Basic Crypto Tools

Dr. Chen Zhang

Department of Computer Science
The Hang Seng University of Hong Kong

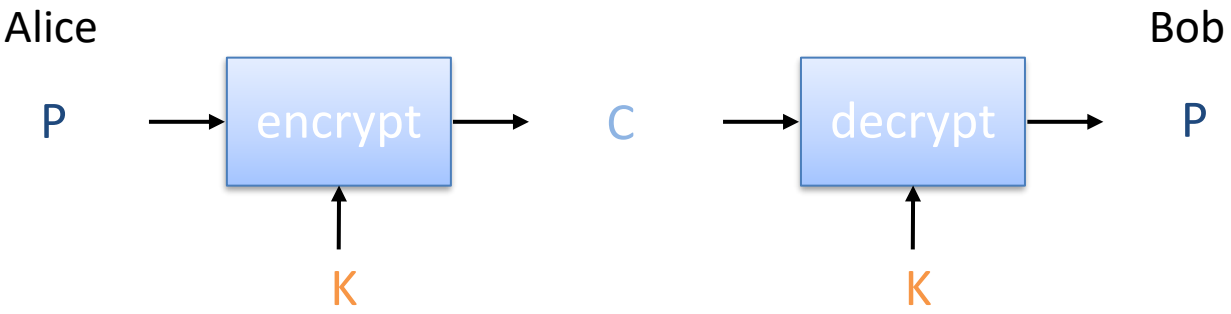
Slides partially adapted from lecture notes by M. Goodrich&R. Tamassia, W. Stallings&L. Brown, and Dan Boneh.

Symmetric Cryptography

Assumes parties already
share a secret key

Symmetric Cryptosystem

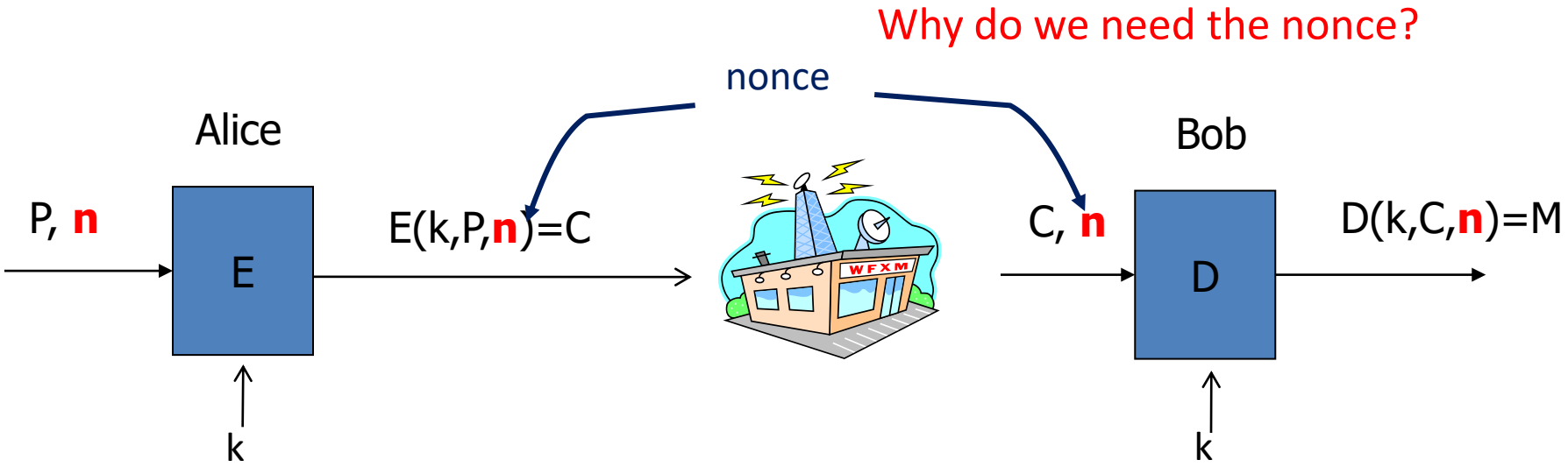
- Scenario
  - Alice wants to send a message (plaintext P) to Bob.
  - The communication channel is insecure and can be eavesdropped
  - If Alice and Bob have previously agreed on a symmetric encryption scheme and a secret key K, the message can be sent encrypted (ciphertext C)
- Issues
  - What is a good symmetric encryption scheme?
  - What is the complexity of encrypting/decrypting?
  - What is the size of the ciphertext, relative to the plaintext?



Basics

- Notation
  - Secret key K
  - Encryption function $E_k(P)$
  - Decryption function $D_K(C)$
  - Plaintext length typically the same as ciphertext length
  - Encryption and decryption are PRP, i.e., pseudorandom permutation functions (bijections), on the set of all n-bit arrays
- Efficiency
  - functions E_K and D_K should have efficient algorithms
- Consistency
  - Decrypting the ciphertext yields the plaintext
  - $D_K(E_K(P)) = P$

Basics (Cont'd)



E, D: cipher

P, C: plaintext, ciphertext

k: secret key (e.g. 128 bits)

n: nonce

Encryption algorithm is publicly known

- Never use a proprietary cipher

Use Cases

Single use key (one time key) :

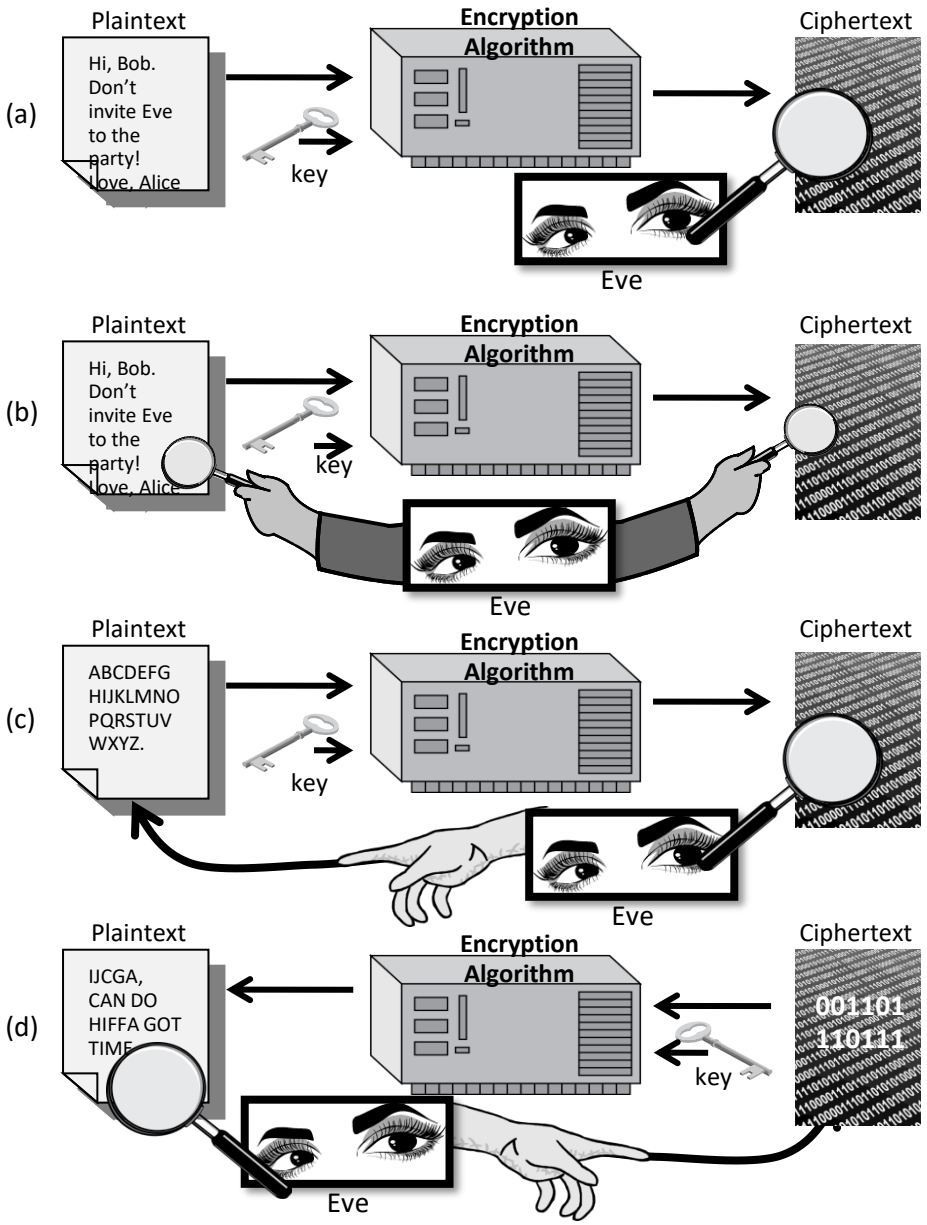
- Key is only used to encrypt one message
  - Encrypted email: new key generated for every email
- No need for nonce (set to 0)

Multi use key (many time key):

- Key used to encrypt multiple messages
  - SSL: same key used to encrypt many packets
- Need either *unique* nonce or *random* nonce

Cryptographic Attacks

- Attacker may have
  a) collection of ciphertexts (ciphertext only attack)
  b) collection of plaintext/ciphertext pairs (known plaintext attack)
  c) collection of plaintext/ciphertext pairs for plaintexts selected by the attacker (chosen plaintext attack, CPA)
  d) collection of plaintext/ciphertext pairs for ciphertexts selected by the attacker (chosen ciphertext attack, CCA)



In-Class Exercise 1

- Alice and Bob work for a real estate agent. Eve is an ex security engineer.
  - Q1. Eve has tricked Alice into decrypting a bunch of ciphertexts that Alice encrypted last month but forgot about. What type of attack is Eve employing?
  - Q2. Eve has an antenna that can pick up Alice's encrypted cell phone conversations. What type of attack is Eve employing?
  - Q3. Eve has bet Bob that she can figure out the AES secret key he shares with Alice if he will simply encrypt 20 messages for Eve using that key. For some unknown reason, Bob agrees. Eve gives him 20 messages, which he then encrypts and emails back to Eve. What kind of attack is Eve using here?

