

Introduction to Cryptography

Lecture 2: Cryptography Algorithm

CYB 650

Introduction to Cryptography

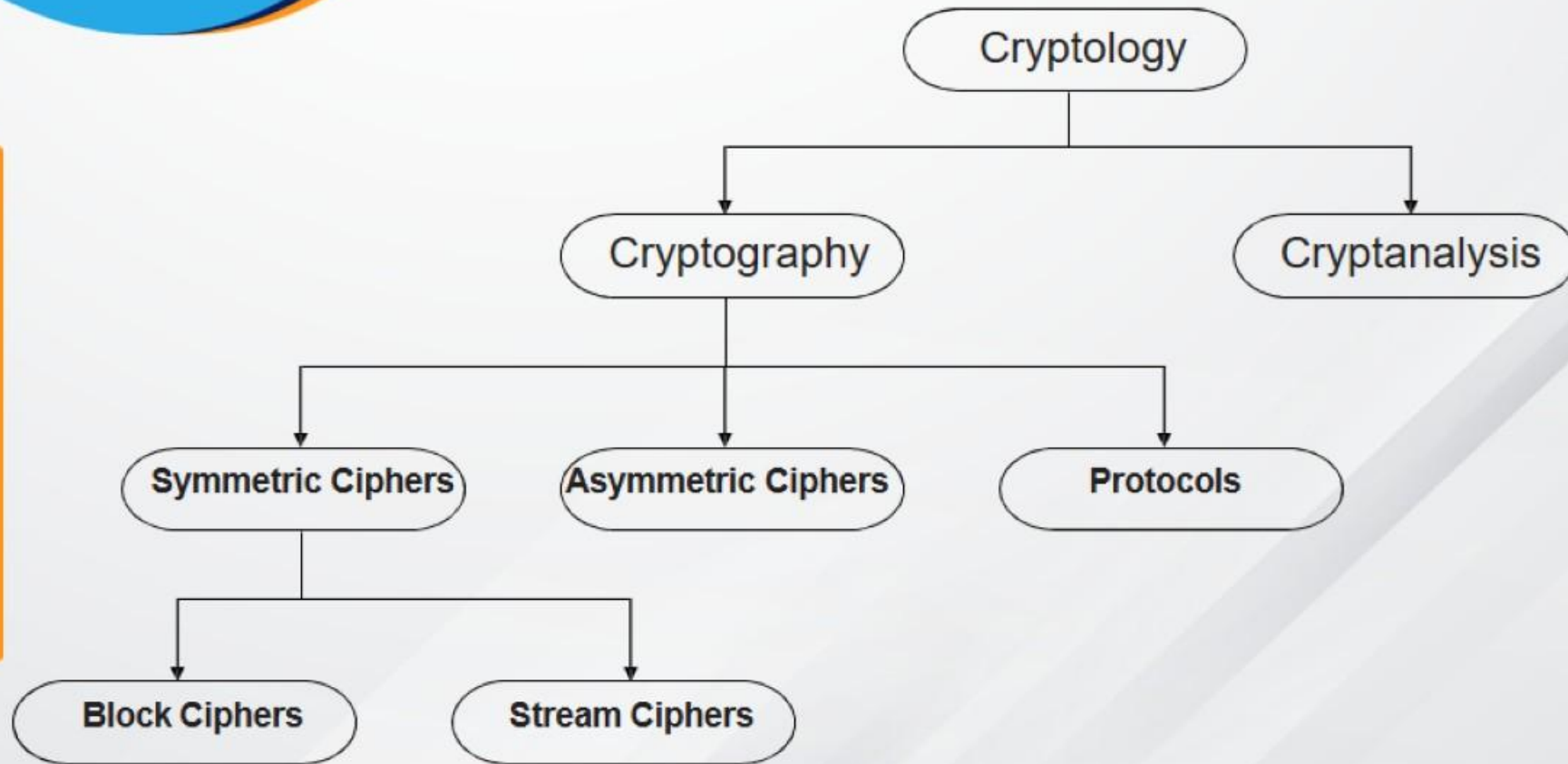
Presented by

Dr. Mohammad Alkhatib

- Overview on Cryptography Algorithms
- Symmetric Ciphers
- Vigenere Cipher
- Transposition Ciphers
- Rail fence Cipher
- Columnar Transposition Cipher
- Stream Ciphers vs. Block Cipher
- Stream Cipher (encryption & decryption)
- One-Time Pad Cipher

