

**HW1: Exercise 1**

a) Which cryptographic goal is about ensuring that information is only accessible to authorized parties?

Confidentiality

b) Which cryptographic goal is to ensure that a sender cannot deny having sent a message? Non-Repudiation

c) Which cryptographic goal involves verifying the identity of a user or device? Authentication

d) Which cryptographic goal involves ensuring that information has not been modified or tampered with during transmission or storage? Integrity

**HW1: Exercise 2**

a) What is the type of cryptanalysis that involves trying to decrypt a message when the attacker has only the encrypted message and no other information about the encryption process? Ciphertext-only cryptanalysis.

b) What is the type of cryptanalysis that involves trying to decrypt a message when the attacker has both the encrypted message and some unencrypted messages that were encrypted using the same encryption key? Known-plaintext cryptanalysis.

**HW1: Exercise 3**

Discuss how cryptography may be applied in the following activities. Specify which cryptographic goals one may want to reach for in each case (and why).

a) Secure communication: Cryptography is often used to secure communication channels, such as when sending emails or messages over the internet. This helps to protect the confidentiality of the communication and ensure that it is not intercepted or modified by unauthorized parties.

→ Confidentiality: you don't want anyone you are no communicating with to find out about your messages.

→ Authentication: you want to be sure that you are sending the messages to the right people.

→ Integrity: you want your meaning to be properly conveyed which is hard enough without your message being changed.

b) E-commerce: Cryptography is used to secure online transaction, such as when you make a purchase online with your credit card. This helps to protect the confidentiality of your financial information and prevent fraud.

→ Confidentiality: you don't want anyone not involved knowing what you are shopping for.

→ Non-repudiation: so that you can't deny having made a purchase.

→ Authentication: so that your shopping history/cart is saved when logging in.

→ Integrity: you don't want your target of purchase to be changed midway through.

c) **Password protection:** Cryptography is used to secure passwords, which helps to protect the confidentiality of your personal accounts and prevent unauthorized access;

→ *Confidentiality: you don't want anyone else to know your password.*

→ *Authentication: hard to login when the password input gets changed before the server accepts it.*

d) **File storage:** Cryptography is used to secure data stores on devices, such as laptops and smartphone. This helps to protect the confidentiality to the data and prevent unauthorized access.

→ *Confidentiality: you don't want anyone not allowed seeing/spying on your stored files.*

→ *Non-repudiation: if your file storage is for your group you want to know who has made changes to witch file.*

→ *Authentication: to make sure you are allowed to make changes to stored file or even see then in the first place.*

→ *Integrity: file storage would be useless if the files could change automatically in storage without any input from you.*

e) **Network security:** Cryptography is used to secure networks, such as corporate networks and public Wi-fi hotspots. This helps to protect the confidentiality and integrity of the data transmitted over the network and prevent unauthorized access.

→ *Confidentiality: you don't want others to know about your network activity.*

→ *Non-repudiation: goes back to e-commerce for example.*

→ *Authentication: most webpages have logins to know that it's truly you to ease your experience on the internet, goes back to ecommerce.*

### HW1: Example

Let the key  $k = (a, b) = (15, 19)$ .

We note that  $\gcd(15, 26) = 1$ .

Hence  $a$  has an inverse  $a^{-1}$ , namely  $a^{-1} = 7$ .

[To check this, we find that  $15 \cdot 7 = 105 = 4 \cdot 26 + 1 \equiv 1 \pmod{26}$ .]

Let us encrypt message  $ATTACK = x_1 x_2 \dots x_6 = 0, 19, 19, 0, 2, 10$  by computing

$$y_1 = 15 \cdot 0 + 19 \pmod{26} = 19$$

$$y_2 = 15 \cdot 19 + 19 \pmod{26} = 18$$

⋮

resulting in ciphertext  $y = y_1 y_2 \dots y_6 = 19, 18, 18, 19, 23, 13 = TSSTXN$ .

To decrypt the ciphertext, we compute

$$x_1 = 7(19 - 19) \pmod{26} = 0$$

$$x_2 = 7(18 - 19) \pmod{26} = 19$$

⋮

resulting back in  $x_1 x_2 \dots x_6 = 0, 19, 19, 0, 2, 10 = ATTACK$

**HW1: Exercise 4**

a) Knowing that the ciphertext **Rpalz huk khyaz** was encrypted using Caesar cipher, find the plaintext and the key  $k$  that was used for encryption.

Using: <https://www.dcode.fr/caesar-cipher>

Plaintext = " Kites and Darts " with  $k = 7$

b) Use the Caesar cipher with key  $k = 14$  to encrypt and the plaintext SECRET. Show the steps of computation.

Plaintext = SECRET                       $k = 14$

$$y_1 (S \rightarrow 18) = 18 + 14 \pmod{26} = 32 = 26 + 6 \rightarrow 6 \rightarrow G$$

$$y_2 (E \rightarrow 4) = 4 + 14 \pmod{26} = 18 \rightarrow S$$

$$y_3 (C \rightarrow 2) = 2 + 14 \pmod{26} = 16 \rightarrow Q$$

$$y_4 (R \rightarrow 17) = 17 + 14 \pmod{26} = 31 = 26 + 5 \rightarrow 5 \rightarrow F$$

$$y_5 (E \rightarrow 4) = 4 + 14 \pmod{26} = 18 \rightarrow S$$

$$y_6 (T \rightarrow 19) = 19 + 14 \pmod{26} = 33 = 26 + 7 \rightarrow 7 \rightarrow H$$

Ciphertext = GSQFSH

**HW1: Exercise 5**

a) Which one of the following is a valid key for Affine cipher and why?

To an Affine cipher key be valid, key  $k = (a, b) \in \mathbb{Z}^2$ , where  $\gcd(a, m) = 1$

( <https://www.wolframalpha.com/> )

a.  $k = (8, 17)$

$$\gcd(8) = 2^3$$

$$\gcd(17) = 2 \cdot 13$$

$$\gcd(8, 26) = 2 \rightarrow \text{so this key is not valid}$$

b.  $k = (21, 8)$

$$\gcd(21, 26) = 1 \rightarrow \text{so this key is valid}$$

c.  $k = (13, 14)$

$$\gcd(13, 26) = 13 \rightarrow \text{so this key is not valid}$$

b) Use the valid key from above to encrypt the plaintext PRIVATE. Show the steps of computation.

*Encryption :  $ax + b \pmod{m}$*

$$k = (21, 8)$$

$$y_1 (P \rightarrow 15) = 21 \cdot 15 + 8 \pmod{26} = 323 = 26 \cdot 12 + 11 \rightarrow 11 \rightarrow L$$

$$y_2 (R \rightarrow 17) = 21 \cdot 17 + 8 \pmod{26} = 365 = 26 \cdot 14 + 1 \rightarrow 1 \rightarrow B$$

$$y_3 (I \rightarrow 8) = 21 \cdot 8 + 8 \pmod{26} = 176 = 26 \cdot 6 + 20 \rightarrow 20 \rightarrow U$$

$$y_4 (V \rightarrow 21) = 21 \cdot 21 + 8 \pmod{26} = 449 = 26 \cdot 17 + 7 \rightarrow 7 \rightarrow H$$

$$y_5 (A \rightarrow 0) = 21 \cdot 0 + 8 \pmod{26} = 8 \rightarrow I$$

$$y_6 (T \rightarrow 19) = 21 \cdot 19 + 8 \pmod{26} = 407 = 26 \cdot 15 + 17 \rightarrow 17 \rightarrow R$$

$$y_7 (E \rightarrow 4) = 21 \cdot 4 + 8 \pmod{26} = 92 = 26 \cdot 3 + 14 \rightarrow 14 \rightarrow O$$

Ciphertext = LBUHIRO

### HW1: Exercise 6

Performing a brute force attack, on average we have to try 50 % of all keys before finding the correct one. Suppose you are trying to break a system protected by a 5-digit PIN code. Each digit of the code is a number between 0 and 9. It takes just 2 seconds to generate and test one code, but after every three failed login attempts the system will be locked for 10 minutes. How long will it take on average to break the system by brute force?