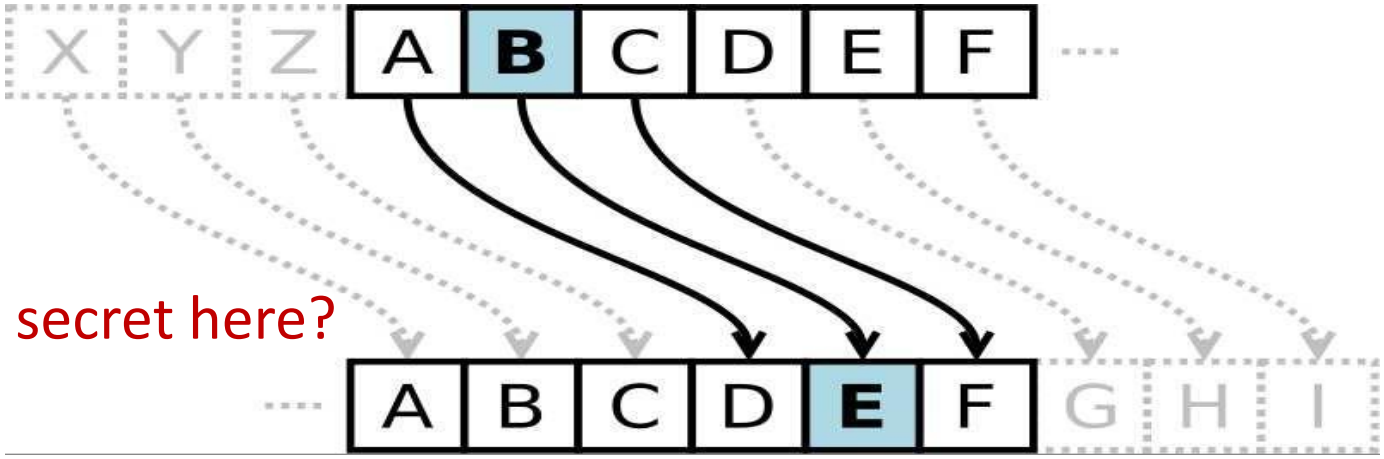
Brute-Force Attack

- Try all possible keys k and determine if $D_k(C)$ is a likely plaintext
  - The key k is a secret
  - Requires some knowledge of the structure of the plaintext (e.g., PDF file or email message)
- To defend against the brute-force attack, Key k should be a sufficiently long random value to make exhaustive search attacks unfeasible

If the following symmetric encryption schemes are good?

Substitution Ciphers

- Each letter is uniquely replaced by another.
- Caesar Cipher: one special type of substitution ciphers.
  - Replace each letter with the one “three over” in the alphabet.



What’s the secret here?

Caesar Cipher

- Secret is the number of characters to shift (0,1,…,25).
- Keyspace: the set of all valid, possible, distinct keys of a given cryptosystem.

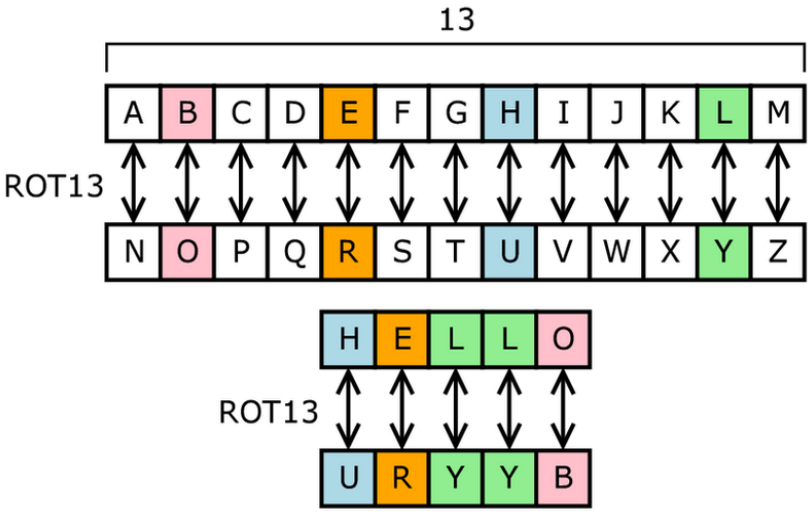
What's the size of Keyspace?

Caesar Cipher

- Secret is the number of characters to shift (0,1,…,25).
- The size of Keyspace: 26

- One popular substitution "cipher" for some Internet posts is ROT13.

https://rot13.com/



Vulnerable to brute-force attacks!

Simple Substitution Ciphers

- Each letter is uniquely replaced by another.
- There are 26! possible substitution ciphers (secrets).
- The size of Keyspace is more than 4.03×10^{26}

Hard to crack by brute-force attacks!

Is it secure?

Frequency Analysis

- Letters in a natural language, like English, are not uniformly distributed.
- Knowledge of letter frequencies, including pairs and triples can be used in cryptologic attacks against substitution ciphers.

a:	8.05%	b:	1.67%	c:	2.23%	d:	5.10%
e:	12.22%	f:	2.14%	g:	2.30%	h:	6.62%
i:	6.28%	j:	0.19%	k:	0.95%	l:	4.08%
m:	2.33%	n:	6.95%	o:	7.63%	p:	1.66%
q:	0.06%	r:	5.29%	s:	6.02%	t:	9.67%
u:	2.92%	v:	0.82%	w:	2.60%	x:	0.11%
y:	2.04%	z:	0.06%				

Letter frequencies in the book *The Adventures of Tom Sawyer*, by Twain.

Substitution Boxes

- Substitution can also be done on binary numbers.
- Such substitutions are usually described by substitution boxes, or S-boxes.

	00	01	10	11
00	0011	0100	1111	0001
01	1010	0110	0101	1011
10	1110	1101	0100	0010
11	0111	0000	1001	1100

(a)

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

(b)

Figure 8.3: A 4-bit S-box (a) An S-box in binary. (b) The same S-box in decimal.

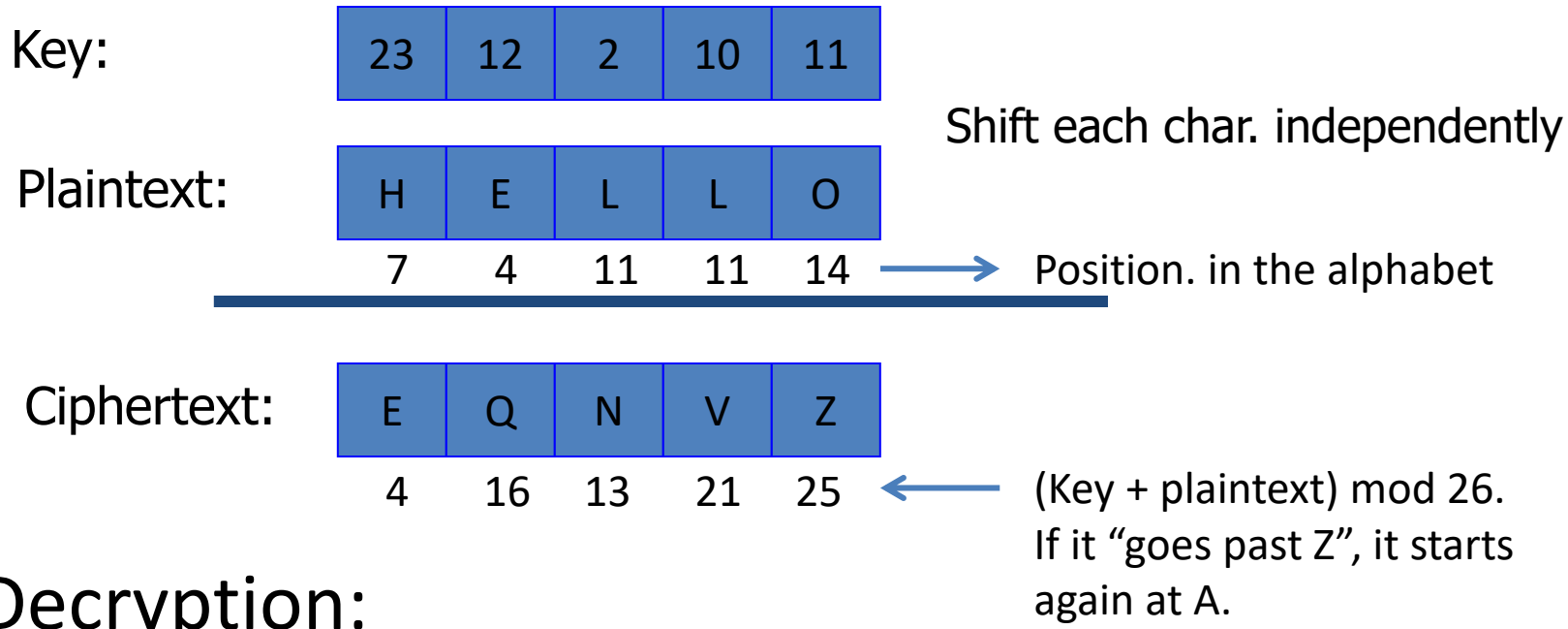
One-Time Pads

- There is one type of substitution cipher that is absolutely unbreakable.
  - The **one-time pad** was invented in 1917 by Joseph Mauborgne and Gilbert Vernam
  - We use a block of shift keys, (k_1, k_2, . . . , k_n), to encrypt a plaintext, P, of length n, with each shift key being chosen **uniformly at random**.
- Since each shift is random, every ciphertext is equally likely for any plaintext.

First example: One Time Pad

(single use key; over 26 **English characters**)

- Vernam (1917)



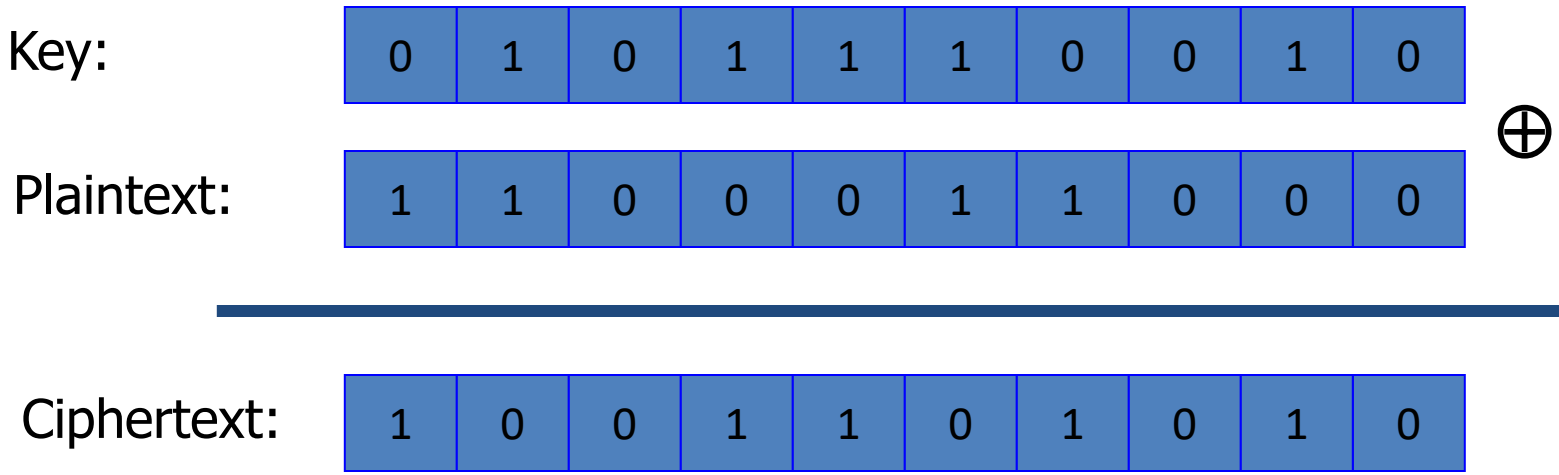
- Decryption:
  - (Ciphertext – Key) mod 26

Does it vulnerable to frequency attacks?