Module 01

Classification of the field of Cryptology



Module 02

Vigenere Cipher

Given the following plaintext and key, use the Vigenere cipher to encrypt the message.

key: *deceptivedeceptivedeceptive*

plaintext: *wearediscoveredsaveyourself*

Ciphertext: *zicvtwqnggrzgvtwavzhcqyglmgj*

EX: $K1(d) = 3, P1(w) = 22$

$$C1 = K1 + P1 \bmod 26$$

$$C1 = 3 + 22 \bmod 26 = 25$$

→ $C1 = z$

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

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

Module 02

Vigenere Cipher

❑ Given the following plaintext and key, use the Vigenere cipher to encrypt the message.

❑ key: *cipher*

❑ plaintext: laminthelibrary

❑ Solution:

❑ Ciphertext: kib

key	2	8	15						
plaintext	8	0	12						
ciphertext	10	8	1						

- ❑ Transposition Ciphers are a bit different to Substitution Ciphers.
- ❑ The Substitution ciphers replace each letter with a different letter or symbol to produce the ciphertext,
- ❑ In a Transposition cipher, the letters are just moved around.
- ❑ The letters or words of the plaintext are reordered in some way, and sometimes fixed by a given rule (the key).

❑ One example of a simple transposition cipher, is to reverse the order of the letters in a plaintext, as follows:

❑ plaintext: “a simple example”

❑ Ciphertext: “elpmaxe elpmis a”

❑ Note that the spaces are included

❑ In this example of a simple transposition cipher, the order of the letters in the plaintext are reversed using a key = 4, as follows:

❑plaintext: “a simple example”

❑Ciphertext: “is aelpmaxe elpm”

❑**Note:** We reverse the order of each sequence of 4 letters in the plaintext.

❑**Note:** To decrypt the message, we need to reverse the order of each sequence of 4 letters in the ciphertext.

❑ To encrypt a message using the Rail Fence Cipher, perform the following instructions:

❑ Write your message in zigzag lines across the page, and then read off each row.

❑ you need to have a key, which for this cipher is the number of rows you are going to have.

❑ You then start writing the letters of the plaintext diagonally down to the right until you reach the number of rows specified by the key. You then bounce back up diagonally until you hit the first row again. This continues until the end of the plaintext.

Module 02

Rail fence Cipher

❑ Encrypt the plaintext: "defend the east wall", with a key of 3 using the rail fence cipher

❑ Encryption:

d				n				e				t				l		
	e		e		d		h		e		s		w		l		x	
		f				t				a				a				x

❑ Note that at the end of the message we have inserted two "X"s. These are called nulls, and act as placeholders. They can be used to make the message fit neatly in to the grid

❑ The ciphertext is read off row by row, as follows:

❑ Ciphertext: "dnetleedheswlxftaax"

- ❑ The decryption process for the Rail Fence Cipher involves reconstructing the diagonal grid used to encrypt the message.
- ❑ We start writing the message, but leaving a dash in place of the spaces yet to be occupied.
- ❑ Gradually, you can replace all the dashes with the corresponding letters, and read off the plaintext from the table.
- ❑ You can start by making a grid with as many rows as the key is, and as many columns as the length of the ciphertext.

- ❑ Then place the first letter in the top left square, and dashes diagonally downwards where the letters will be.
- ❑ When you get back to the top row, place the next letter in the ciphertext.
- ❑ Continue like this across the row, and start the next row when you reach the end.

Module 02

Rail fence Cipher

❑ Decrypt the ciphertext: " dnetleedheswlxftaax", with a key of 3 using the rail fence cipher

❑ Decryption:

❑ The first row can be filled as shown in the table

d				n				e				t				l		
	-		-		-		-		-		-		-		-		-	
		-				-				-				-				-

❑ Then, start filling the next row, as following:

d				n				e				t				l		
	e		e		d		h		e		s		w		l		x	
		-				-				-				-				-

Module 02

Rail fence Cipher

❑ The last row can be filled as shown in the table

d				n				e				t				l		
	e		e		d		h		e		s		w		l		x	
		f				t				a				a				x

❑ Finally, you can obtain the plaintext by reading the letters in zigzag lines across the table.

