

COMPUTER SECURITY

SYMMETRIC-KEY
CRYPTOGRAPHY
(STREAM CIPHER)

Willy Sudiarto Raharjo

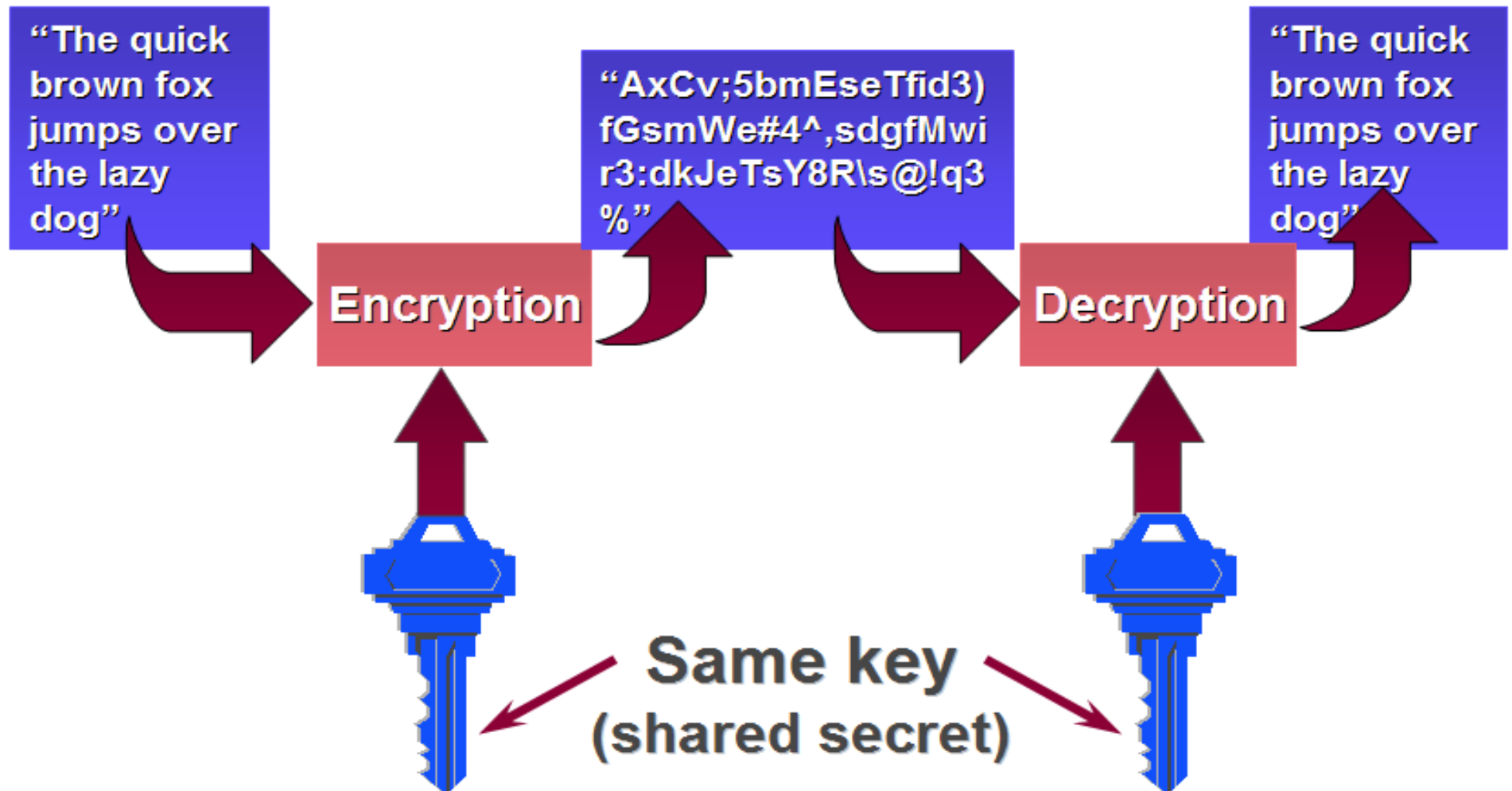
Teknik Informatika
Universitas Kristen Duta Wacana

