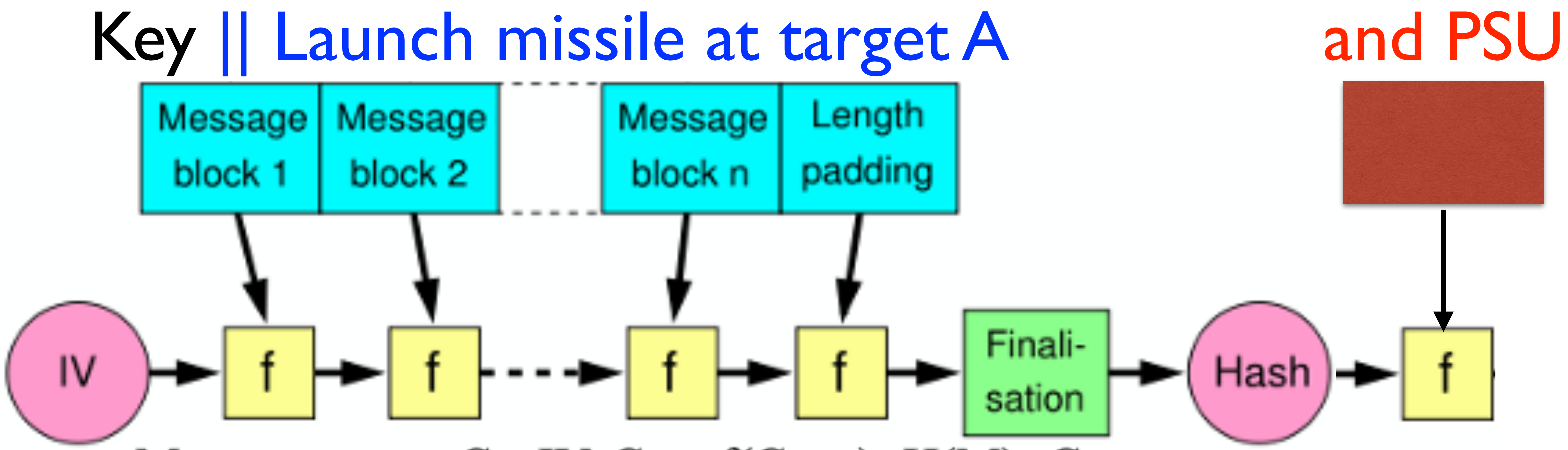
Key || Launch missile at target A



$M=m_1m_2\ldots m_n;\ C_0=IV,\ C_{i+1}=f(C_i,m_i);\ H(M)=C_n$

Hash Length Extension Attack

- Message is divided into fixed-size blocks and padded
- Uses a compression function f, which takes a chaining variable (of size of hash output) and a message block, and outputs the next chaining variable
- Final chaining variable is the hash value



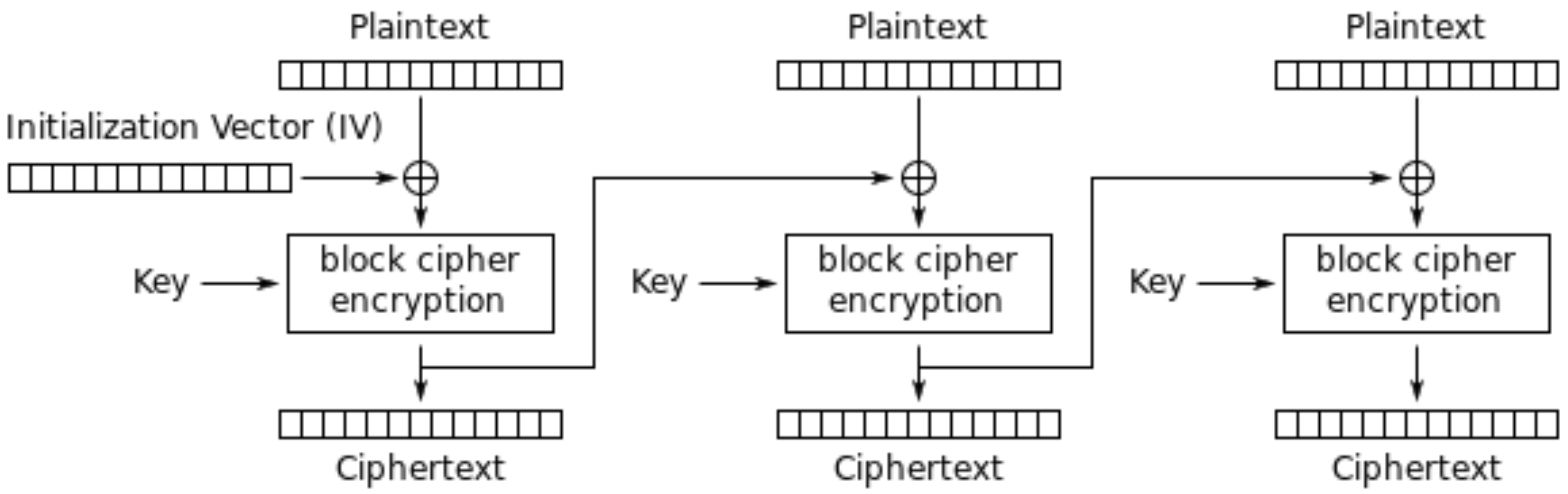
$M=m_1m_2\ldots m_n$; $C_0=IV$, $C_{i+1}=f(C_i,m_i)$; $H(M)=C_n$