❑ The plaintext: **defend the east wall**

Module 02

Columnar Transposition Cipher

❑ To encrypt a plaintext using the columnar transposition cipher, follow the instructions below:

- i. First, pick a keyword for the encryption process.
- ii. Write the plaintext out in a grid where the number of columns is the number of letters in the keyword.
- iii. Then title each column with the respective letter from the keyword.
- iv. Take the letters in the keyword in alphabetical order, and read down the columns in this order to get the ciphertext.
- v. If a letter is repeated, we do the one that appears first, then the next and so on

Module 02

Columnar Transposition Cipher

❑ Encrypt the message "meet me in the library" using the keyword *tomato*.

t	o	m	a	t	o
5	3	2	1	6	4
m	e	e	t	m	e
i	n	t	h	e	l
i	b	r	a	r	y

❑ Now, read the letters in the columns in alphabetical order, so you can get the ciphertext:

tha etr enb ely mii mer

Module 02

Columnar Transposition Cipher

❑ To decrypt a plaintext using the columnar transposition cipher follow the instructions below:

- i. Start by writing out the keyword and the alphabetical order of the letters of the keyword.
- ii. Divide the length of the ciphertext by the length of the keyword. The answer to this is the number of rows you need to add to the grid.
- iii. Then write the ciphertext down the first column until you reach the last row. The next letter becomes the first letter in the second column (by the alphabetical order of the keyword), and so on.
- iv. Finally, read the letters in the grid row by row to get the plaintext.

Module 02

Columnar Transposition Cipher

❑ Decrypt the message “thaetrenbelymiimer” using the keyword *tomato*.

t	o	m	a	t	o
5	3	2	1	6	4
m	e	e	t	m	e
i	n	t	h	e	l
i	b	r	a	r	y

❑ The Plaintext: “meet me in the library”

Module 02

Columnar Transposition Cipher

❑ Encrypt and then Decrypt the message “I-will-be-in-the-market”
using the keyword *battle*.

Module 02

Columnar Transposition Cipher

❑ Encrypt and then Decrypt the message “I will be in the market” using the keyword *battle*.

b	a	t	t	l	e
2	1	5	6	4	3
i	-	w	i	l	l
-	b	e	-	i	n
-	t	h	e	-	m
a	r	k	e	t	x

❑ The ciphertext: “-btri—alnmxli-twehki-ee”

Module 02

Columnar Transposition Cipher

❑ Decrypt the ciphertext: “-btri—alnmxli-twehki-ee”