- To get the code needs try at least 50% of all keys
- 5-digit code [0-9]
- 2 seconds to generate 1 code
- After 3 fail attempts locks for 10 minutes

$$\text{Key space } K = 10^5 = 100000 \rightarrow 50\% \text{ is } 50000$$

$$3 \text{ attempts are } \rightarrow 2\text{seconds} + 2\text{seconds} + 2\text{seconds} + 10\text{minutes} = 2+2+2+600=606\text{seconds}$$

$$5000/3 = 16666 \text{ rounds of 3 attempts}$$

$$5000 \text{ attempts are } 16666 \cdot 3 \text{ rounds of attempts} + 2 \text{ attempts}$$

$$16666 \text{ rounds} \cdot 606 \text{ seconds} = 10099596 \text{ seconds}$$

$$2 \text{ attempts} = 4 \text{ seconds}$$

$$\text{Total attempts} = 10099596 + 4 = 10099600 \text{ seconds} = 2805.44 \text{ hours} \approx 117 \text{ days}$$

### HW1: Exercise 7

Oscar is trying to break ciphertext  $y$  on brute force. It takes him 1 second to generate a key  $k'$  and check if  $dk'(y) = x$ . How long will it take him (on average) to decipher  $y$ , when the size of key space  $K$  is

a.  $2^{10}$

$$2^{10} = 1024 \text{ combinations}$$

$$50\% \text{ of the } 1024 \text{ combinations} = 512 \text{ combinations}$$

$$\frac{2^{10}}{2} \cdot 1 \text{ second} = 512 \text{ seconds} = 8.5333 \dots \text{ minutes}$$

b.  $2^{30}$  $2^{30} = 1073741824$  combinations

50% of the 1073741824 combinations = 536870912 combinations

$$\frac{2^{30}}{2} \cdot 1 \text{ second} = 536870912 \text{ seconds} = 8947848.533 \dots \text{minutes}$$

**HW1: Exercise 8**

Run a frequency analysis on the text of this Wikipedia article on cryptography:

<https://en.wikipedia.org/wiki/Cryptography> .

Analyze the frequencies of letters A-Z for single characters, bigrams and trigrams.

Show your results. What can you say about the results compared to the respective general frequencies in English?

Analyzed with the website : <https://www.dcode.fr/frequency-analysis>

↑↓	↑↓	↑↓		↑↓	↑↓	↑↓
E	4964x	12%	Analyzed with the website : <a href="https://www.dcode.fr/frequency-analysis">https://www.dcode.fr/frequency-analysis</a>  Comparing with the general Frequencies of single letters in English ,we can see that the tables are very similar.	TH	428x	2.07%
T	3702x	8.95%		HE	387x	1.87%
A	3284x	7.94%		IN	354x	1.71%
I	3124x	7.55%		ER	345x	1.67%
R	2823x	6.83%		RE	310x	1.5%
S	2814x	6.8%		TI	293x	1.42%
N	2771x	6.7%		EN	277x	1.34%
O	2674x	6.47%		ST	267x	1.29%
C	2021x	4.89%		AN	260x	1.26%
H	1768x	4.27%		ES	257x	1.24%
L	1625x	3.93%		( ... )		
P	1364x	3.3%		#N : 479	Σ = 20680	Σ = 99.580
D	1355x	3.28%				
Y	1197x	2.89%		↑↓	↑↓	↑↓
M	1158x	2.8%		THE	180x	1.31%
U	1071x	2.59%		YPT	108x	0.78%
G	819x	1.98%		TIO	100x	0.73%
F	786x	1.9%		RYP	97x	0.7%
B	587x	1.42%		CRY	95x	0.69%
W	430x	1.04%		ION	86x	0.62%
V	425x	1.03%		HER	70x	0.51%
K	335x	0.81%		AND	62x	0.45%
X	127x	0.31%		ING	61x	0.44%
Q	83x	0.2%		ENT	58x	0.42%
Z	35x	0.08%		( ... )		
J	19x	0.05%		#N : 2713	Σ = 13787	Σ = 100.89
#N : 26	Σ = 41361	Σ = 100.01				

**HW2: Exercise 1**

a) What is the main difference between a stream cipher and a block cipher?

A stream cipher encrypts plaintext one bit at a time, while block ciphers encrypt a fixed number of bits ( called a block size ) at a time.

b) What are some common uses of stream ciphers?

Because of their simplicity and efficiency, stream ciphers are primarily used for real-time, low-latency encryption of data that is being transmitted over a communication channel. These are often wireless and mobile communications like cellular networks , satellite communications, wireless LAN's and various industrial controls systems, SCADA systems and IoT devices.

c) How does a stream cipher generate the keystream used for encryption?

A stream cipher generates a key stream bases on a fixed key and an internal state. The key is used to initialize the cipher, and the keystream is generated by repeatedly applying a pseudorandom function to the internal state.

In the case of a synchronous stream cipher , the keystream is generated based on a fixed k and an internal state. It is independent from the plaintext and ciphertext, and the key is generated by repeatedly applying a pseudorandom function to the internal state.

In the case of an asynchronous stream sypher, the keystream is generated on the plaintext and ciphertext the key is used to initialize the cipher , and the keystream is then generated by repeatedly applying a pseudorandom function to the plain text and ciphertext.

d) What is the main benefit of using a stream cipher for encryption?

The main benefit is that it can encrypt data in real-time, as the keystream can be generated and used to encrypt the data as it is being transmitted or stored.

e) How are the encryption and decryption operations performed in a stream cipher?

Both encryption and decryption are based on the XOR operation.

In the encryption, the plaintext is XORed with a keystream generated by the stream cipher:

$$y_i = e_{si}(x_i) = x_i + s_i \pmod{2}$$

In the decryption, the cyphertext is XORed with the keystream to obtain the original plaintext:

$$x_i = d_{si}(y_i) = y_i + s_i \pmod{2}$$

**HW2: Exercise 2****True or false?**

a) A truly random number is generated by a physical process, such as radioactive decay. **True**

*"True random number generators (TRNGs) use physical processes, such as radioactive decay or thermal noise, to generate truly random numbers. TRNGs are nondeterministic, and relatively slow and expensive to use"*

b) Pseudorandom number generators are typically slower and less efficient than true random number generators. **False**

*"Pseudorandom number generators are deterministic, and typically faster and more efficient than TRNGs."*

c) Cryptographically secure pseudorandom number generators are designed to be resistant to attacks that attempt to predict the next number in the sequence. **True**

*"CSPRNGs are designed to be difficult to predict, even if an attacker knows the algorithm and the seed."*

d) Ordinary pseudorandom number generators can be used in security-related applications. **False**

*"Ordinary PRNGs are typically used in non-security related applications, such as generating random numbers for simulations or games."*

e) The one-time pad encryption method is considered unbreakable as long as the key is truly random and kept secret. **True**

*"The one-time pad is a method of encryption that is considered to be completely unbreakable, as long as it is used correctly. (...) It is unconditionally secure in that if the key is kept secret, it cannot be mathematically broken even with unlimited resources."*

## HW2: Exercise 3

An adversary has access to a plaintext-ciphertext pair (a known-plaintext attack). They know that the method of encryption is stream cipher. The texts are as follows:

Plaintext:      0100    1000    0100    0101    0100    1100    0101    0000

Ciphertext:    0001    0011    1111    1000    1011    1011    1011    1111

Show how to extract the key stream used for encryption of the plaintext. Why it is vital that the key stream appears random and does not give away any previous or subsequent key stream bits?

Due to the XOR's properties we can use also XOR to get the key stream used:

Plaintext:	0100 1000 0100 0101 0100 1100 0101 0000	<del><math>x_i</math></del>
Ciphertext:	0001 0011 1111 1000 1011 1011 1011 1111	<del><math>+ x_i + s_i</math></del>
Key:	0101 1011 1011 1101 1111 0111 1110 1111	$s_i$

It's important that the key stream appears random and don't give away any previous or subsequent key stream bits because this form of encryption/description only involve simple bitwise operations.

## HW2: Exercise 4

The stream cipher can be generalized to work in alphabets other than the binary. Consider a scheme which operates on the English alphabet represented by the numbers 0,1,...,25. Note that in this case, the encryption and decryption functions are not identical.

a) What are the encryption and decryption functions here?

Encryption:     $y_i = e_{s_i}(x_i) = x_i + s_i \pmod{26}$

Decryption:     $x_i = d_{s_i}(y_i) = y_i - s_i \pmod{26}$

b) Decrypt ciphertext rviodthvays which was encrypted using the key rsidpydkawo.

Ciphertext	r	v	i	o	d	t	h	v	a	y	s
Numerical Ciphertext	17	21	8	14	3	19	7	21	0	24	18
Key	r	s	i	d	p	y	d	k	a	w	o
Numerical key	17	18	8	3	15	24	3	10	0	22	14
Numerical Plaintext:	0	3	0	11	14	21	4	11	0	2	4
Plaintext:	a	d	a	l	o	v	w	l	a	c	e

### Example : LFSR – Linear Feedback Shift Register

**Example.** We find the sequence produced by the LFSR of the previous example, given initial values  $s_0 = 0, s_1 = 1, s_2 = 0$ .