HMAC

- MAC that meets the following properties
  - Collision-resistant
  - Attacker cannot compute a proper digest without knowing K
    - Even if attacker can see an arbitrary number of digests H(k+x)
- Simple MAC has a flaw
  - Block hash algorithms mean that new content can be added
  - Turn H(K+m) to H(K+m+m’) where m’ is controlled by an attacker
- HMAC(K, d) = H(K + H(K + d))
  - Attacker cannot extend MAC as above
  - Prove it to yourself

CBC-MAC

- You can also produce a MAC using a symmetric encryption function in CBC mode
  - Encryption in CBC produces ciphertext that is dependent on all prior plaintext blocks
  - Last block of ciphertext is suitable as a MAC
    - Use different key than for encryption



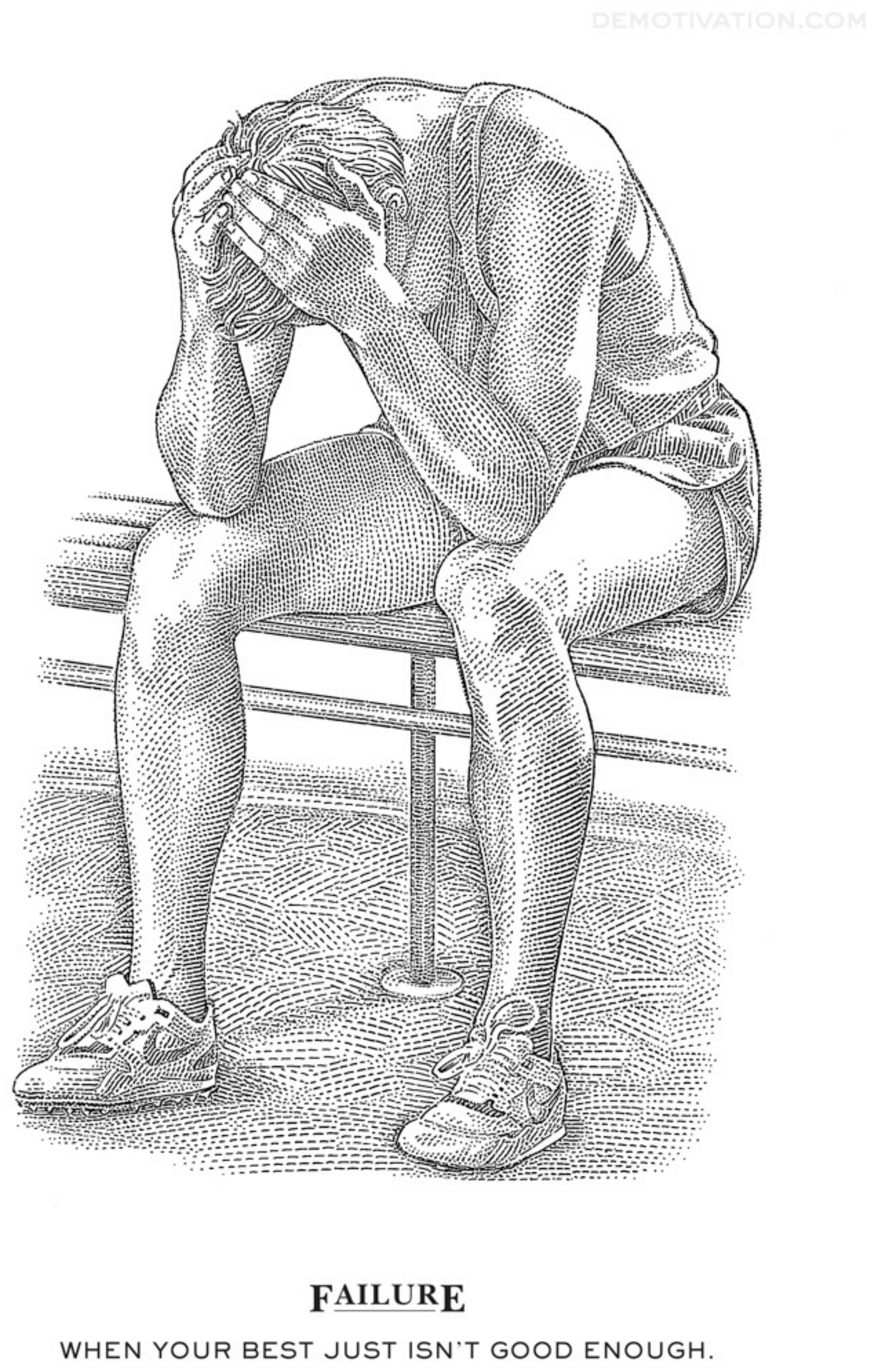
Cipher Block Chaining (CBC) mode encryption