b	a	t	t	l	e
2	1	5	6	4	3
i	-	w	i	l	l
-	b	e	-	i	n
-	t	h	e	-	m
a	r	k	e	t	x

❑ The plaintext: I-will-be-in-the-marketx

❑ Then, remove the unnecessary *dashes* and *x*

❑ The final plaintext: I will be in the market

Module 02

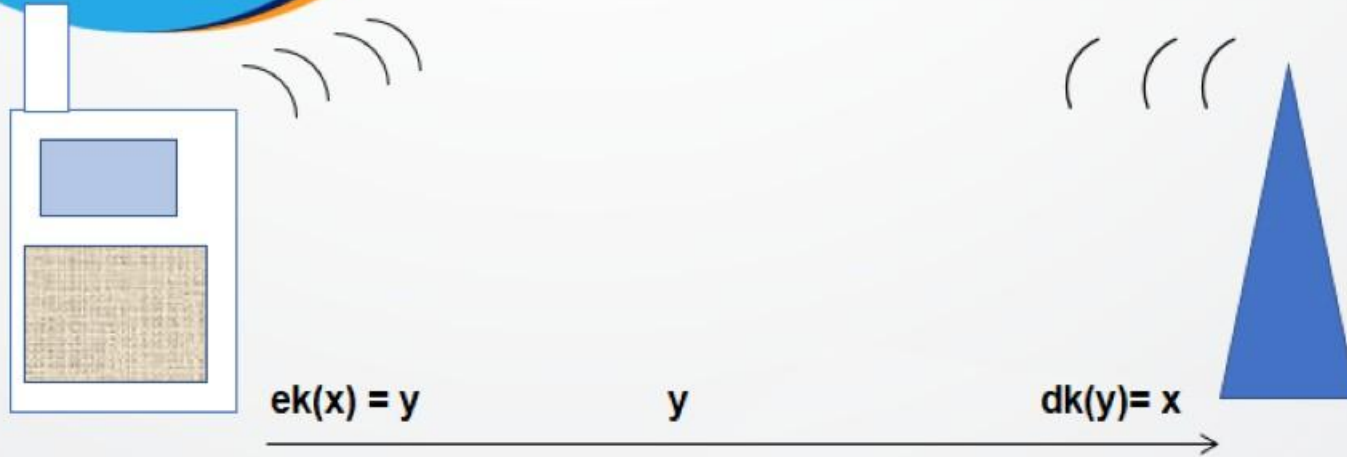
Stream Ciphers vs. Block Ciphers

- Symmetric cryptography is split into block ciphers and stream ciphers.
- **Stream Cipher** encrypts bits (of plaintext) individually
- **Block cipher** encrypts an entire block of bits (of plaintext) at a time
- The following figure illustrates the difference between two types of stream ciphers

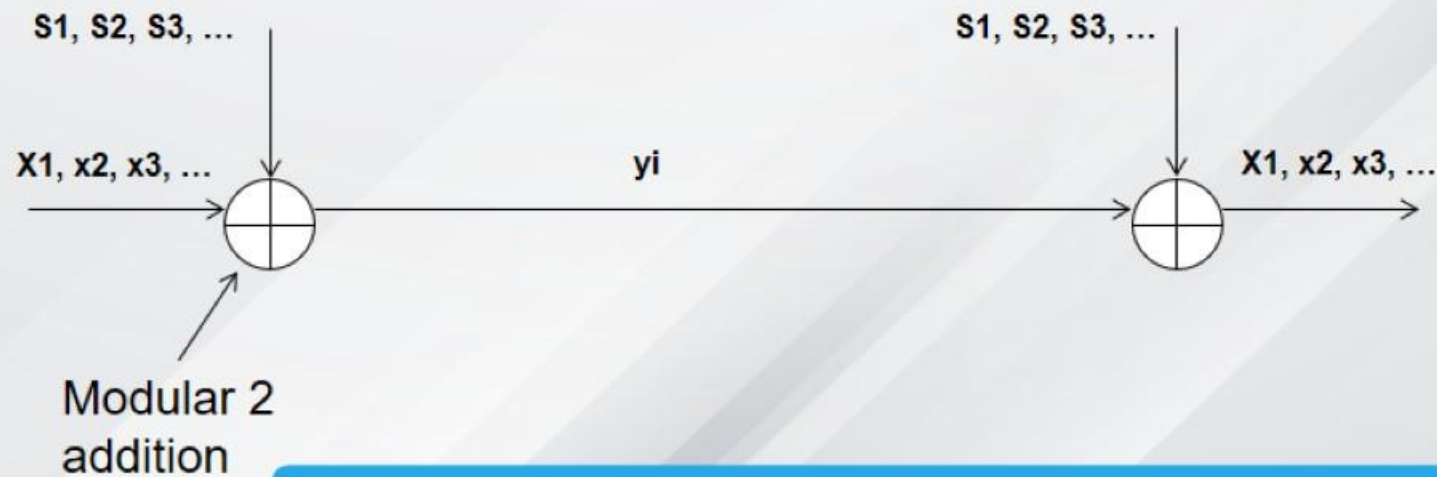


Module 02

Stream Ciphers (motivation)



Encryption and Decryption process is performed using stream ciphers



❑ **Stream ciphers** encrypt bits individually. This is achieved by adding a bit from a key stream to a plaintext bit. There are synchronous stream ciphers where the key stream depends only on the key, and asynchronous ones where the key stream also depends on the ciphertext

❑ **Block ciphers** encrypt an entire block of plaintext bits at a time with the same key. This means that the encryption of any plaintext bit in a given block depends on every other plaintext bit in the same block. Examples: Data Encryption Standard (DES) and Advanced Encryption Standard (AES)

Module 02

Stream Ciphers (Encryption & Decryption)

Definition for Stream Cipher Encryption and Decryption

Remember that the plaintext, the ciphertext and the key stream consist of individual bits, as follows:

$$x_i, y_i, s_i \in \{0, 1\}.$$

Encryption: $y_i = e_{s_i}(x_i) \equiv x_i + s_i \text{ mod } 2$

Decryption: $x_i = d_{s_i}(y_i) \equiv y_i + s_i \text{ mod } 2$

Note that encryption and decryption functions are both simple additions modulo 2

What do you notice from the above functions??