Symmetric Key Cryptography

Plain-text input

Cipher-text

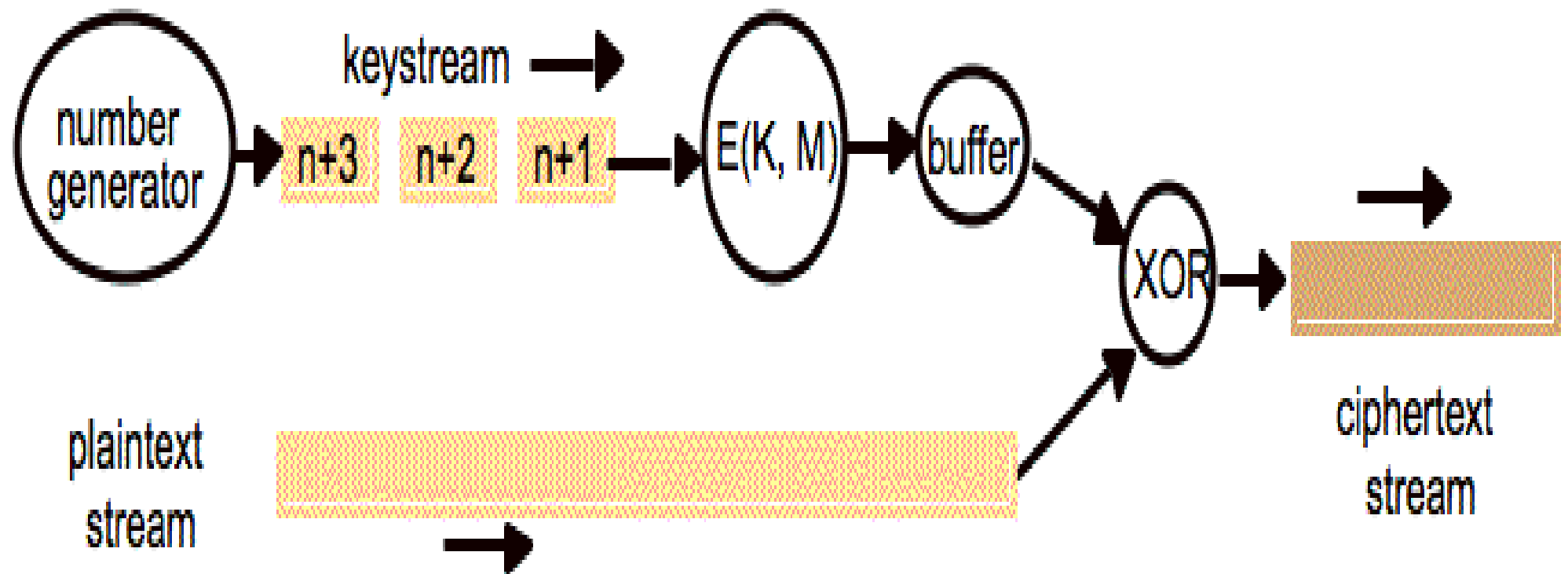
Plain-text output



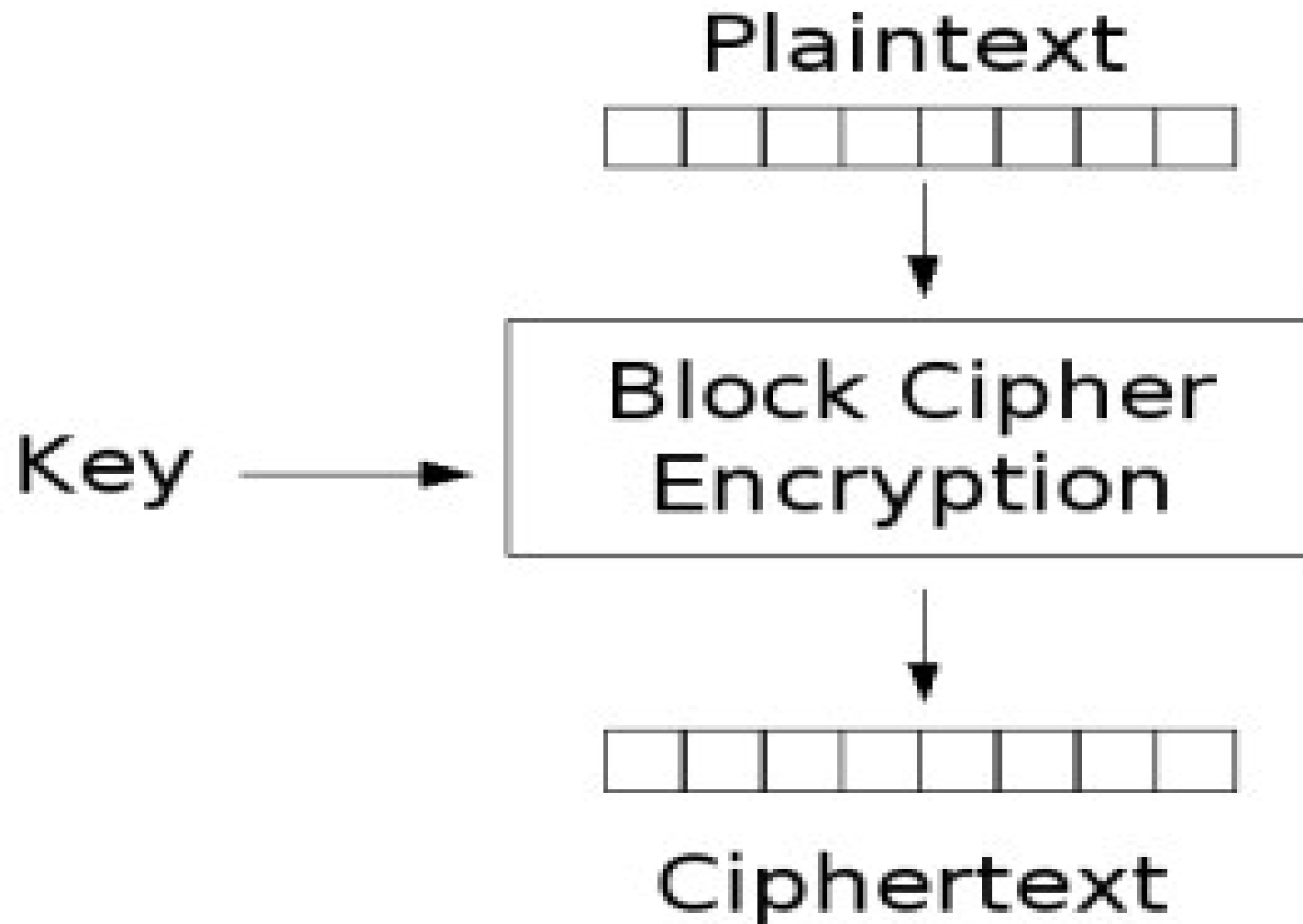
Symmetric Key Cryptography

- Stream Ciphers
 - ◆ Plaintext are combined with a keystream
 - ◆ Keystream are generated by PRG (Pseudo Random Generators)
 - ◆ Each plaintext are encrypted one at a time
- Block Ciphers
 - ◆ Plaintext are divided into n fixed length blocks
 - ◆ Each block will be processed individually
 - ◆ Multiple round with multiple sub keys