Second example: One Time Pad

(single use key; **the binary version**)

- Vernam (1917)



$C = E(k,P) = P \oplus k$

$P = D(k,C) = C \oplus k$

- Shannon '49:
  - OTP is "secure" against ciphertext-only attacks

Recall XOR

True when one (but not both) input is true

Truth table

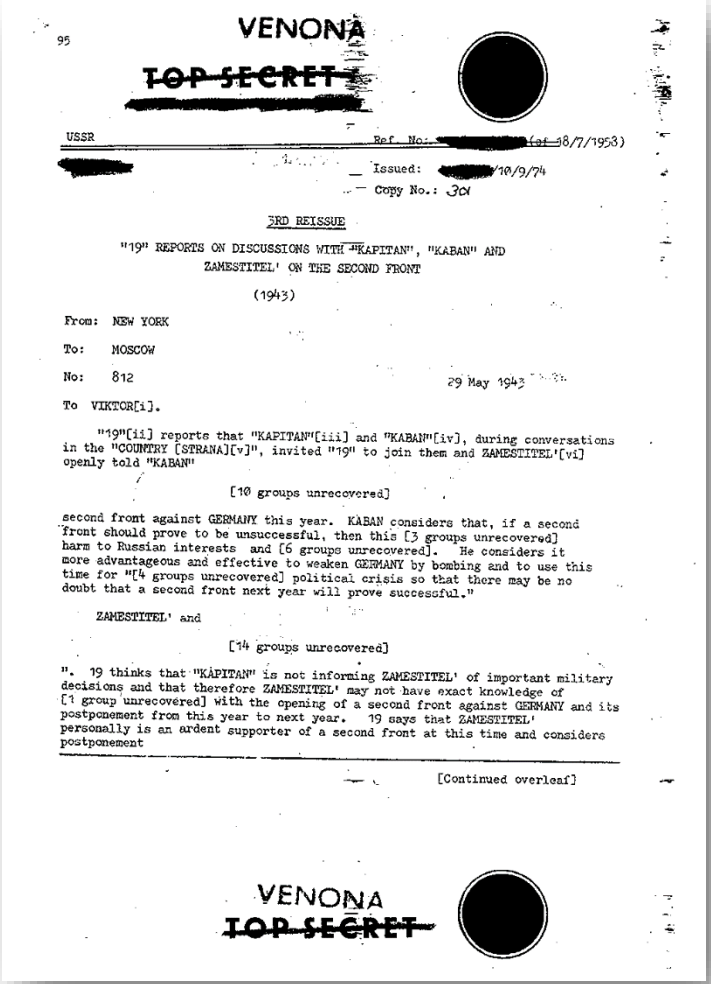
Input 1	Input 2	Output
1	1	0
1	0	1
0	1	1
0	0	0

An important feature: **$A \oplus B \oplus A = B$**

Is the One Time Pad Method Good Enough?

Weaknesses of the One-Time Pad

- The key has to be **as long as the plaintext**
- Keys can **never be reused**
  - Repeated use of one-time pads allowed the U.S. to break some of the communications of Soviet spies during the Cold War.
- Problems in generation & safe distribution of keys.



95

VENONA

~~TOP SECRET~~

USSR

Ref No: [redacted] (of 8/7/1953)

[redacted] Issued: [redacted]/10/9/74

Copy No.: 304

3RD REISSUE

"19" REPORTS ON DISCUSSIONS WITH "KAPITAN", "KABAN" AND ZAMESTITEL' ON THE SECOND FRONT

(1943)

From: NEW YORK

To: MOSCOW

No: 812

29 May 1943

To VIKTOR[i].

"19"[ii] reports that "KAPITAN"[iii] and "KABAN"[iv], during conversations in the "COUNTRY [STRANA][v]", invited "19" to join them and ZAMESTITEL'[vi] openly told "KABAN"

[10 groups unrecovered]

second front against GERMANY this year. KABAN considers that, if a second front should prove to be unsuccessful, then this [3 groups unrecovered] harm to Russian interests and [6 groups unrecovered]. He considers it more advantageous and effective to weaken GERMANY by bombing and to use this time for "[4 groups unrecovered] political crisis so that there may be no doubt that a second front next year will prove successful."

ZAMESTITEL' and

[14 groups unrecovered]

". 19 thinks that "KAPITAN" is not informing ZAMESTITEL' of important military decisions and that therefore ZAMESTITEL' may not have exact knowledge of [1 group unrecovered] with the opening of a second front against GERMANY and its postponement from this year to next year. 19 says that ZAMESTITEL' personally is an ardent supporter of a second front at this time and considers postponement

[Continued overleaf]

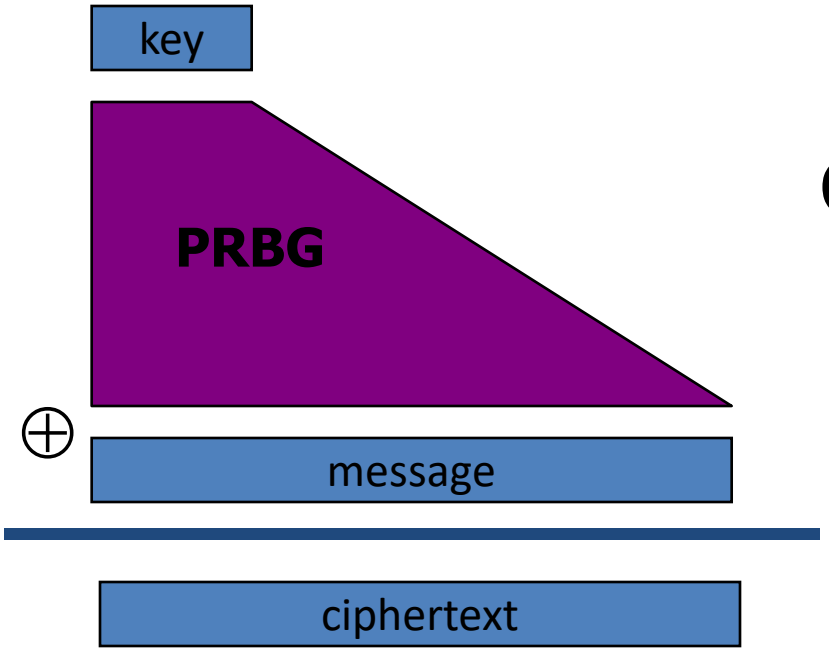
VENONA

~~TOP SECRET~~

Stream ciphers (single use key)

Problem: One-Time Pad key is as long as the message

Solution: Pseudo random key -- stream ciphers



$C \leftarrow PRBG(k) \oplus P$

Pseudo-random bit generator

Stream ciphers: RC4 (113MB/sec) , SEAL (293MB/sec)

Stream Cipher

- Key stream
  - key input to a pseudorandom bit generator
    - produces stream of random like numbers
  - Pseudo-random sequence of bits S = S[0], S[1], S[2], …
    - unpredictable without knowing input key
  - Can be generated on-line one bit (or byte) at the time
- Stream cipher
  - XOR the message with the key stream $C[i] = S[i] \oplus P[i]$
  - Suitable for plaintext of arbitrary length generated **on the fly**, e.g., media stream
  - processes input elements continuously
- Faster and use far less code

Dangers in using stream ciphers

One time key !! “Two time pad” is insecure:

$$\begin{cases} C_1 \leftarrow P_1 \oplus \text{PRBG}(k) \\ C_2 \leftarrow P_2 \oplus \text{PRBG}(k) \end{cases}$$

Eavesdropper does:

$$C_1 \oplus C_2 \rightarrow P_1 \oplus P_2$$

Known-plaintext attack is very dangerous if key is repeated used:

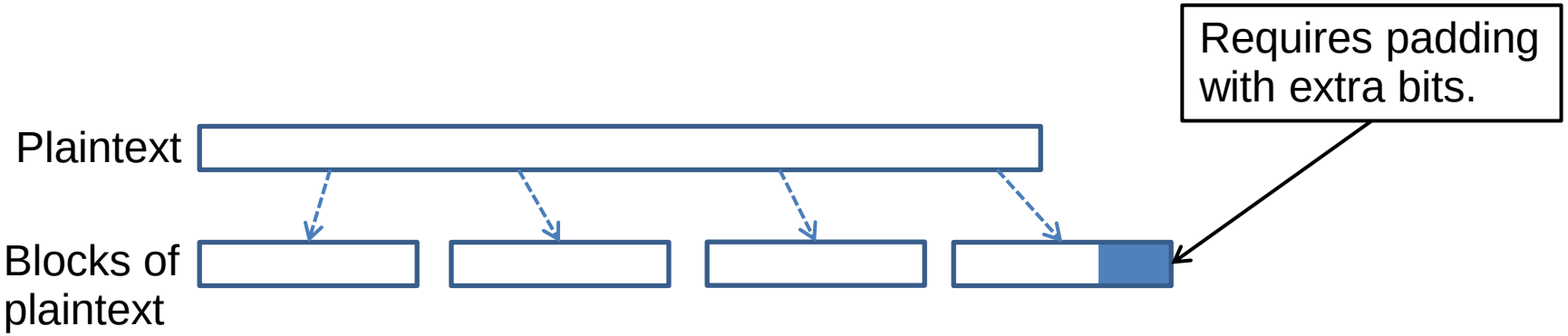
$$P_1 \oplus P_2 \rightarrow P_1, P_2$$

Block Cipher

The block cipher processes fixed-size blocks simultaneously, as opposed to a stream cipher, which encrypts data one bit at a time.

Block Ciphers

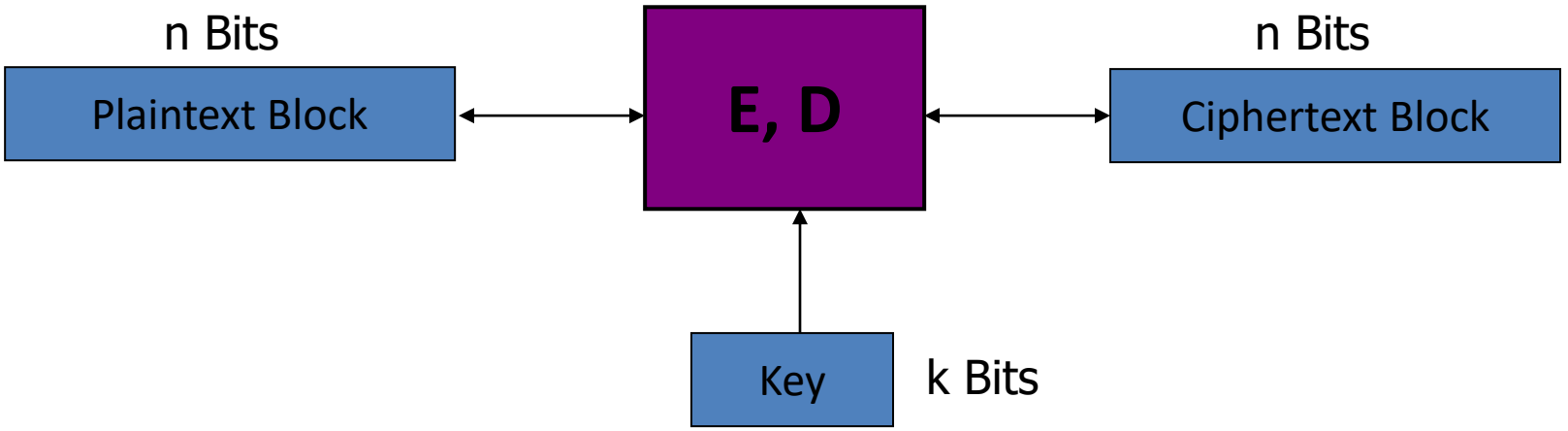
- In a **block cipher:**
  - Plaintext and ciphertext have fixed length b (e.g., 128 bits)
  - A plaintext of length n is partitioned into a sequence of m **blocks**, P[0], …, P[m–1], where $n \leq bm < n + b$
- Each message is divided into a sequence of blocks and encrypted or decrypted in terms of its blocks.