$\leftarrow s_0 + s_1 \pmod{2}$  from previous cycle

Clock cycle $i$	$FF_2$	$FF_1$	$FF_0 = s_i$
0	0	1	0
1	1	0	1
2	1	1	0
3	1	1	1
4	0	1	1
5	0	0	1
6	1	0	0
7	0	1	0

period (maximum length)  $\rightarrow$  (bracketed next to cycles 0-7)

Note, original state (bracketed next to cycle 0)

**Example.** Let us find the sequence produced by the LFSR with  $m = 4$  and feedback coefficients  $p_0 = p_1 = p_2 = p_3 = 1$ , given initial values

- a)  $s_0 = s_1 = s_2 = s_3 = 1$   
 b)  $s_0 = s_1 = 1, s_2 = s_3 = 0$

$\leftarrow$  max. period would be  $2^4 - 1 = 15$

Clock cycle	$FF_3$	$FF_2$	$FF_1$	$FF_0$
0	1	1	1	1
1	0	1	1	1
2	1	0	1	1
3	1	1	0	1
4	1	1	1	0
5	1	1	1	1

period 5 (bracketed next to cycles 0-5)

Clock cycle	$FF_3$	$FF_2$	$FF_1$	$FF_0$
0	0	0	1	1
1	0	0	0	1
2	1	0	0	0
3	1	1	0	0
4	0	1	1	0
5	0	0	1	1

period 5 (bracketed next to cycles 0-5)



**HW2: Exercise 5**

Consider the LFSR with degree  $m = 3$  and feedback coefficients  $p_0 = 1$ ,  $p_1 = 1$  and  $p_2 = 0$ .

Find the sequence generated from the initial values  $s_0 = 0$ ,  $s_1 = 1$ ,  $s_2 = 1$ .

Clock Cycle	FF <sub>2</sub>	FF <sub>1</sub>	FF <sub>0</sub>	degree $\rightarrow m = 3$ feedback coefficients $\rightarrow p_0 = 1, p_1 = 1, p_2 = 0$ initial values $\rightarrow s_0 = 0, s_1 = 1, s_2 = 1$ max period $\rightarrow 2^3 - 1 = 7$  <b>Resolution:</b> <b>FF<sub>1</sub> XOR FF<sub>0</sub> = FF<sub>2</sub> and then shift</b>
0	1	1	0	
1	1	1	1	
2	0	1	1	
3	0	0	1	
4	1	0	0	
5	0	1	0	
6	1	0	1	
7	1	1	0	

**HW2: Exercise 6**

Consider the LFSR with degree  $m = 4$ , feedback coefficients  $p_0 = 0$ ,  $p_1 = 1$ ,  $p_2 = 0$ ,  $p_3 = 1$  and initial values  $s_0 = 0$ ,  $s_1 = 1$ ,  $s_2 = 1$ ,  $s_3 = 1$ .

Find the period of this LFSR. Is it a maximum-length LFSR?

Clock Cycle	FF <sub>3</sub>	FF <sub>2</sub>	FF <sub>1</sub>	FF <sub>0</sub>	degree $\rightarrow m = 4$ feedback coefficients $\rightarrow p_0 = 0, p_1 = 1, p_2 = 0, p_3 = 1$ initial values $\rightarrow s_0 = 0, s_1 = 1, s_2 = 1, s_3 = 1$ max period $\rightarrow 2^4 - 1 = 15$  <b>Resolution:</b> <b>FF<sub>1</sub> XOR FF<sub>3</sub> = FF<sub>2</sub> and then shift</b>
0	1	1	1	0	
1	0	1	1	1	
2	1	0	1	1	
3	0	1	0	1	
4	0	0	1	0	
5	1	0	0	1	
6	1	1	0	0	
7	1	1	1	0	
8	0	1	1	1	
9	1	0	1	1	

**HW3: Exercise 1.**

a) What is the main weakness of DES and why it is no longer considered a secure encryption algorithm?

By today's standards, DES is no longer considered secure due to the small key size (56-bit to 64-bit ) which makes it vulnerable to brute-force attacks.

b) What is the goal of confusion in symmetric-key cryptography and how does the round function of DES achieve it?

The goal of the **confusion** is making the relationship between the encryption key and the ciphertext as complex and non-linear as possible. The DES confusion is achieved, by several rounds of the round function. The round function incorporates multiple operations: expansion permutation, key mixing and substitution box. In each round, the S-boxes that result from the substitution box are non – linear which means that the output of the S-box is not a simple function of the input. - This is what makes it difficult for the attacker to predict the output of the S-box based on the input.

c) How do the initial permutation and final permutation of DES help to achieve diffusion?

Diffusion is the process of spreading the plaintext information uniformly over the entire ciphertext. In the case of DES, the initial and final permutation operations contribute to diffusion, as well as the repeated application of the substitution and permutation operation in each round. Permutations is essentially shifting bits in the blocks therefore, anytime there is a permutation operation, **diffusion** happens.

d) How does diffusion differ from the avalanche effect?

**Diffusion is the process** of spreading the plaintext information uniformly over the entire ciphertext, so a change in one plaintext bit is likely to affect many ciphertext bits. This process is the source of the **avalanche effect**. The avalanche effect is the property of cryptographic systems that states that small changes in the plaintext or key result in large and unpredictable changes.

**HW3: Exercise 2.**

Find the values  $S_1(x_1)$  and  $S_1(x_2)$  where  $S_1$  is the first S-box of DES and inputs are as follows:

a)  $x_1 = 000000$ ,  $x_2 = 000001$

b)  $x_1 = 111111$ ,  $x_2 = 100000$

c)  $x_1 = 101010$ ,  $x_2 = 010101$

Present the values  $S_1(x_1)$  and  $S_1(x_2)$  in binary.

S-box  $S_1$

$S_1$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	04	13	01	02	15	11	08	03	10	06	12	05	09	00	07
1	00	15	07	04	14	02	13	01	10	06	12	11	09	05	03	08
2	04	01	14	08	13	06	02	11	15	12	09	07	03	10	05	00
3	15	12	08	02	04	09	01	07	05	11	03	14	10	00	06	13

**a)  $x_1 = 000000$ ,  $x_2 = 000001$**

For  $x_1 = 000000$ , the row number is 0 and the column number is 0.

**So,  $S_1(x_1) = 14$  in binary is 1110.**

For  $x_2 = 000001$ , the row number is 0 and the column number is 1.

**So,  $S_1(x_2) = 4$  in binary is 0100.**

**b)  $x_1 = 111111$ ,  $x_2 = 100000$**

For  $x_1 = 111111$ , the row number is 3 and the column number is 63 (in binary 111111 is 63 in decimal).

**So,  $S_1(x_1) = 13$  in binary is 1101.**

For  $x_2 = 100000$ , the row number is 1 and the column number is 32 (in binary 100000 is 32 in decimal).

**So,  $S_1(x_2) = 14$  in binary is 1110.**

**c)  $x_1 = 101010$ ,  $x_2 = 010101$**

For  $x_1 = 101010$ , the row number is 2 and the column number is 42 (in binary 101010 is 42 in decimal).