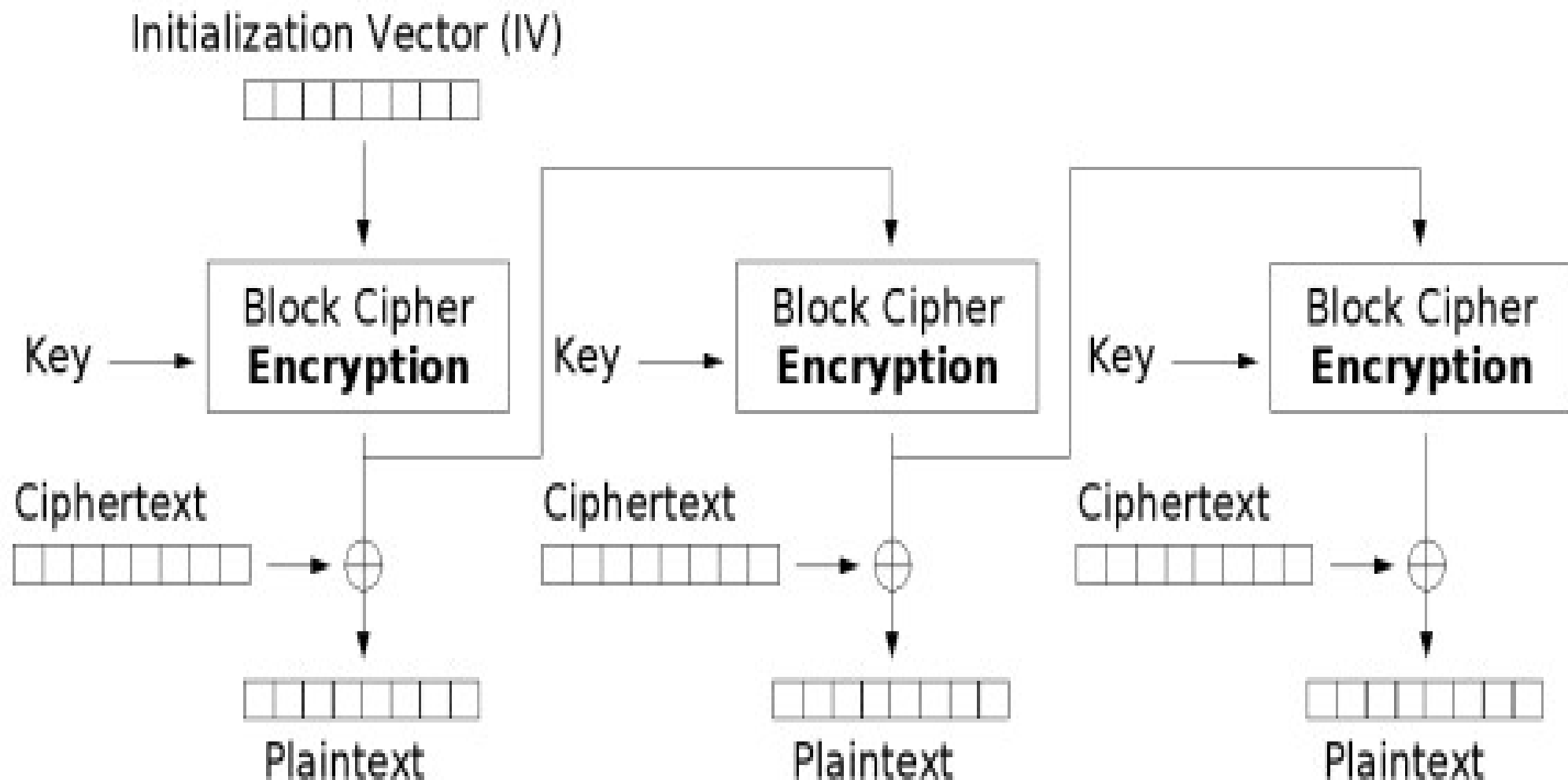
Stream Ciphers



Block Ciphers



Block Ciphers

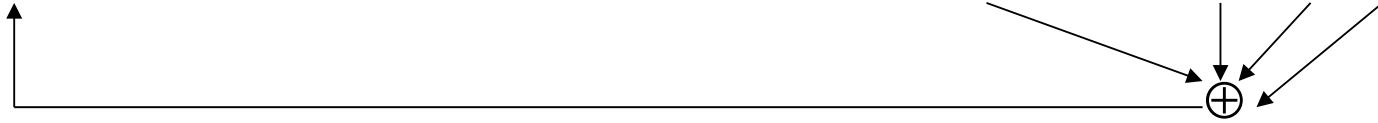


A5/1

- Used in many GSM mobile phone systems
- Designed for hardware
- Using 64 bit keys
- A5/1 uses 3 *shift registers*
 - ◆ X: 19 bits ($x_0, x_1, x_2, \dots, x_{18}$)
 - ◆ Y: 22 bits ($y_0, y_1, y_2, \dots, y_{21}$)
 - ◆ Z: 23 bits ($z_0, z_1, z_2, \dots, z_{22}$)
- Each step of A5/1 produces only a bit