Padding

- Block ciphers require the length n of the plaintext to be a multiple of the block size b
- Padding the last block needs to be unambiguous (cannot just add zeroes)
- When the block size and plaintext length are a multiple of 8, a common padding method (PKCS5) is a sequence of identical bytes, each indicating the length (in bytes) of the padding
- Example for b = 128 (16 bytes)
  - Plaintext: “Roberto” (7 bytes)
  - Padded plaintext: “Roberto999999999” (16 bytes), where 9 denotes the number and not the character

Block Ciphers



Typical examples:

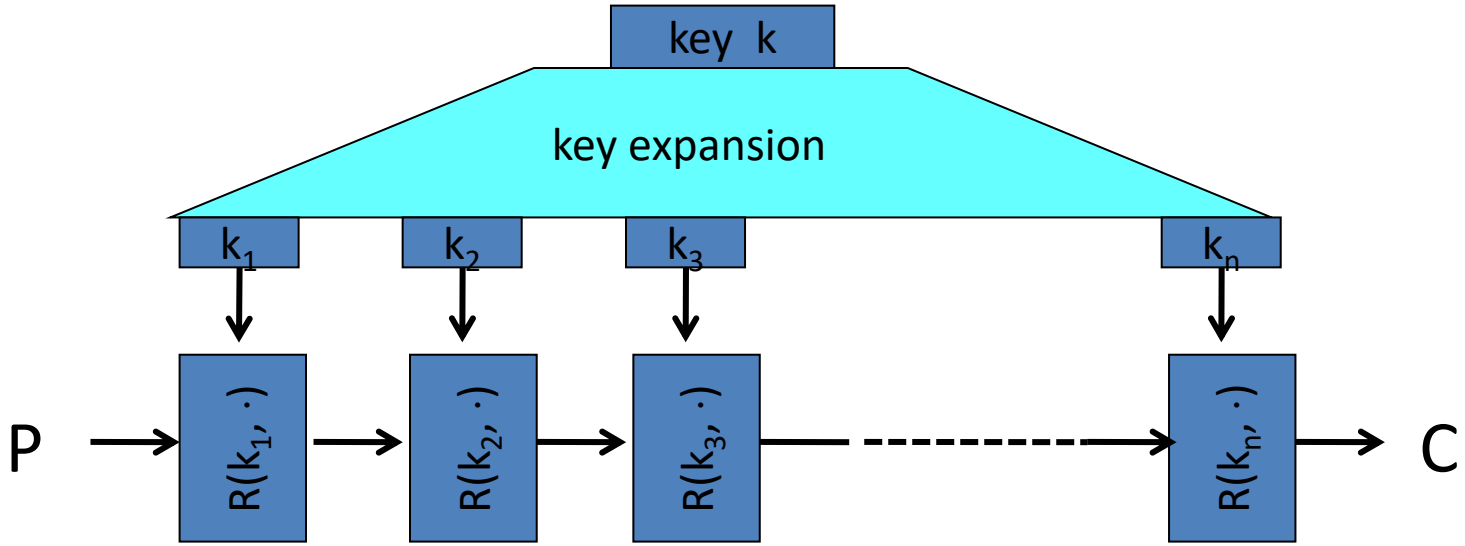
1. 3DES: block size b= 64 bits, k = 168 bits (56*3)
2. AES: block size b=128 bits, k = 128, 192, 256 bits

Block Ciphers in Practice

- Data Encryption Standard (DES)
  - Developed by IBM and adopted by NIST in 1977
  - 64-bit blocks and 56-bit keys
  - Small key space makes exhaustive search attack feasible since late 90s
- Triple DES (3DES)
  - Nested application of DES with three different keys KA, KB, and KC
  - Effective key length is 168 bits, making exhaustive search attacks unfeasible
  - $C = E_{KC}(D_{KB}(E_{KA}(M)))$; $P = D_{KA}(E_{KB}(D_{KC}(C)))$
  - Equivalent to DES when KA=KB=KC (backward compatible)
- Advanced Encryption Standard (AES)
  - Selected by NIST in 2001 through **open international competition and public discussion**.
  - 128-bit blocks and several possible key lengths: 128, 192 and 256 bits
  - Exhaustive search attack not currently possible
  - AES-256 is the symmetric encryption algorithm of choice

How to encrypt each Block?

Block Ciphers Built by Iteration



R(k,P): round function

n: the number of iterations

for DES (n=16), for AES (n=10)

Building a block cipher

Input: (m, k)

Repeat simple "mixing" operation several times

- DES: Repeat 16 times

$$\begin{cases} P_L \leftarrow P_R \\ P_R \leftarrow P_L \oplus R(k, P_R) \end{cases}$$

*P_L and P_R denote the left and right part of plaintext P

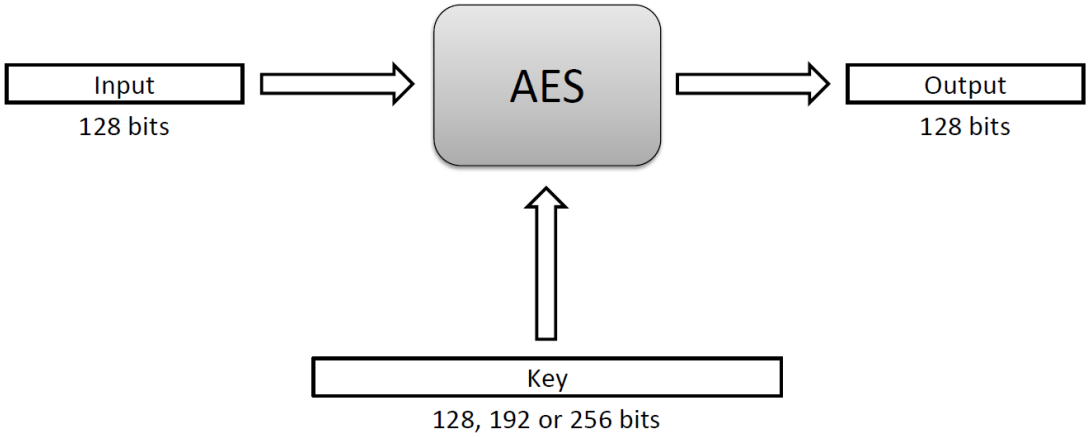
- AES-128: Mixing step repeated 10 times

Difficult to design: must resist subtle attacks

- differential attacks, brute-force, …

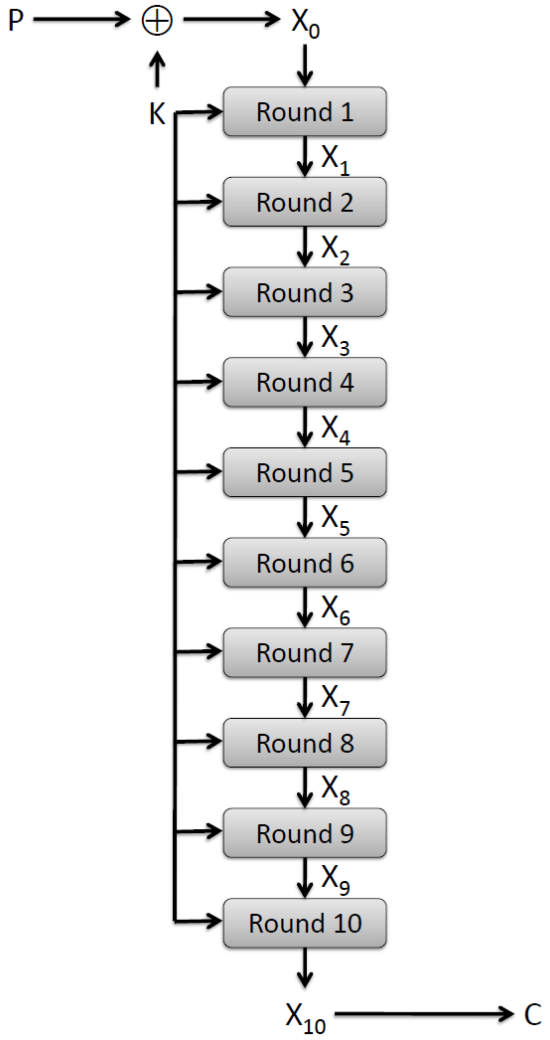
The Advanced Encryption Standard (AES)

- In 1997, the U.S. National Institute for Standards and Technology (NIST) put out a public call for a replacement to DES.
- It narrowed down the list of submissions to five finalists, and ultimately chose an algorithm that is now known as the **Advanced Encryption Standard** (**AES**).
- AES is a block cipher that operates on 128-bit blocks. It is designed to be used with **keys** that are 128, 192, or 256 bits long, yielding ciphers known as AES-128, AES-192, and AES-256.



AES Round Structure

- The 128-bit version of the AES encryption algorithm proceeds in ten rounds.
- Each round performs an invertible transformation on a 128-bit array, called **state**.
- The initial state X_0 is the XOR of the plaintext P with the key K:
- $X_0 = P \text{ XOR } K$.
- Round i (i = 1, …, 10) receives state X_{i-1} as input and produces state X_i.
- The ciphertext C is the output of the final round: $C = X_{10}$.



AES Rounds

- Each round is built from four basic steps:

1. **SubBytes step**: an S-box substitution step
2. **ShiftRows step**: a permutation step
3. **MixColumns step**: a matrix multiplication step
4. **AddRoundKey step**: an XOR step with a **round key** derived from the 128-bit encryption key