**So,  $S_1(x_1) = 6$  in binary is 0110.**

For  $x_2 = 010101$ , the row number is 1 and the column number is 21 (in binary 010101 is 21 in decimal). So,

**$S_1(x_2) = 10$  in binary is 1010.**

**HW3: Exercise 3.**

DES has the following *weak keys* and *semi-weak key pairs*:

weak key (hexadecimal)	$C_0$	$D_0$
0101 0101 0101 0101	$\{0\}^{28}$	$\{0\}^{28}$
FEFE FEFE FEFE FEFE	$\{1\}^{28}$	$\{1\}^{28}$
1F1F 1F1F 0E0E 0E0E	$\{0\}^{28}$	$\{1\}^{28}$
E0E0 E0E0 F1F1 F1F1	$\{1\}^{28}$	$\{0\}^{28}$

**Table 7.5:** Four DES weak keys.

$C_0$	$D_0$	semi-weak key pair (hexadecimal)	$C_0$	$D_0$
$\{01\}^{14}$	$\{01\}^{14}$	01FE 01FE 01FE 01FE, FE01 FE01 FE01 FE01	$\{10\}^{14}$	$\{10\}^{14}$
$\{01\}^{14}$	$\{10\}^{14}$	1FE0 1FE0 0EF1 0EF1, E01F E01F F10E F10E	$\{10\}^{14}$	$\{01\}^{14}$
$\{01\}^{14}$	$\{0\}^{28}$	01E0 01E0 01F1 01F1, E001 E001 F101 F101	$\{10\}^{14}$	$\{0\}^{28}$
$\{01\}^{14}$	$\{1\}^{28}$	1FFE 1FFE 0EFE 0EFE, FE1F FE1F FE0E FE0E	$\{10\}^{14}$	$\{1\}^{28}$
$\{0\}^{28}$	$\{01\}^{14}$	011F 011F 010E 010E, 1F01 1F01 0E01 0E01	$\{0\}^{28}$	$\{10\}^{14}$
$\{1\}^{28}$	$\{01\}^{14}$	E0FE E0FE F1FE F1FE, FEE0 FEE0 FEF1 FEF1	$\{1\}^{28}$	$\{10\}^{14}$

**Table 7.6:** Six pairs of DES semi-weak keys (one pair per line).

What is the meaning of these terms, and why should these keys not be used?

In the context of the Data Encryption Standard (DES), which is a symmetric encryption algorithm used for secure data communication, the terms "weak keys" and "semi-weak key pairs" refer to certain keys that exhibit specific properties which can make them susceptible to cryptographic attacks.

**Weak Keys:** Weak keys in DES are the keys that result in encryption or decryption operations that produce a pattern of repeated or easily predictable ciphertext. In other words, weak keys in DES generate a limited set of possible ciphertexts, which can significantly weaken the security of the encryption. There are a total of four weak keys in DES: 0x0101010101010101, 0xFEFEFEFEFEFEFEFE, 0xE0E0E0E0F1F1F1F1, and 0x1F1F1F1F0E0E0E0E.

**Semi-Weak Key Pairs:** Semi-weak key pairs in DES are pairs of keys that, when used together for encryption and decryption, can result in the same ciphertext being produced for two different plaintexts. This can lead to security vulnerabilities, as it violates the basic requirement of encryption where the same plaintext should not produce the same ciphertext when encrypted with the same key. There are a total of four semi-weak key pairs in DES.

These weak keys and semi-weak key pairs should not be used in DES because they can significantly weaken the security of the encryption. Attackers can take advantage of the patterns or repetitions in ciphertext generated by weak keys or semi-weak key pairs to launch attacks such as known-plaintext attacks or ciphertext-only attacks, which can potentially reveal the plaintext or the key itself, undermining the confidentiality and integrity of the encrypted data. Therefore, it is crucial to avoid using weak keys or semi-weak key pairs in DES or any other cryptographic algorithm to ensure robust security.

**HW3: Exercise 4.**

There are two variants of DES that were designed to enhance the security: DES-X and Triple DES. What are the basic mechanisms/ideas for each of them to achieve the goal of elevated security compared to original DES?

**DES-X:**

- **Whitening Step:** DES-X adds an additional "whitening" step to the original DES algorithm. A 64-bit constant value (referred to as the "XOR constant") is combined with the plaintext using a bitwise XOR operation before encryption and after decryption. This additional step helps to provide additional diffusion and confusion of the plaintext, making it harder for attackers to analyze the encryption process and recover the original plaintext.
- **Increased Effective Key Size:** DES-X increases the effective key size from 56 bits to 120 bits by XORing the 64-bit XOR constant with the original 56-bit key to produce a 120-bit key. This larger key size provides increased resistance to brute-force attacks, as the search space for possible keys is significantly larger, making it more computationally expensive and time-consuming for attackers to exhaustively search for the correct key.

**Triple DES (3DES):**

- **Multiple Encryption Rounds:** 3DES applies the DES encryption algorithm three times in a "cascade" fashion using three different 56-bit keys. The encryption can be done in various modes such as ECB (Electronic Codebook), CBC (Cipher Block Chaining), and others. This triple encryption process with multiple keys provides increased security compared to the original DES, as it adds more encryption rounds, making it harder for attackers to break the encryption through brute-force or cryptanalysis attacks.
- **Larger Key Size:** 3DES uses a larger key size compared to the original DES. The three 56-bit keys used in 3DES result in a key size of 168 bits, providing a much larger key space compared to the 56-bit key used in DES. This increased key size makes 3DES more resistant to brute-force attacks, as the search space for possible keys is significantly larger.

Both DES-X and Triple DES were designed to provide enhanced security compared to the original DES by adding additional steps or using larger key sizes. However, it's worth noting that these variants are considered outdated and less secure compared to modern encryption algorithms such as AES with larger key sizes. It's generally recommended to use modern encryption algorithms with larger key sizes for securing sensitive data in contemporary systems.

**HW3: Exercise 5.**

Find all equivalence classes for modulus  $m = 5$ , show at least six elements for each of them.

$$\left. \begin{aligned} \bar{0} &= \{\dots, -10, -5, 0, -5, 10 \dots\} \\ \bar{1} &= \{\dots, -9, -4, 1, -6, 11 \dots\} \\ \bar{2} &= \{\dots, -8, -3, 2, -7, 12 \dots\} \\ \bar{3} &= \{\dots, -7, -2, 3, -8, 13 \dots\} \\ \bar{4} &= \{\dots, -6, -1, 4, -9, 14 \dots\} \end{aligned} \right\} \mathbb{Z}_5$$

Are the following congruences correct? Why, or why not?

a)  $4 \equiv -24 \pmod{5}$  False  $\rightarrow 4 \not\equiv -24 \pmod{5}$  as  $5 \nmid (4 + 24)$

b)  $-2 \equiv 38 \pmod{5}$  True  $\rightarrow -2 \equiv 38 \pmod{5}$  as  $5 \mid (-2 - 38)$

$\rightarrow$  what really matters is the distance be divisible by 5

**HW3: Exercise 6.**

Consider integer ring  $\mathbb{Z}_{29}$ . Name the following:

- Neutral element with respect to addition  $\rightarrow \bar{0} + \bar{29} = \bar{29} \rightarrow 0$
- Neutral element with respect to multiplication  $\rightarrow \bar{1} + \bar{29} = \bar{29} \rightarrow 1$
- Additive inverse of element 5  $\rightarrow -5 \equiv \bar{24}$
- Additive inverse of 235  $\rightarrow 235 = 8 \cdot 29 + 3$ , so  $235 \equiv 3 \pmod{29}$   
 $26 \Rightarrow -3 \equiv 26 \pmod{29}$

**Example :** In  $\mathbb{Z}_{18}$

**Element 5** is invertible, as  $\gcd(5, 18) = 1$ .

The inverse  $5^{-1} = 11$ , as  $5 \cdot 11 = 55 = 3 \cdot 18 + 1 \equiv 1 \pmod{18}$ .

**Element 6** is not invertible, as  $\gcd(18, 6) = 6 (\neq 1)$ .

**Element 7** is invertible, as  $\gcd(7, 18) = 1$ .

The inverse  $7^{-1} = 13$ , as  $7 \cdot 13 \equiv 7 \cdot (-5) = -35 \equiv 1 \pmod{18}$ .

**Element 8** has no inverse, as  $\gcd(18, 8) = 2 (\neq 1)$ .

**HW3: Exercise 7. [ incomplete ]**

Is element 5 invertible in  $Z_{11}$ ,  $Z_{12}$  or  $Z_{13}$ ? Why?

Yes, because  $\gcd(5,11) = \gcd(5,12) = \gcd(5,13) = 1$

If yes, then find the multiplicative inverses of 5 in each integer ring.

You can do a trial-and-error search using a calculator or computer.

$$\gcd(5,11) = 1 \text{ and } 5 \cdot 9 = 45 \equiv 1 \pmod{11} \rightarrow 5^{-1} = 9 \text{ in } Z_{11}$$

$$\gcd(5,12) = 1 \text{ and } 5 \cdot 5 = 25 \equiv 1 \pmod{12} \rightarrow 5^{-1} = 5 \text{ in } Z_{11}$$

$$\gcd(5,13) = 1 \text{ and } 5 \cdot 8 = 40 \equiv 1 \pmod{13} \rightarrow 5^{-1} = 8 \text{ in } Z_{11}$$

**HW4: Exercise 1.**

Find some industrial standards or commercial applications in which AES is used.