X

x_0 x_1 x_2 x_3 x_4 x_5 x_6 x_7 **x_8** x_9 x_{10} x_{11} x_{12} x_{13} x_{14} x_{15} x_{16} x_{17} x_{18}



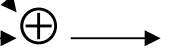
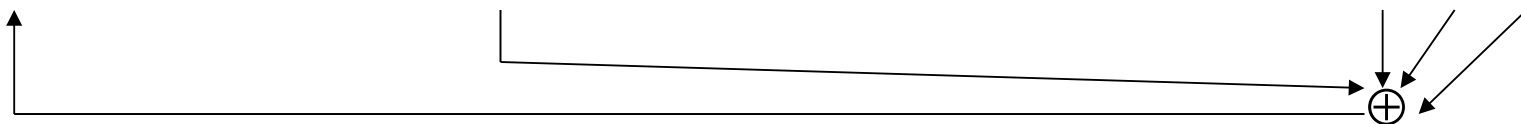
Y

y_0 y_1 y_2 y_3 y_4 y_5 y_6 y_7 y_8 y_9 **y_{10}** y_{11} y_{12} y_{13} y_{14} y_{15} y_{16} y_{17} y_{18} y_{19} y_{20} y_{21}

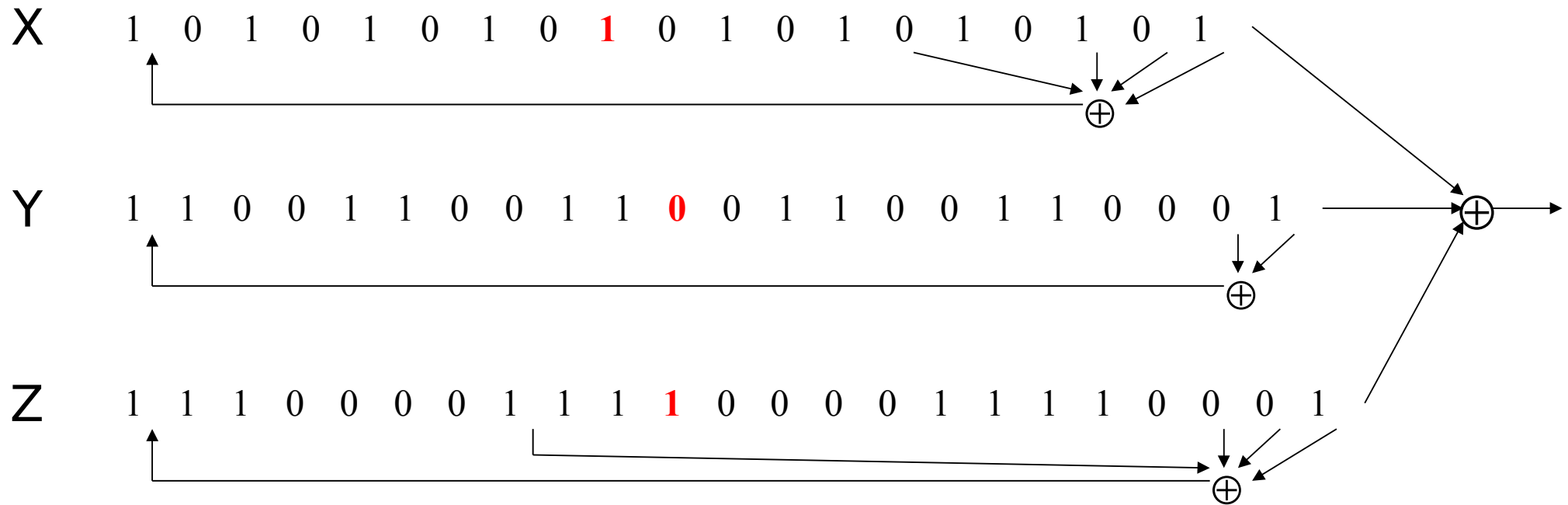


Z

z_0 z_1 z_2 z_3 z_4 z_5 z_6 z_7 z_8 z_9 **z_{10}** z_{11} z_{12} z_{13} z_{14} z_{15} z_{16} z_{17} z_{18} z_{19} z_{20} z_{21} z_{22}