AES Round Structure

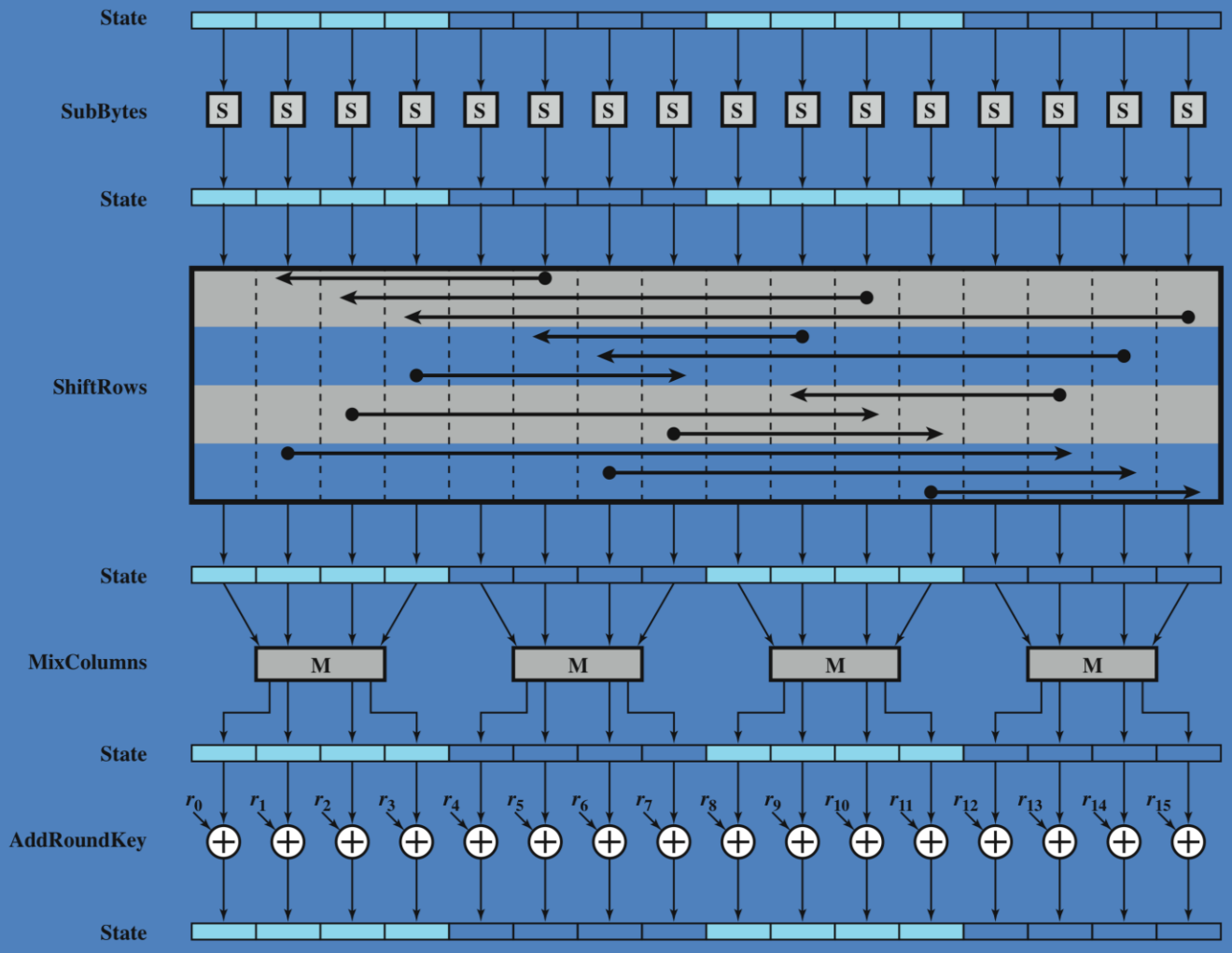


Figure 20.4 AES Encryption Round

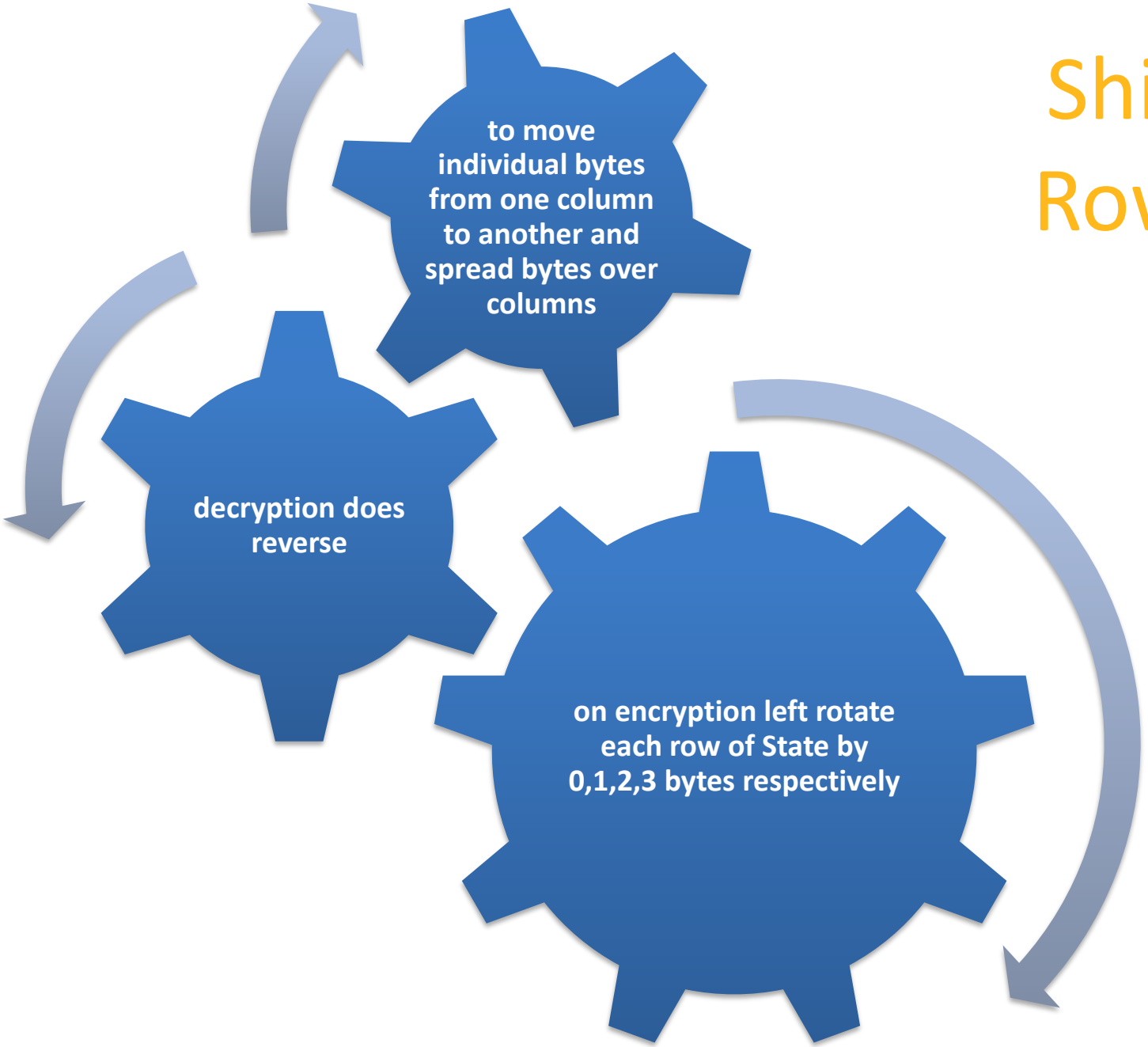
(a) S-box

		y															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
x	0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
	1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
	2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
	3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
	4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
	5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
	6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
	7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
	8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
	9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
	A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
	B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
	C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
	D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
	E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
	F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

(b) Inverse S-box

		y															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
x	0	52	09	6A	D5	30	36	A5	38	BF	40	A3	9E	81	F3	D7	FB
	1	7C	E3	39	82	9B	2F	FF	87	34	8E	43	44	C4	DE	E9	CB
	2	54	7B	94	32	A6	C2	23	3D	EE	4C	95	0B	42	FA	C3	4E
	3	08	2E	A1	66	28	D9	24	B2	76	5B	A2	49	6D	8B	D1	25
	4	72	F8	F6	64	86	68	98	16	D4	A4	5C	CC	5D	65	B6	92
	5	6C	70	48	50	FD	ED	B9	DA	5E	15	46	57	A7	8D	9D	84
	6	90	D8	AB	00	8C	BC	D3	0A	F7	E4	58	05	B8	B3	45	06
	7	D0	2C	1E	8F	CA	3F	0F	02	C1	AF	BD	03	01	13	8A	6B
	8	3A	91	11	41	4F	67	DC	EA	97	F2	CF	CE	F0	B4	E6	73
	9	96	AC	74	22	E7	AD	35	85	E2	F9	37	E8	1C	75	DF	6E
	A	47	F1	1A	71	1D	29	C5	89	6F	B7	62	0E	AA	18	BE	1B
	B	FC	56	3E	4B	C6	D2	79	20	9A	DB	C0	FE	78	CD	5A	F4
	C	1F	DD	A8	33	88	07	C7	31	B1	12	10	59	27	80	EC	5F
	D	60	51	7F	A9	19	B5	4A	0D	2D	E5	7A	9F	93	C9	9C	EF
	E	A0	E0	3B	4D	AE	2A	F5	B0	C8	EB	BB	3C	83	53	99	61
	F	17	2B	04	7E	BA	77	D6	26	E1	69	14	63	55	21	0C	7D

Shift Rows


to move individual bytes from one column to another and spread bytes over columns
decryption does reverse
on encryption left rotate each row of State by 0,1,2,3 bytes respectively

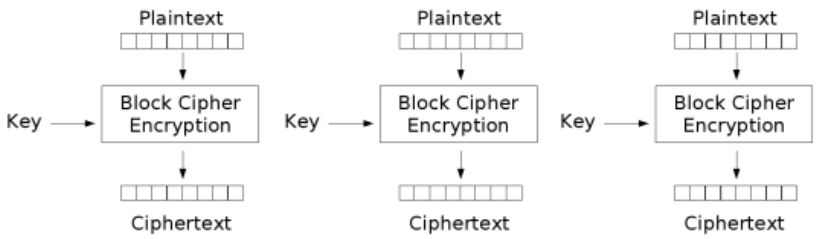
Mix Columns and Add Key

- mix columns
  - operates on each column individually
  - mapping each byte to a new value that is a function of all four bytes in the column
  - use of equations over finite fields
  - to provide good mixing of bytes in column
- add round key
  - simply XOR State with bits of expanded key
  - security from complexity of round key expansion and other stages of AES

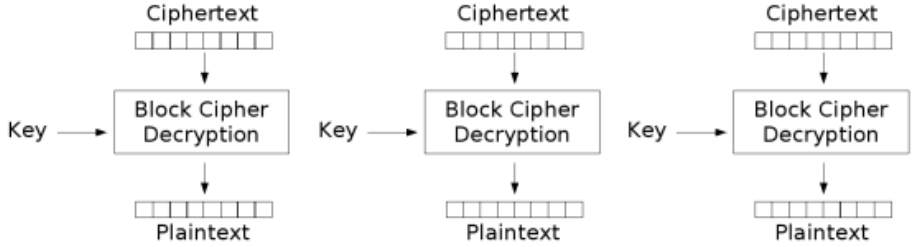
Modes of Operation

Block Cipher Modes

- A block cipher mode describes the way a block cipher encrypts and decrypts a sequence of message blocks.
- Electronic CodeBook (ECB) Mode (is the simplest):
  - Block P[i] encrypted into ciphertext block $C[i] = E_K(P[i])$
  - Block C[i] decrypted into plaintext block $P[i] = D_K(C[i])$



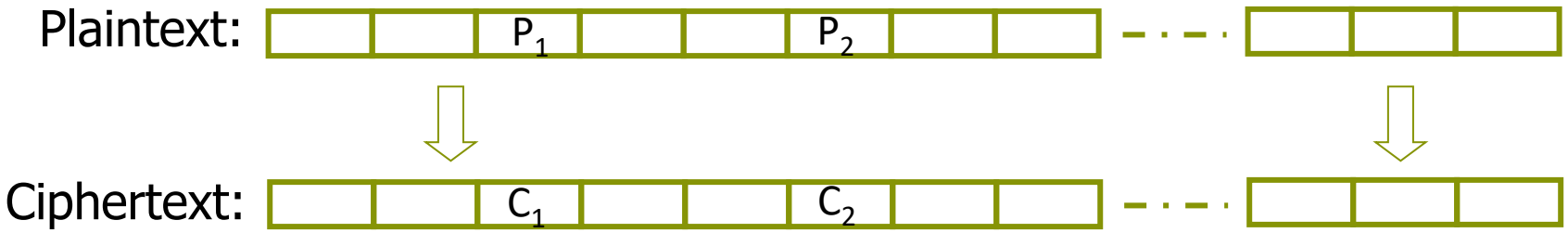
Electronic Codebook (ECB) mode encryption