The AES algorithm is widely used in a variety of applications, including wireless security, processor security, file encryption, and SSL/TLS. Some concrete examples are SSDs for data storage, Google Cloud storage service, internet browser programs such as Firefox and Opera, security certificates for websites. Many popular apps (such as Snapchat and Facebook Messenger) use AES encryption to safely send information such as photographs and messages.

AES is used in computing devices such laptops and smartphones for self-encrypting disk drivers , database encryption, storage encryption. Also in VPN implementations, file transfer protocols (FTPS, HTTPS...) , WI-FI security protocols, mobile apps (WhatsApp, Facebook Messenger and some other apps like password managers , video games and disk partition encryption )

- VPN implementations.
- File Transfer Protocols (FTPS, HTTPS, SFTP,OFTP,AS2,WebDAVS);
- Wi-Fi Security Protocols (WPA-PSK, WPA2-PSK);
- Programing language libraries.
- Mobile apps (WhatsApp, Facebook Messenger );
- File compressing tools.
- Some other apps like password tools, video games, disk partition encryption.

**HW4: Exercise 2.**

True or false?

a) Byte Substitution Layer provides confusion to AES. **True**

*" (...) Each layer has a unique function. Byte Substitution Layer introduces confusion by substituting each byte by another one using an S-box. (...)"*

b) Key length affects the block size of AES. **False**

*" (...) AES has block size 128 and options for key sizes 128, 192 and 256. It has an iterative structure with 10, 12 or 14 encryption rounds. The number of internal rounds depends on chosen key length. (...)"*

c) AES has an inherent key whitening step. **True**

*"(...) Key addition is the final step of each encryption round. It is also performed as the very first step of AES encryption, thereby creating an inherent key whitening step to AES. (...) "*

d) The irreducible polynomial for multiplications in  $GF(2^8)$  determined in the AES standard. **True**

*"(...) All arithmetic is done in the Galois extension field  $GF(2^8)$  (...)"*

#### HW4: Exercise 3.

a) What is AES T-box (or T-table)?

AES T-box, or T-table, is a lookup table used in the Advanced Encryption Standard (AES) algorithm. It is constructed during AES key expansion and is used in the SubBytes step to replace bytes in the input data block. The T-box provides non-linearity and diffusion properties critical for AES security. It is typically implemented as a 256-byte lookup table and is carefully designed to resist various attacks. Overall, the AES T-box is an important component of AES encryption, contributing to its strength and security.

*"the Rijndael designers proposed a method which results in fast software implementations. The core idea is to merge all round functions (except the rather trivial key addition) into one table look-up. This results in four tables, each of which consists of 256 entries, where each entry is 32 bits wide. These tables are named a T-Box. Four table accesses yield 32 output bits of one round. Hence, one round can be computed with 16 table look-ups."* From Understanding Cryptography page 130

b) What can be said about AES T-box / S-box lookup table implementations and side-channel attacks? [Well, a lot could be said. I'm expecting a short(ish) answer here.]

A side-channel attack is a security exploit that instead of targeting the program or its code, seeks to get information from the program or influencing it by evaluating the indirect effects. This type of attacks has been shown to be effective in breaking the secret key used in AES.

While AES has been deemed "secure enough" for most of today's standards, there has been various successful cache-based side-channel attacks.

**Trace-driven:** creates profiles of a cache hit or miss for every access to memory during an encryption.

**Access-driven:** Needs information only about which lines of cache have been accessed, not their precise order.

#### **Few notable methods:**

Prime + probe - fill cache with own data, wait victim to perform encryption and probe reload times.

Evict + Time - measure encryption time (T1), evict cache set, measure encryption time (T2) (same plaintext used). If  $T2 > T1$  = evicted line is used in encryption.



**HW4: Exercise 4.**

Find  $A^{-1}$  for AES byte  $A = 11001001$  from the AES multiplicative inverse lookup table. Verify by performing the multiplication  $A \cdot A^{-1} \pmod{x^8 + x^4 + x^3 + x + 1}$ .

$$A = 1100\ 1001_2 \rightarrow X = 1100_2 = C_{16}, y = 1001_2 = 9_{16} \rightarrow A^{-1} = 27_{16} = 0010\ 0111_2$$

Multiplicative inverse table in  $GF(2^8)$  for bytes  $xy$  used within the AES S-Box

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

The byte  $A^{-1}$  is 27 or 0010 0111 in binary.

By multiplying them, we get:

=

$$X^{12} + X^9 + X^8 + X^7 + X^{11} + X^8 + X^7 + X^6 + X^8 + X^5 + X^4 + X^3 + X^5 + X^2 + X + 1$$

$$X^{12} + X^{11} + X^9 + 3X^8 + 2X^7 + X^6 + 2X^5 + X^4 + X^3 + X^2 + X + 1$$

Since it is binary, we can use mod 2 on the coefficient

$$X^{12} + X^{11} + X^9 + X^8 + X^6 + X^4 + X^3 + X^2 + X + 1$$

$$\begin{array}{r}
 x^{12} + x^{11} + x^9 + x^8 + x^6 + x^4 + x^3 + x^2 + x + 1 \\
 + x^{12} \quad \quad \quad + x^8 + x^2 \quad \quad \quad + x^5 + x^4 \\
 \hline
 x^{11} + x^9 \quad \quad \quad + x^7 + x^6 + x^5 \quad \quad \quad + x^3 + x^2 + x + 1 \\
 + x^{11} \quad \quad \quad + x^7 + x^6 \quad \quad \quad + x^4 + x^3 \\
 \hline
 x^9 \quad \quad \quad + x^5 + x^4 \quad \quad \quad + x^2 + x + 1 \\
 + x^9 \quad \quad \quad + x^3 + x^4 \quad \quad \quad + x^2 + x \\
 \hline
 \text{① QED}
 \end{array}$$

So we verified that the table gave us the good answer.

**HW4: Exercise 5.**

Using the AES multiplicative inverse table, find the inverses for bytes 29, F3 and 01, where each byte is given in hexadecimal notation.

$$\text{S-Box } (29^{-1}) = \text{A0}$$

$$\text{S-Box } (F3^{-1}) = 34$$

$$\text{S-Box } (01^{-1}) = 01$$

Multiplicative inverse table in  $GF(2^8)$  for bytes  $xy$  used within the AES S-Box

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

**HW4: Exercise 6.**

Using the AES S-box lookup table, perform the Byte substitution operation for bytes 29, F3 and 01.

$$\text{S-Box } (29) = \text{A5}$$

$$\text{S-Box } (F3) = 0D$$

$$\text{S-Box } (01) = 7C$$

	y															
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
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

**HW5: Exercise 1.**

True or false?

a) ECB mode provides confidentiality and authenticity for the data being encrypted.

**False → only confidentiality**

b) In Cipher Block Chaining mode, each block of plaintext is encrypted using a different key.

**False → The same key is used all the time.**

c) Counter mode is not suitable for parallel encryption and decryption.

**False → each block can be decrypted and encrypted independently.**

d) CFB mode is not recommended for most applications due to its lack of diffusion.

**False → it provides diffusion, but it's slower and more complex than other modes, like CBC or GCM**

e) The choice of mode of operation in symmetric ciphers depends solely on the amount of data being encrypted.

**False → it depends on other factors , like security properties, the size of the data, available computer resources...**

**HW5: Exercise 2.**

True or false?

a) Galois Counter Mode (GCM) is a mode of operation for asymmetric-key cryptographic block ciphers. **False**

**"GCM mode is a mode of operation that provides confidentiality, authenticity and integrity of the data being encrypted.**

**In GCM mode, a block cipher (typically AES)(...)"**

b) In GCM mode, the message authentication code (MAC) is calculated using a symmetric key. **True**

**"(...) In GCM mode, a block cipher (typically AES) is used to encrypt a counter value, which is then XORed with the plaintext to generate the ciphertext. The counter value is encrypted using an IV. The GCM mode also calculates a message authentication code (MAC) (...)"**

c) The Galois field multiplier is used to calculate the encryption key in GCM mode. **False**

**That statement is partially correct. The Galois/Counter Mode (GCM) is a mode of operation for symmetric key encryption that provides authenticated encryption and is widely used in modern cryptography.**

d) GCM is commonly used in disk encryption and wireless communication. **True**

**"It is also used in disk encryption and wireless communication, among other applications. (...)"**

**HW5: Exercise 3.**

What is the role of an IV and why should we use a nonce and not a fixed sequence?

The use of IV provides with a nonce allows the encryption to be randomized, it helps to obtain nondeterminism and avoid “codebook” effect of EBC mode.