- At each step: $m = \text{maj}(x_8, y_{10}, z_{10})$
 - ◆ Examples: $\text{maj}(0,1,0) = 0$ and $\text{maj}(1,1,0) = 1$
- If $x_8 = m$ then *X steps*
 - ◆ $t = x_{13} \oplus x_{16} \oplus x_{17} \oplus x_{18}$
 - ◆ $x_i = x_{i-1}$ for $i = 18, 17, \dots, 1$ and $x_0 = t$
- If $y_{10} = m$ then *Y steps*
 - ◆ $t = y_{20} \oplus y_{21}$
 - ◆ $y_i = y_{i-1}$ for $i = 21, 20, \dots, 1$ and $y_0 = t$
- If $z_{10} = m$ then *Z steps*
 - ◆ $t = z_7 \oplus z_{20} \oplus z_{21} \oplus z_{22}$
 - ◆ $z_i = z_{i-1}$ for $i = 22, 21, \dots, 1$ and $z_0 = t$
- Keystream **bit** is $x_{18} \oplus y_{21} \oplus z_{22}$



In this example, $m = \text{maj}(x_8, y_{10}, z_{10}) = \text{maj}(\mathbf{1}, \mathbf{0}, \mathbf{1}) = \mathbf{1}$

Register X steps, Y does not step, and Z steps

Keystream bit is XOR of right bits of registers

Here, keystream bit will be $0 \oplus 1 \oplus 0 = 1$

RC4

- A self-modifying lookup table
- Table always contains a permutation of the byte values $0, 1, \dots, 255$
- Initialize the permutation using key
- At each step, RC4 does the following
 - ◆ Swaps elements in current lookup table
 - ◆ Selects a keystream byte from table
- Each step of RC4 produces a **byte**
 - ◆ Efficient in software

RC4

`S[]` is permutation of `0,1,...,255`
`key[]` contains `N` bytes of key

```
for i = 0 to 255
    S[i] = i
    K[i] = key[i (mod N)]
next i
j = 0
for i = 0 to 255
    j = (j + S[i] + K[i]) mod 256
    swap(S[i], S[j])
next i
i = j = 0
```

RC4

- For each keystream byte, swap elements in table and select byte
 - $i = (i + 1) \bmod 256$
 - $j = (j + S[i]) \bmod 256$
 - $\text{swap}(S[i], S[j])$
 - $t = (S[i] + S[j]) \bmod 256$
 - $\text{keystreamByte} = S[t]$
- Use keystream bytes like a one-time pad
- **Note:** first 256 bytes should be discarded
 - ◆ Otherwise, related key attack exists

One Time Pad

- Invented by Vernam (1917)
- $c = E(k, m) = k \oplus m$
- $n = D(k, c) = k \oplus c$
- Plain text 1 0 1 1 0 0 1 0
- Key 0 0 1 0 1 1 0 0
- Cipher text 1 0 0 1 1 1 1 0

$$D(k, E(k, m)) = D(k, k \oplus m) = k \oplus (k \oplus m) = (k \oplus k) \oplus m = m$$

One Time Pad

- Very fast encoding/decoding
- Long keys required to make it secure
- Provides no integrity / authentication
- Proven to have a perfect secrecy

Information Theoretic Security (Shanon, 1949)

Basic idea: CT should reveal no “info” about PT

A cipher (E, D) over (K, C, M) has perfect secrecy IF

$\forall m_0, m_1 \in M \longrightarrow (\text{len}(m_0) = \text{len}(m_1)) \text{ and } \forall c \in C$

$$\Pr[E(k, m_0) = c] == \Pr[E(k, m_1) = c]$$

$(k \xleftarrow{R} K)$
R = random variables

Given CT, attacker can't tell
if the message is m_0 or m_1

Bad News:

$|K| \geq |M|$ key length $\geq$ message length

One Time Pad – Perfect Secrecy

$$\forall m_0, m_1 \in M \longrightarrow (\text{len}(m_0) = \text{len}(m_1)) \text{ and } \forall c \in C$$

$$\Pr[E(k, m_0) = c] == \Pr[E(k, m_1) = c]$$

$$k \oplus m = c \longrightarrow k = m \oplus c$$

$$\#[k \in K: E(k, m) = c] = 1 \quad \forall m, c$$

Given CT, attacker can't tell
if the message is m_0 or m_1
As long as the key is used
ONCE!!!