Electronic Codebook (ECB) mode decryption

Incorrect use of block ciphers

- Electronic Code Book (ECB):

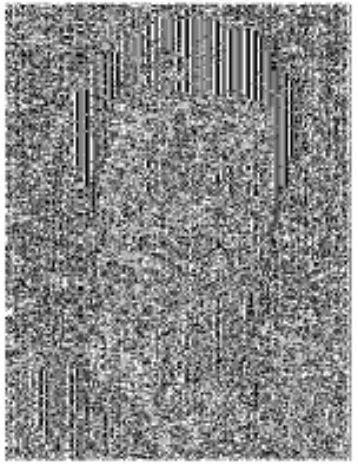


- Problem:
  - if $P_1=P_2$ then $C_1=C_2$

In pictures

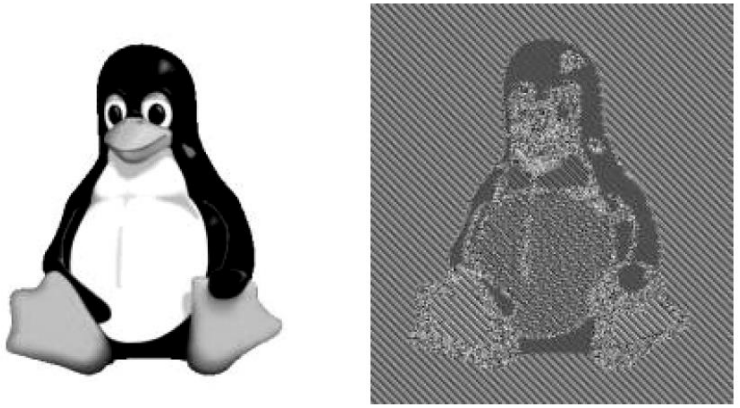
An example plaintext

Encrypted with AES in ECB mode



Strengths and Weaknesses of ECB

- Strengths:
  - Is very simple
  - Allows for parallel encryptions of the blocks of a plaintext
  - Can tolerate the loss or damage of a block
- Weakness:
  - Documents and images are not suitable for ECB encryption since patterns in the plaintext are repeated in the ciphertext:



(a) (b)

Figure 8.6: How ECB mode can leave identifiable patterns in a sequence of blocks: (a) An image of Tux the penguin, the Linux mascot. (b) An encryption of the Tux image using ECB mode. (The image in (a) is by Larry Ewing, lewing@isc.tamu.edu, using The Gimp; the image in (b) is by Dr. Juzam. Both are used with permission via attribution.)

Attacks to ECB

- Change the order of blocks
  - The plaintext can be manipulated without knowing the key.
  - E.g., transfer money
    - block 1=account of the sender->C1
    - block 2=account of the receiver->C2
    - block 3=transfer amount-> C3
    - After encryption get C1, C2, C3. The attacker reverse the order of C1 and C2. Then after decryption, the account information of sender and receiver is reversed.

Cipher Block Chaining (CBC)

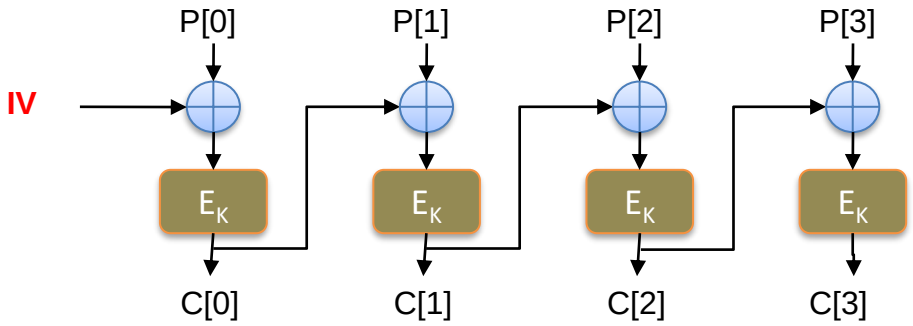
- The blocks connect with others like a chain.
- For a block, before encryption, it first performs the XOR operation with the ciphertext of the previous block.

Cipher Block Chaining (CBC) Mode (Cont'd)

- **Sequential** Encryption Process: $C[i] = E_K(C[i-1] \oplus P[i])$
- **Parallel** Decryption Process, if all ciphertext blocks are all available: $M[i] = C[i-1] \oplus D_K(C[i])$

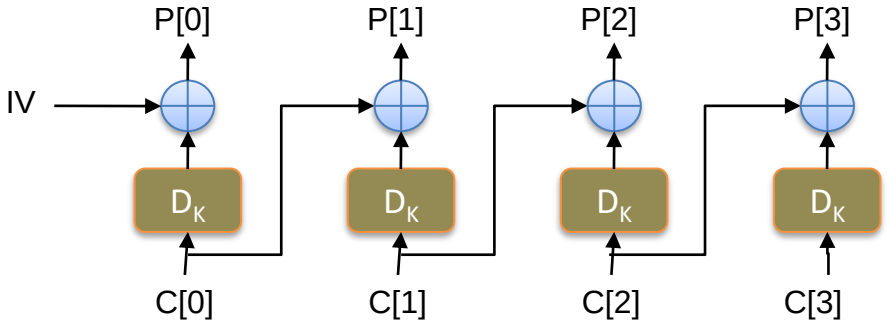
Cipher Block Chaining with **IV (initialization vector)**:

CBC Encryption:



C[i-1] is required before we generate C[i]

CBC Decryption:

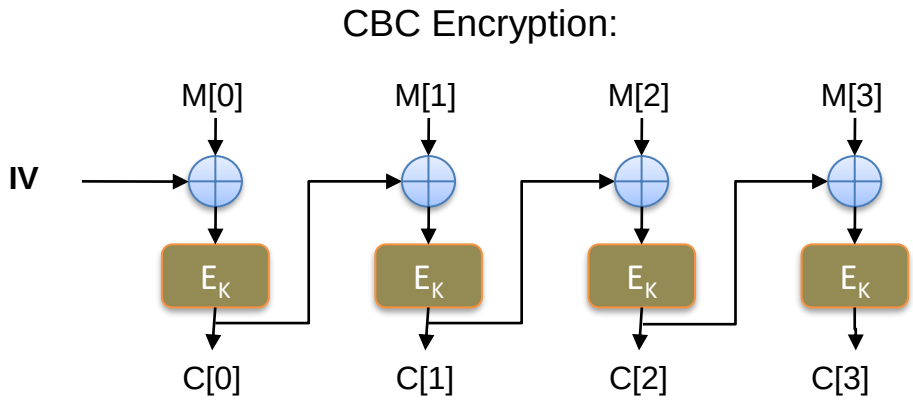


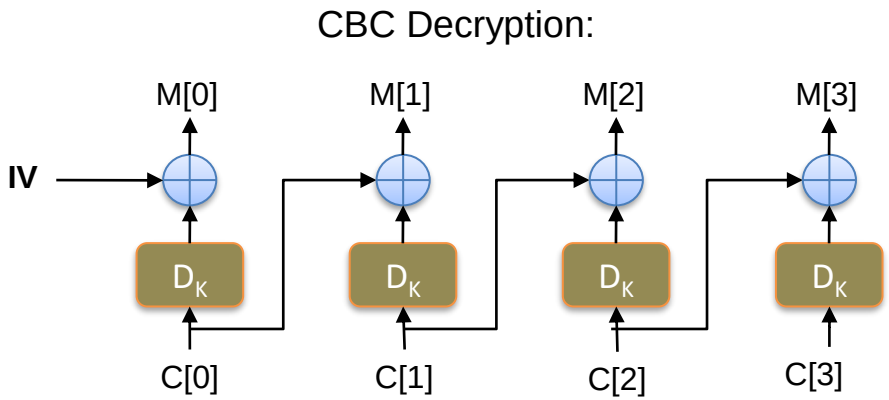
All P[i] can be generated simultaneously, when all C[i] are available.

Some Tolerance of Ciphertext Blocks Loss in CBC mode?

If C[i] is lost, it implies the decryption of blocks C[i] and C[i+1],i.e., P[i] and P[i+1] are lost.

Can the decryption of blocks C[i+2] still be done?



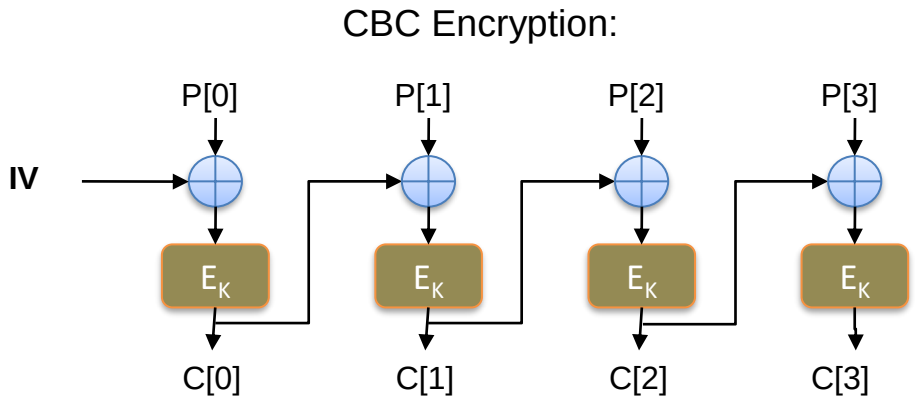


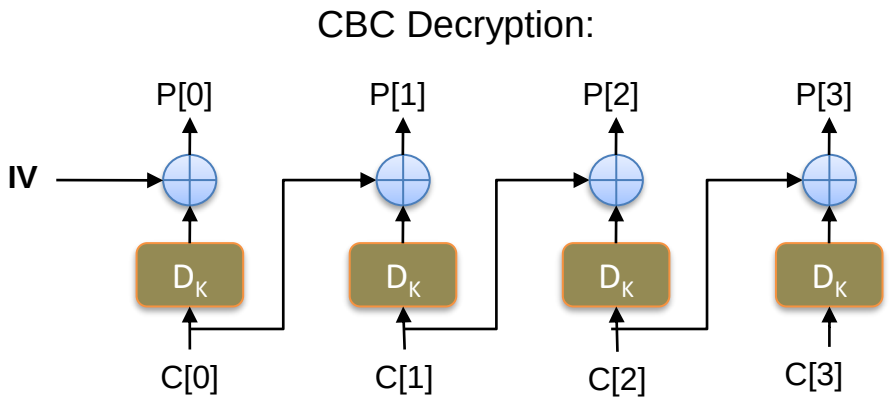
Some Tolerance of Ciphertext Blocks Loss in CBC mode?

If C[i] is lost, it implies the decryption of blocks C[i] and C[i+1],i.e., P[i] and P[i+1] are lost.