Basic truths of cryptography …

- Cryptography is not frequently the source of security problems
  - Algorithms are well known and widely studied
    - Use of crypto commonly is … (e.g., WEP)
  - Vetted through crypto community
  - Avoid any "proprietary" encryption
  - Claims of "new technology" or "perfect security" are almost assuredly snake oil

Why Cryptosystems Fail

- In practice, what are the causes of cryptosystem failures
  - Not crypto algorithms typically



Case Study

- ATM Systems
  - Some public data
  - High value information
  - Of commercial enterprises, banks have most interest in security
- How do they work?
  - Card: With account number
  - User: Provides PIN
  - ATM: Verifies that PIN corresponds to encryption of account number with PIN key (offset can be used)
- Foundation of security
  - PIN key (for ATM) and PIN (for users)

Simple Fraud

- Insiders
  - Make an extra card; special ops allow debit of any acct
- Outsiders
  - Shoulder surfing; fake ATMs; replay "pay" response
- PIN Keys
  - Weak entropy of PIN keys
- User-chosen PINs
  - Bad; Store encrypted in a file (find match); Encrypted on card
- Italy
  - Fake ATMs; Offline ATMs (attack all at once)

Products Have Problems

- Despite well understood crypto foundations, products don't always work securely
  - Leak secrets due to encryption in software
  - Incompatibilities (borrow my terminal)
  - Poor product design
    - Backdoors enabled, non-standard crypto, lack of entropy, etc.
  - Sloppy operations
    - Ignore attack attempts, share keys, procedures are not defined or followed
  - Cryptanalysis sometimes
    - Home-grown algorithms!, improper parameters, cracking DES

Problems

- Systems may work in the lab/theory, but
  - Are difficult to use in practice
  - Counter-intuitive
  - Rewards aren't clear
  - Correct usage is not clear
  - Too many secrets ultimately

- Fundamentally, two problems
  - Too complex to use
  - No way to determine if use is correct

What Can We Do?

- Anderson suggests
  - Determine exactly what can go wrong
    - Find all possible failure modes
  - Put in safeguards
    - Describe how preventions protect system
  - Correct implementation of safeguards
    - Implementation of preventions meets requirements
  - Decisions left to people are small in number and clearly understood
    - People know what to do

- Problems of security in general

Building systems with cryptography

- Use quality libraries
  - E.g., OpenSSL, Libgcrypt, Cryptlib, BouncyCastle
  - Find out what cryptographers think of a package
- 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