Perfect secrecy!!

		H		E		L		L		O	message
	7	(H)		4	(E)	11	(L)	11	(L)	14	(O) message
+	23	(X)		12	(M)	2	(C)	10	(K)	11	(L) key
=	30			16		13		21		25	message + key
=	4	(E)		16	(Q)	13	(N)	21	(V)	25	(Z) message + key (mod 26)
		E			Q		N		V		Z → ciphertext

		E		Q		N		V		Z	ciphertext
	4	(E)		16	(Q)	13	(N)	21	(V)	25	(Z) ciphertext
-	23	(X)		12	(M)	2	(C)	10	(K)	11	(L) key
=	-19			4		11		11		14	ciphertext - key
=	7	(H)		4	(E)	11	(L)	11	(L)	14	(O) ciphertext - key (mod 26)
		H			E		L		L		O → message

	4	(E)		16	(Q)	13	(N)	21	(V)	25	(Z) ciphertext
-	19	(T)		16	(Q)	20	(U)	17	(R)	8	(I) possible key
=	-15			0		-7		4		17	ciphertext-key
=	11	(L)		0	(A)	19	(T)	4	(E)	17	(R) ciphertext-key (mod 26)

Two Time Pad is Insecure

- $m1 \oplus k \Rightarrow c1$
- $m2 \oplus k \Rightarrow c2$

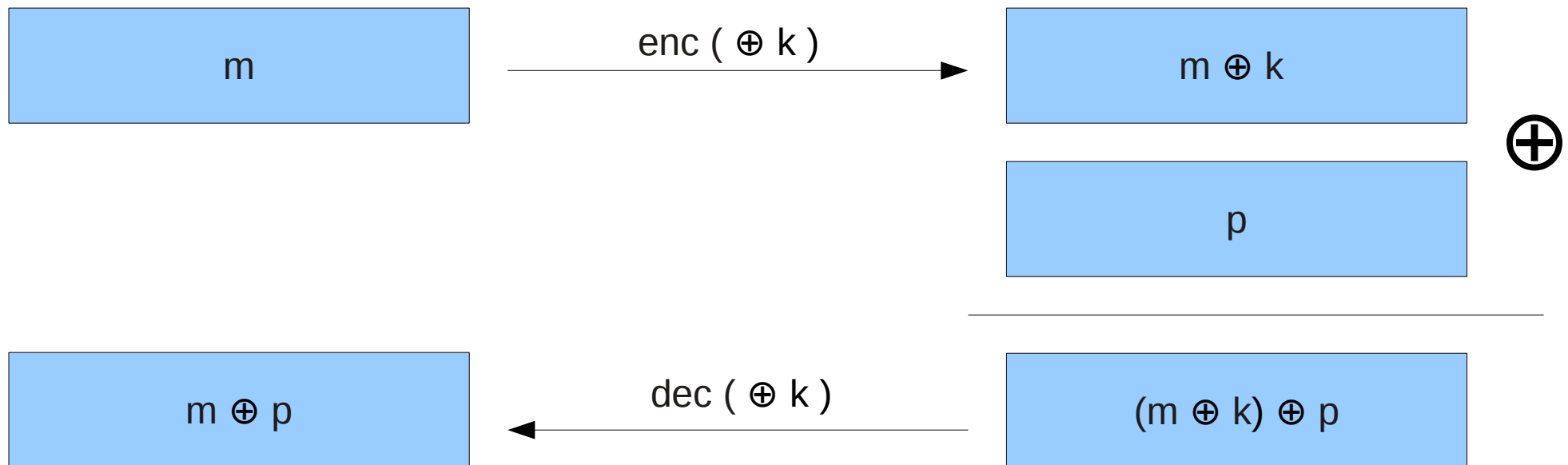
$$c1 \oplus c2 \Rightarrow m1 \oplus m2$$

- ASCII encodings has lots of redundancy

See Demo on 2 Time Pad

<http://web.mit.edu/6.857/OldStuff/Fall03/two-time-pad.html>

OTP – No Integrity



OTP – No Integrity (Malleability)