The role is to make sure every time plaintext is encrypted it is encrypted to a different ciphertext. Makes bruteforcing it and finding patterns much much harder. Nonce prevents plaintext attacks by encrypting the plaintext differently each time.

#### HW5: Exercise 4.

For which modes of operation are the following statements true?

a) It is used to build a stream cipher

b) It is deterministic (i.e., encrypts the same PT into the same CT every time)

Only ECB doesn't use on IV

c) In the encryption process, some data is fed back to the system.

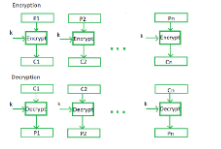
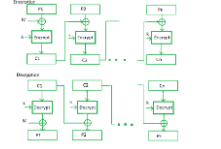
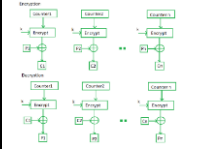
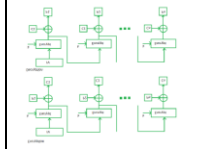
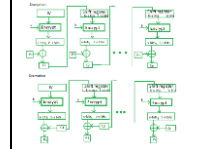
In CBC and in CFB, ciphertext block  $y_i$  is fed back.

In OFB, stream block  $s_i$  is fed back.

d) A change in one plaintext block affects the encryption of every following block.

e) Encryption process can be parallelized.

f) Decryption process can be parallelized.

	ECB: Electronic Codebook	CBC: Cipher Block Chaining	CTR: Counter	OFB: Output Feedback	CFB: Cipher Feedback
					
a			X	X	X
b	X				
c		X		X	X
d		X			X
e	X		X		
f	X	X	X		X

#### HW5: Exercise 5.

Suppose there is an error in the plaintext block  $x_i$  on Alice's side.

Which plaintext blocks are affected on Bob's side...

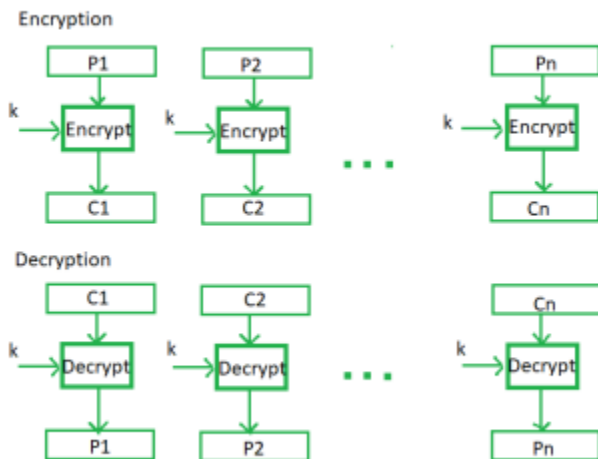
a) when using the ECB mode?

In the ECB mode, every block of plaintext is encrypted independently so only the ciphered block ( $y_i$ ) will be affected.

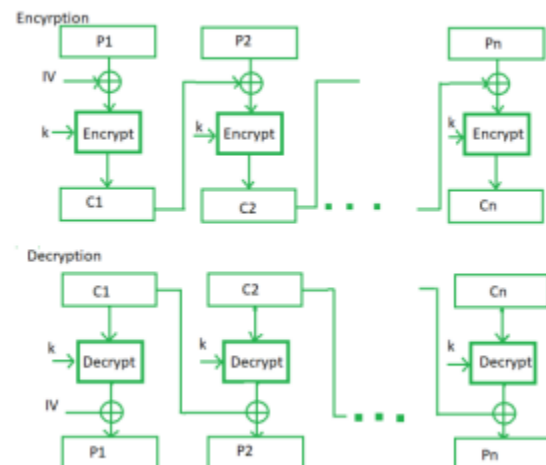
b) when using the CBC mode?

In the CBC mode, the encryption of all blocks is chained so every block after the corrupted one will also be corrupted, as the effect cascades to all following blocks.

**ECB:**



**CBC:**



**HW5: Exercise 6.** Suppose a bit error occurs in one ciphertext block  $y_i$  during the transmission. Which plaintext blocks are affected in decryption...

a) when using the ECB mode?

**$x_1$  only :** The error in one ciphertext block affects only the correspondent plaintext block during decryption. The other blocks are unaffected.

b) when using the CBC mode?

**$x_1$  and  $x_{i+1}$  :** Only the corrupted block and the following one ( i.e. the block the corrupted one is used to decrypt) are affected, as after that, the system only receives and uses uncorrupted blocks.

**HW5: Exercise 7.**

a) What are the benefits of using GCM mode over other modes of operation, such as CBC or ECB?

- GCM mode provides both confidentiality and authenticity, which is a significant advantage over CBC and ECB modes, which only provide confidentiality.
- GCM mode also provides high-speed encryption and decryption operation and is well-suited for application where high throughput is required.

b) What are some potential security concerns when using GCM mode, and how can these be addressed?

Some potential security concerns when using GCM mode include the reuse of nonces, which can lead to cryptographic collisions, and the use of weak key, which can be easily guessed. These concerns can be addressed by using a secure key generation and management process and by using a random nonce for each encryption operation.

**HW6: Exercise 1.**

Discuss the following:

a) What are the advantages of using symmetric ciphers over asymmetric ciphers for encryption and decryption?

→ Advantages of symmetric keys: speed , efficiency, security

Encryption and decryption processing is fast and has a big data throughput rate, with hardware encryption implementation reaching several hundred megabytes per second, and software can also reach megabytes per second throughput rates. The key is relatively short.

b) What are the advantages of using asymmetric ciphers over symmetric ciphers for encryption and decryption?

→ Advantages of asymmetric ciphers: key distribution, digital signatures, secure communication

- Each user in the network only needs to protect their private key. N users only need to generate N pairs of keys, fewer keys and easier to manage.
- Key distribution is simple and does not require secret channels and complex protocols to transmit keys.
- Digital signatures can be implemented -> non-repudiation.

**HW6: Exercise 2.**

Considering one-way functions, answer the following questions:

a) What is a trapdoor?

A trapdoor is a one-way function that is easy to calculate in one direction but inverting it , is very hard unless you know some secret information of the function called "trapdoor". Then calculating the inverse becomes very easy.

b) Give some examples of the uses of one-way functions in cryptography.

Using the Blum-Micali Generator, one-way functions can be used to construct Pseudo Random Number Generators, which enable us to construct Pseudo Random Functions (by using the GGM method for example), which in turn can be used to make Pseudo Random Permutations via the Luby-Rackoff construction.

(<https://crypto.stanford.edu/pbc/notes/crypto/oneway.html>)

**HW6: Exercise 3.**

Assume a company has 150 employees. The new company policy requires encrypted message exchange with a symmetric cipher. How many keys are required if encryption is needed for communication between any two employees?

$$n = 150 \quad \text{keys} = \frac{n(n-1)}{2} = \frac{150(150-1)}{2} = 11175 \text{ keys} \quad 149 \text{ keys per person}$$



**HW6: Exercise 4.**Find  $\gcd(7035, 864)$ 

a) based on their respective prime power factorizations

$$7035 = 3 \cdot 5 \cdot 7 \cdot 67$$

$$864 = 2^5 \cdot 3^3$$

$$\gcd(7035, 864) = 3$$

b) using the Euclidian algorithm.

$$7035^{(r0)} = 8 \cdot 864^{(r1)} + 123^{(r2)}$$

$$864^{(r1)} = 7 \cdot 123^{(r2)} + 3^{(r3)}$$

$$123^{(r2)} = 41 \cdot 3^{(r3)} + 0^{(r4)}$$

$$\gcd(7035, 864) = 3$$

**HW6: Exercise 5.**

Using Extended Euclidean Algorithm, compute the greatest common divisor of 198 and 243 and solve the Diophantine equation  $198s + 243t = \gcd(198, 243)$ .

$$198s + 243t = \gcd(243, 198)$$

Euclidean:

$$243 = 1 \cdot 198 + 45$$

$$198 = 4 \cdot 45 + 18$$

$$45 = 2 \cdot 18 + 9$$

$$18 = 2 \cdot 9 + 0$$

$$45 = 243 - 1 \cdot 198$$

$$18 = 198 - 4 \cdot 45$$

$$9 = 45 - 2 \cdot 18$$

$$\gcd(243, 198)$$

$$DE: 198s + 243t = 9$$

Diaphantine equation:

$$EEA: 9 = 45 - 2 \cdot 18$$

$$9 = 45 - 2(198 - 4 \cdot 45)$$

$$9 = 45 - 2 \cdot 198 + 8 \cdot 45$$

$$9 = 9 \cdot 45 - 2 \cdot 198$$

$$9 = 9 \cdot (243 - 198) - 2 \cdot 198$$

$$9 = 9 \cdot 243 - 9 \cdot 198 - 2 \cdot 198$$

$$9 = 9 \cdot 243 - 11 \cdot 198$$

Thus,  $\gcd(243, 198) = 9$ ,  $s = 9$ ,  $t = -11$  is a solution to  $9 = 243s + 198t$

**Example: Greatest common divisor (gcd) using Euclidean Algorithm (EA).**

**Note.** EA is very efficient and always terminates. The number of iterations is close to the number of digits of the input operands.