Module 02

Stream Cipher)

The following figure illustrates the encryption and decryption processes with stream ciphers



- Encryption and decryption are the same functions!
- Why can we use a simple modulo 2 addition as encryption?
- What is the nature of the key stream bits s_i ?

We need to prove that the decryption function actually produces the plaintext bit x_i again. The ciphertext bit y_i was computed using the encryption function $y_i \equiv x_i + s_i \text{ mod } 2$. We insert this encryption expression in the decryption function as follows:

$$d_{s_i}(y_i) \equiv y_i + s_i \text{ mod } 2$$

$$\equiv (x_i + s_i) + s_i \text{ mod } 2$$

$$\equiv x_i + s_i + s_i \text{ mod } 2$$

$$\equiv x_i + 2s_i \text{ mod } 2$$

$$\equiv x_i + 0 \text{ mod } 2$$

$$\equiv x_i \text{ mod } 2$$

x_i	s_i	y_i	X_i
0	0	0	0
0	1	1	0
1	0	1	1
1	1	0	1

➤ If the key bit s_i behaves perfectly randomly, i.e., it is unpredictable and has exactly a 50% chance to have the value 0 or 1, then both possible ciphertexts also occur with a 50% likelihood.

➤ Note that the XOR function is perfectly balanced, which distinguishes it from other Boolean functions

Module 02

Stream Cipher

Example 1: Alice wants to encrypt the letter A, where the letter is given in ASCII code. The ASCII value for A is $65_{10} = 1000001_2$. Assume that the first key stream bits are $(s_0, \dots, s_6) = 0101100$.

Alice

Oscar

Bob

$$x_0, \dots, x_6 = 1000001 = A$$

$$\oplus$$

$$s_0, \dots, s_6 = 0101100$$

$$y_0, \dots, y_6 = 1101101 = m$$

$$m = 1101101$$

$$y_0, \dots, y_6 = 1101101$$

$$\oplus$$

$$s_0, \dots, s_6 = 0101100$$

$$x_0, \dots, x_6 = 1000001 = A$$

**Encryption & Decryption Scenario for
Stream Ciphers**

- ❑ The *key stream*, is the central issue for the security of stream ciphers.
- ❑ The security of a stream cipher *completely depends on the key stream*.
- ❑ key stream bits should appear like a random sequence to an attacker
- ❑ In other words, the key stream bits should be unpredictable

- ❑ The perfect cipher can be defined as unconditionally secure cryptosystem
- ❑ A cryptosystem is unconditionally secure if it cannot be broken even with infinite computational resources.
- ❑ Definition of One-Time Pad cipher:

A stream cipher that has the following properties:

1. The key stream $s_0, s_1, s_2, \dots$ is generated by a true random number generator, and
2. The key stream is only known to the legitimate communicating parties, and
3. Every key stream bit s_i is only used once

Is called one-time pad, and it is unconditionally secure.

□ For every ciphertext bit we get an equation of this form:

$$y_0 \equiv x_0 + s_0 \pmod{2}$$

$$y_1 \equiv x_1 + s_1 \pmod{2}$$

⋮

□ Note that each individual relation is a linear equation modulo 2 with two unknowns.

□ Note that if the values for $s_0, s_1, \dots$ are *not truly random*, there will be some functional relationship between them, and the equations shown above are not independent anymore.

- ❑ Key stream bits cannot be re-used
- ❑ This implies that we need one key bit for every bit of plaintext
- ❑ The key is as long as the plaintext!
- ❑ Consider the following scenario:

Ex: Alice and Bob wants to share a CD with 1 MByte of true random numbers. If they send a single email with an attachment of 1 MByte, they could encrypt and decrypt it only once, but after that they would need to exchange a true random key stream again

There are three main types of random number generators, as follows:

1. **True Random Number Generators (TRNG)**
2. **Pseudorandom Number Generators (PRNG)**
3. **Cryptographically Secure Pseudorandom Number Generators (CSPRNG)**



Thank You