Can the decryption of blocks C[i+2] still be done?

Yes, since it relies only on C[i+1] and C[i+2].

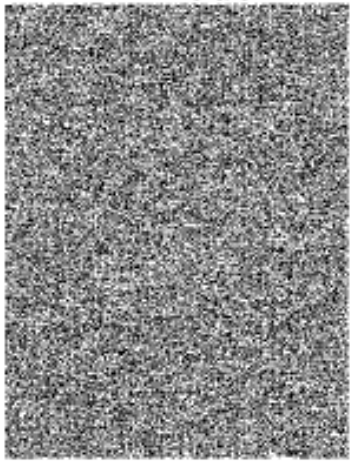




In pictures

An example plaintext

Encrypted with AES in CBC mode



Strengths and Weaknesses of CBC

- Strengths:
  - Doesn't show patterns in the plaintext
  - Is the most common mode
  - Is fast and relatively simple
- Weaknesses:
  - CBC requires the encryption of blocks to be done **sequentially**

Use cases: how to choose an IV

Single use key: no IV needed (IV=0)

Multi use key: (CPA Security)

Best: use a fresh *random* IV for every message (IV ← X), and send IV together with the ciphertexts.

Can use *unique and non-random* IV (e.g counter) [Bitlocker]

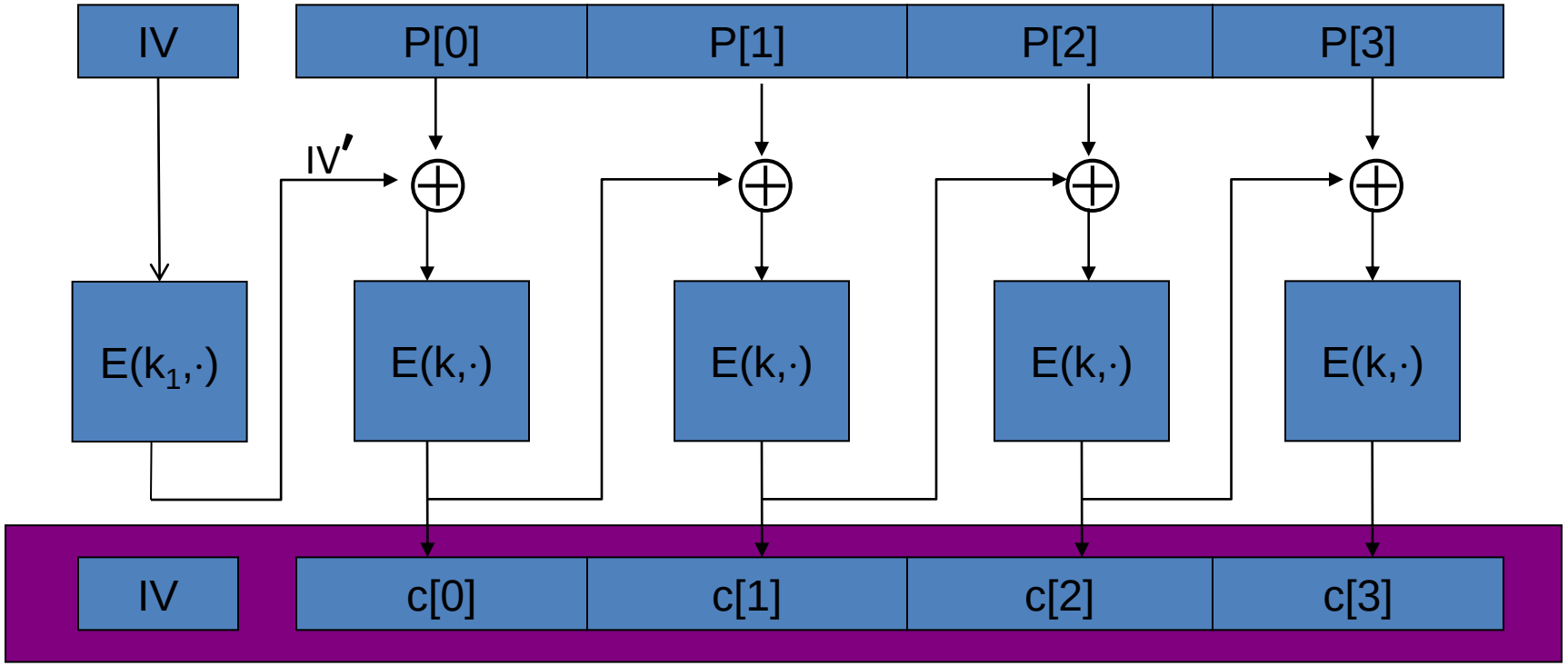
but then first step in CBC must be $IV' \leftarrow E(k_1, IV)$

benefit: may save transmitting IV with ciphertext

For example, Alice and Bob share both K and K_1, and synchronize the choice of initial counter IV, which might start with 1,2,3,… for message 1,2,3….

CBC with Unique IVs (nonce-based encryption)

Unique (non-random) IV means: (k,IV) pair is used for only one message. As non-random IV may be predictable, so use $E(k_1,\cdot)$ as PRF (pseudorandom func.)



ciphertext

Other (Stream) Modes of Operation

- block modes encrypt entire block
- may need to operate on smaller units
  - real time data
- convert block cipher into stream cipher
  - counter (CTR) mode
  - cipher feedback (CFB) mode
  - output feedback (OFB) mode
- use block cipher as some form of **pseudo-random number** generator

Counter (CTR) Mode

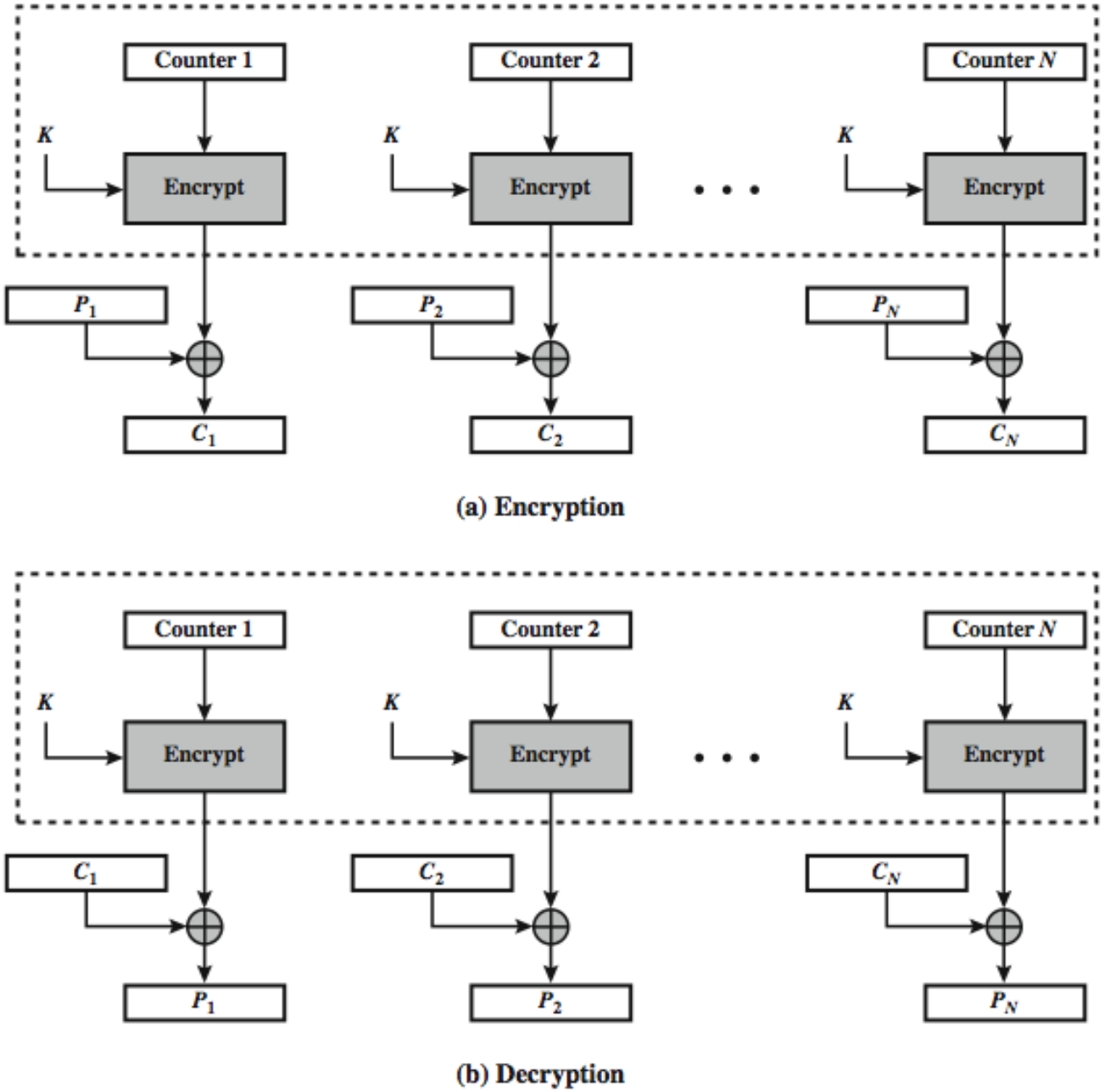
- Encrypts counter value
- must have a different key & counter value for every plaintext block (never reused)

$$O_i = E_K(i)$$

$$C_i = P_i \oplus O_i$$

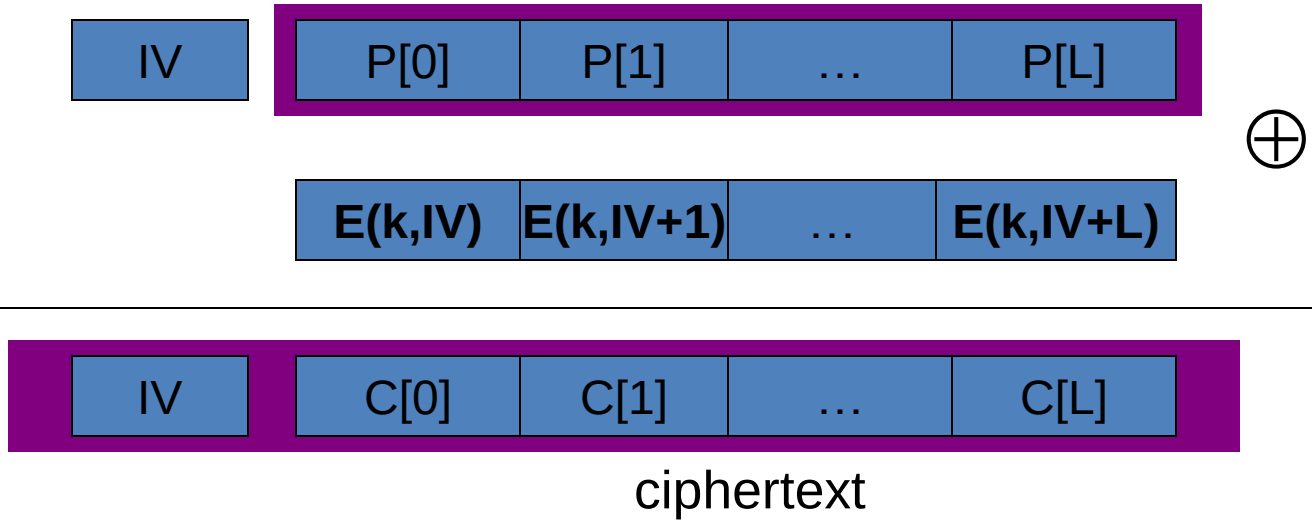
- uses: high-speed network encryptions

Counter (CTR)