**Example.** a) Find  $\gcd(3910, 720)$  using EA. b) Find  $\gcd(12345, 284)$  using EA.

$$\begin{aligned} a) \quad r_0 \quad r_1 \quad r_2 \quad r_3 \quad r_4 \\ 3910 &= 5 \cdot 720 + 310 \\ 720 &= 2 \cdot 310 + 100 \\ 310 &= 3 \cdot 100 + 10 \\ 100 &= 10 \cdot 10 + 0 \\ \Rightarrow \gcd(3910, 720) &= 10 \end{aligned}$$

$$\begin{aligned} b) \quad r_0 \quad r_1 \quad r_2 \quad r_3 \quad r_4 \quad r_5 \quad r_6 \quad r_7 \quad r_8 \\ 12345 &= 43 \cdot 284 + 133 \\ 284 &= 2 \cdot 133 + 18 \\ 133 &= 7 \cdot 18 + 7 \\ 18 &= 2 \cdot 7 + 4 \\ 7 &= 1 \cdot 4 + 3 \\ 4 &= 1 \cdot 3 + 1 \\ 3 &= 3 \cdot 1 + 0 \\ \Rightarrow \gcd(12345, 284) &= 1 \quad (\text{coprimes}) \end{aligned}$$

**HW6: Exercise 6.**

Use Extended Euclidean Algorithm to find the inverse of 321 in  $\mathbb{Z}_{2707}$ .

*Note.* If you get a negative answer from this one, you have to add the modulus to it once, to bring your answer to range  $\{0, 1, \dots, 2706\}$ .

$$\begin{aligned} \text{EA:} \quad 2707 &= 8 \cdot 321 + 139 \\ 321 &= 2 \cdot 139 + 43 \\ 139 &= 3 \cdot 43 + 10 \\ 43 &= 4 \cdot 10 + 3 \\ 10 &= 3 \cdot 3 + 1 \\ 3 &= 3 \cdot 1 + 0 \\ \Rightarrow 1 &= 10 - 3 \cdot 3 \\ \Rightarrow 321^{-1} &\text{ exists in } \mathbb{Z}_{2707} \end{aligned}$$

$\gcd(2707, 321) = 1$   
 $\Rightarrow$  they are coprimes

$$\begin{aligned} 139 &= 2707 - 8 \cdot 321 \\ 43 &= 321 - 2 \cdot 139 \\ 10 &= 139 - 3 \cdot 43 \\ 3 &= 43 - 4 \cdot 10 \end{aligned}$$

$$\begin{aligned} \text{EEA:} \quad 1 &= 10 - 3 \cdot 3 \\ 1 &= 10 - 3 \cdot (43 - 4 \cdot 10) = 13 \cdot 10 - 3 \cdot 43 \\ 1 &= 13 \cdot (139 - 3 \cdot 43) - 3 \cdot 43 = 13 \cdot 139 - 42 \cdot 43 \\ 1 &= 13 \cdot 139 - 42 \cdot (321 - 2 \cdot 139) = 97 \cdot 139 - 42 \cdot 321 \\ 1 &= 97 \cdot (2707 - 8 \cdot 321) - 42 \cdot 321 = 97 \cdot 2707 - 818 \cdot 321 \\ 97 \cdot 2707 - 818 \cdot 321 &= 1 \quad || \quad (\text{mod } 2707) \quad \equiv 0 \quad (\text{mod } 2707) \\ 1889 \cdot 321 &= 1 \quad (\text{mod } 2707) \\ 321^{-1} &= 1889 \quad (\text{mod } 2707) \end{aligned}$$

$$-818 + 2707 = 1889$$

**HW7: Exercise 1.**

Find  $\varphi(n)$  for numbers  $n = 30, 301$ , and  $3001$ .



$$\begin{aligned}
 \varphi(30) &= (2 \cdot 3 \cdot 5) \\
 &= 2^{1-1} (2-1) \cdot 3^{1-1} (3-1) \cdot 5^{1-1} (5-1) \\
 &= 2^0 \cdot 1 \cdot 3^0 \cdot 2 \cdot 5^0 \cdot 4 \\
 &= 2 \cdot 4 \\
 &= 8
 \end{aligned}$$

$$\begin{aligned}
 \varphi(p \cdot q) &= (p-1)(q-1) \\
 \varphi(p \cdot q \cdot r) &= (p-1)(q-1)(r-1)
 \end{aligned}$$

$$\begin{aligned}
 \varphi(301) &= (7 \cdot 43) \\
 &= (7-1)(43-1) \\
 &= 6 \cdot 42 \\
 &= 252
 \end{aligned}$$

$$\begin{aligned}
 \varphi(3001) &= (3001) \rightarrow 3001 \text{ is a prime number so, } \varphi(p) = p-1 \\
 &= 3001^{1-1} (3001-1) \\
 &= 1 \cdot 3000 \\
 &= 3000
 \end{aligned}$$

**HW7: Exercise 2.**

Compute the inverse  $a^{-1} \bmod n$  with Fermat's Little Theorem (if applicable) or Euler's Theorem:

a)  $a = 4, n = 7$   *$n$  is a prime so we can use FLT:  $a^{p-1} \equiv 1 \pmod{p}$*   

$$\begin{aligned}
 a^{-1} &= a^{n-2} \\
 &= 4^{7-2} \\
 &= 4^5 \\
 &= 1024 \\
 &= 146 \cdot 7 + 2 \\
 &= 2 \pmod{7}
 \end{aligned}$$
 *$\Leftrightarrow a^{-1} = a^{n-x}$*   
*and to verify,  $4 \cdot 2 = 8 \equiv 1 \pmod{7}$ , 2 is the multiplicative inverse of 4 with  $m=7$*

b)  $a = 5, n = 12$   *$\rightarrow$  not a prime, have to use Euler:  $a^{\varphi(n)} \equiv 1 \pmod{n}$*   

$$5^{-1} \cdot 5^3 = 125 = 10 \cdot 12 + 5 \equiv 5 \pmod{12}$$
  
*5 is autoinverse*  
*to check:  $5 \cdot 5 = 25 = 2 \cdot 12 + 1 \equiv 1 \pmod{12}$*

c)  $a = 6, n = 13$

13, is prime,  
 FLT:  $a^{p-2} \equiv a^{-1} \pmod{p}$

$$\begin{aligned}
 a^{p-2} &\equiv 6^{11} \pmod{13} \\
 a^{p-2} &= 6^{11} \\
 a^{p-2} &= 362\,797\,056 \\
 a^{p-2} &= 27\,907\,465 \cdot 13 + 11 \\
 a^{p-2} &\equiv 11 \pmod{13}
 \end{aligned}$$

verification:  $6 \cdot 11 = 66 \equiv 1 \pmod{13}$

$$\begin{aligned}
 \text{if } n \in \mathbb{P} &\Rightarrow a^{-1} = a^{n-2} \pmod{n} \\
 \text{if } n \notin \mathbb{P} &\Rightarrow a^{-1} = a^{\varphi(n)-2} \pmod{n}
 \end{aligned}$$

**HW7: Exercise 3.**

Let primes  $p = 59$  and  $q = 47$  be given as set-up parameters for RSA. Which of the following are valid parameter choices for RSA public exponent  $e$ , and why / why not?