Counter 1
Counter 2
Counter N
K
Encrypt
P_1
P_2
P_N
C_1
C_2
C_N
(a) Encryption
Counter 1
Counter 2
Counter N
K
Encrypt
C_1
C_2
C_N
P_1
P_2
P_N
(b) Decryption

Counter (CTR) Mode (Cont'd)

Counter mode with a random IV: (parallel encryption)



Advantages and Limitations of CTR

- efficiency
  - can do parallel encryptions in hardware and software
  - can preprocess in advance of need
  - good for bursty high speed links
- random access to encrypted data blocks
- provable security (good as other modes)
- but must ensure never reuse key/counter values, otherwise could break

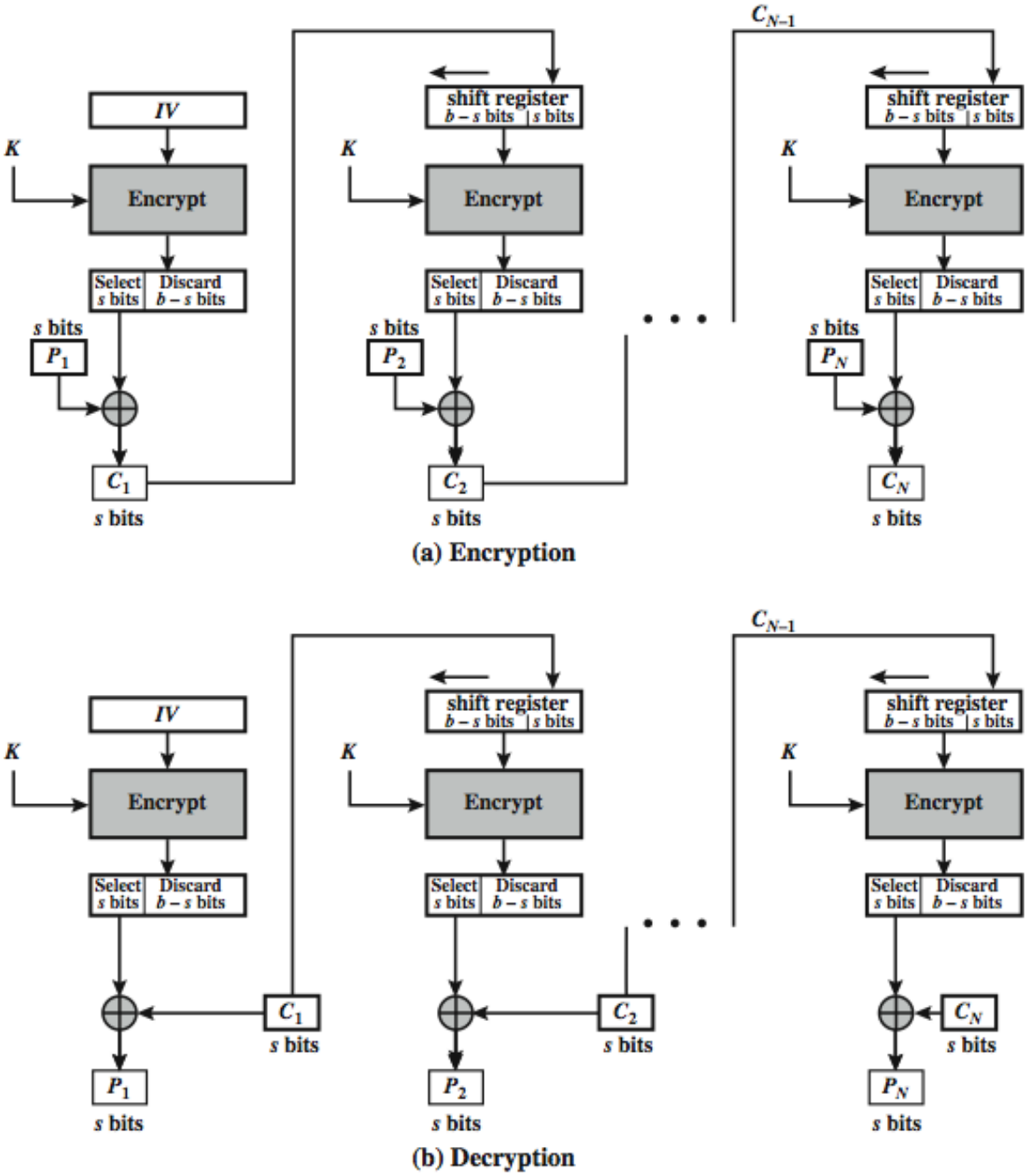
Cipher FeedBack (CFB)

- message is treated as a stream of bits
- added to the output of the block cipher
- result is feed back for next stage (hence name)
- standard allows any number of bit (1,8, 64 or 128 etc) to be feed back
  - denoted CFB-1, CFB-8, CFB-64, CFB-128 etc
- most efficient to use all bits in block (64 or 128)

$$C_i = M_i \oplus E_K(C_{i-1})$$
$$C_{-1} = IV$$

- uses: stream data encryption, authentication

s-bit Cipher FeedBack (CFB-s)



Advantages and Limitations of CFB

- appropriate when data arrives in bits/bytes
- most common stream mode
- note that the block cipher is used in **encryption** mode at **both** ends
- errors propogate for several blocks after the error

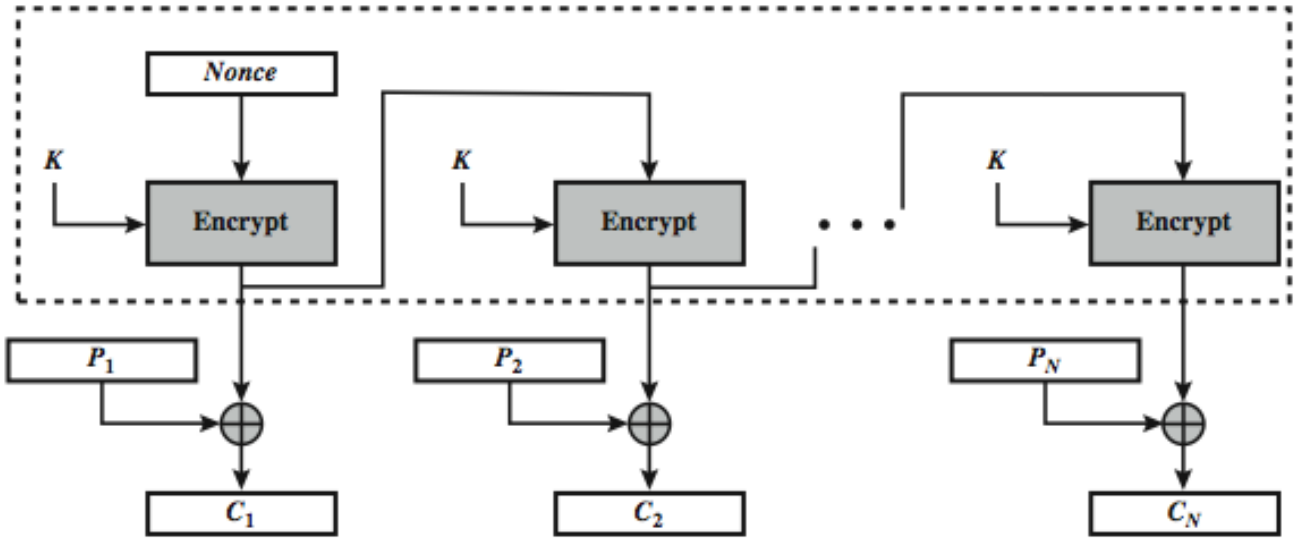
Output FeedBack (OFB)

- message is treated as a stream of bits
- output of cipher is added to message
- output is then feedback (hence name)
- feedback is independent of message
- can be computed in advance

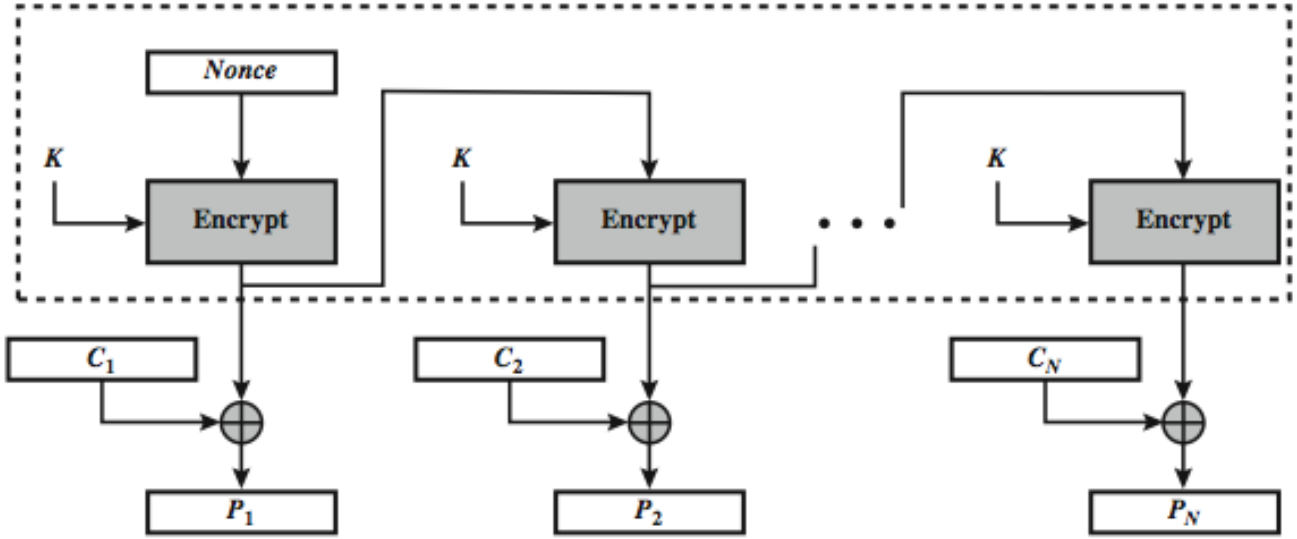
$$O_i = E_K(O_{i-1})$$
$$C_i = P_i \oplus O_i$$
$$O_{-1} = IV$$

- uses: stream encryption on noisy channels

Output FeedBack (OFB)



(a) Encryption



(b) Decryption

Advantages and Limitations of OFB

- needs an IV which is unique for each use
  - if ever reuse attacker can recover outputs
- bit errors do not propagate
- more vulnerable to message stream modification

Summary on Modes of Operation

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 DES output.	•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

Summary: Block vs Stream Ciphers

- Block cipher
  - processes the input one block of elements at a time
  - produces an output block for each input block
  - can reuse keys
  - more common and better analyzed
- Stream cipher
  - processes the encryption/decryption elements continuously, a bit or a byte at a time
  - primary advantage is that they are almost always faster and use far less code
  - pseudorandom stream is one that is unpredictable without knowledge of the input key