→ To check if  $e$  is a valid choice for the RSA public exponent, we need to check if it is relatively prime to the totient of  $n = pq$  and the totient is given by  $n$ ,  $\varphi(n) = (p-1)(q-1)$ .

$$n = pq = 59 \cdot 47 = 2773$$

$$\varphi(n) = (p-1)(q-1) = \varphi(2773) = (59-1)(47-1) = 2668$$

→ To check if  $e$  and  $\varphi(n)$  are relatively prime we can use the Euclidean algorithm to compute their greatest common divisor (GCD).

a)  $e = 17$

$$\text{EA: } 2668 = 156 \cdot 17 + 16$$

$$17 = 1 \cdot 16 + 1$$

$$16 = 16 \cdot 1 + 0$$

$$\gcd(e, \varphi(n)) = \gcd(17, 2668) = 1 \rightarrow e = 17 \text{ is a valid choice}$$

b)  $e = 23$

$$\text{EA: } 2668 = 116 \cdot 23 + 0$$

$$\gcd(e, \varphi(n)) = \gcd(23, 2668) = 23 \rightarrow 23 \text{ is not coprime with } 2668 \text{ so is not a valid choice}$$

c)  $e = 31$

$$\text{EA: } 2668 = 86 \cdot 31 + 2$$

$$31 = 15 \cdot 2 + 1$$

$$2 = 2 \cdot 1 + 0$$

$$\gcd(e, \varphi(n)) = \gcd(31, 2668) = 1 \rightarrow e = 31 \text{ is a valid choice}$$

**HW7: Exercise 4.**

A message was encrypted using the RSA public key pair  $(n, e) = (86609, 17)$ .

Given the set-up parameters  $p = 257$ ,  $q = 337$ , decrypt the ciphertext  $y = 12345$ .

$$n = 86609$$

$$\varphi(86609) = (p-1)(q-1) = 256 \cdot 336 = 86016 \quad \checkmark$$

~~So, we must find  $d$  such that  $e \cdot d \equiv 1 \pmod{86016}$~~   $d = 17^{-1} \pmod{86016}$  by EEA  
this yields  $d = 65777$ .

$$\begin{aligned} \text{Finally } x &= y^d = 12345^{65777} \\ &\pmod{86016} \\ &= 17^{-1} \pmod{86016} \end{aligned}$$

~~To decrypt cipher text using provide key  $(n, d): x = y^d \pmod{d}$ .~~

$$(\dots) - 2023 \cdot 13 = 1 \parallel \pmod{86016}$$

$$- 2023 + 86016 = 65777 \Rightarrow d$$

$$x = y^d \pmod{m} = 12$$

### HW7: Exercise 5.

Apply the square-and-multiply algorithm to compute  $x^e \pmod{m}$ , where  $x = 2$ ,  $e = 79$ ,  $m = 101$ . Show the exponent in binary notation after every iteration step.

$$y = 2^{79} \pmod{101}$$

$$79_{10} = 1001111_2$$

79 in binary is 1001111

Step		(mod 101)	Power of x in binary
0	$2^1$	2	$2^1$
1A	$2^2$	4	$2^{10}$
2A	$2^4$	16	$2^{100}$
3A	$2^8$	54	$2^{1000}$
3B	$2^9$	7	$2^{1001}$
4A	$2^{18}$	49	$2^{10010}$
4B	$2^{19}$	98	$2^{10011}$
5A	$2^{38}$	9	$2^{100110}$
5B	$2^{39}$	18	$2^{100111}$
6A	$2^{78}$	21	$2^{1011110}$
6B	$2^{79}$	42	$2^{1001111}$

**HW7: Exercise 6.**

How many random odd integers do we have to test on average, until we expect to find a prime factor  $p$  for RSA modulus  $n$  of size 2028?

For  $n$  to have size 2048,  $p$  must have length 1024. Therefore,

$$P(\tilde{p} \text{ is prime}) \approx \frac{2}{\ln(2^{1024})} = \frac{2}{1024 \cdot \ln 2} \approx \frac{1}{355}$$

So, we need to test approximately 355 candidates to hit a prime of length 1024 bits.

**HW7: Exercise 7.**

Shortly explain the following terms:

a) Hamming weight

*Hamming weight refers to the number of non-zero bits in a string of binary digits. For example, the Hamming weight of the binary number 101101 is 4.*

b) RSA trapdoor

*RSA trapdoor is a concept used in the RSA encryption scheme. It involves generating a public key that is easy to compute, but a corresponding private key that is difficult to compute without specific information or a "trapdoor" that allows the computation to be performed more efficiently.*

= RSA is based on the assumption that integer factorization is a one-way function. We assume that it is computationally infeasible to find the totient  $\phi(n)$  for a given large integer  $n$ , and therefore it is also computationally infeasible to find the inverse  $e^{-1}$  of an integer  $e \pmod{\phi(n)}$  without extra information. However, if one does know the factorization of  $n$ , then one can find  $e^{-1} \pmod{\phi(n)}$  using the Extended Euclidean Algorithm. This is the trapdoor for RSA.

c) false positive (in primality testing)

*In primality testing, a false positive occurs when a composite number is incorrectly identified as a prime number. This can happen when using probabilistic primality tests that have a small chance of incorrectly identifying a composite number as prime. If this happens, it could lead to creation of weak encryption keys that are easy to crack. This can be a huge risk.*

**HW8: Exercise 1.**

Find the members of the multiplicative group  $\mathbb{Z}_7^*$ . Find the order of each element. Are there any primitive elements in  $\mathbb{Z}_7^*$ ? Is  $\mathbb{Z}_7^*$  cyclic?

$|\mathbb{Z}_7^*| = 6 \rightarrow$  order of the group

$\mathbb{Z}_7^* = \{1, 2, 3, 4, 5, 6\} \rightarrow$  elements of the multiplicative group

$\mathbb{Z}_7^* = \{3, 5\} \rightarrow$  primitive elements (because they have maximum order)

$\mathbb{Z}_7^*$  is cyclic because it has **primitive elements**.

	$2^1 \equiv 2 \pmod{7}$ $2^2 \equiv 4 \pmod{7}$ $2^3 \equiv 1 \pmod{7}$	$3^1 \equiv 3 \pmod{7}$ $3^2 \equiv 2 \pmod{7}$ $3^3 \equiv 6 \pmod{7}$ $3^4 \equiv 4 \pmod{7}$ $3^5 \equiv 5 \pmod{7}$ $3^6 \equiv 1 \pmod{7}$	$4^1 \equiv 4 \pmod{7}$ $4^2 \equiv 2 \pmod{7}$ $4^3 \equiv 1 \pmod{7}$	$5^1 \equiv 5 \pmod{7}$ $5^2 \equiv 4 \pmod{7}$ $5^3 \equiv 6 \pmod{7}$ $5^4 \equiv 2 \pmod{7}$ $5^5 \equiv 3 \pmod{7}$ $5^6 \equiv 1 \pmod{7}$	$6^1 \equiv 6 \pmod{7}$ $6^2 \equiv 1 \pmod{7}$
<b>ord(1) = 1</b>	<b>ord(2) = 3</b>	<b>ord(3)=6</b>	<b>ord(4) = 3</b>	<b>ord(5) = 6</b>	<b>ord(6) = 2</b>

**HW8: Exercise 2.**

Compute public keys  $A$ ,  $B$  and the shared session key  $K_{AB}$  for the DHKE scheme with parameters  $p=467$  and  $\alpha=2$ , and

a)  $a = 3$ ,  $b = 5$

1	Alice's Key	$A = \alpha^a \pmod{p} = 2^3 \pmod{467} = 8$
2	Bob's Key	$B = \alpha^b \pmod{p} = 2^5 \pmod{467} = 32$
3	Alice and Bob exchange keys	-----
4	Alice's shared key	$K_{AB} = B^a \pmod{p} = 32^3 \pmod{467} = 78 \pmod{467}$
5	Bob's shared key	$K_{AB} = A^b \pmod{p} = 8^5 \pmod{467} = 78 \pmod{467}$

So, the public keys  $A$  and  $B$  are 8 and 32.

The shared key is  $K_{AB}$  is 432.

b)  $a = 228, b = 57$ 

1	Alice's Key	$A = \alpha^a \pmod{p} = 2^{228} \pmod{467} = 394$
2	Bob's Key	$B = \alpha^b \pmod{p} = 2^{57} \pmod{467} = 313$
3	Alice and Bob exchange keys	-----
4	Alice's shared key	$K_{AB} = B^a \pmod{p} = 313^{228} \pmod{467} = 206 \pmod{467}$
5	Bob's shared key	$K_{AB} = A^b \pmod{p} = 394^{57} \pmod{467} = 206 \pmod{467}$

So, the public keys  $A$  and  $B$  are 394 and 313.The shared key is  $K_{AB}$  is 206.

Other way to present it:

Let's assume that we have Alice and Bob. Alice private key is 228 and Bob's private key is 57!

Alice	Bob
<i>determining the public key</i>	
$A = \alpha^a \pmod{p} = 2^{228} \pmod{467} = 394$	$B = \alpha^b \pmod{p} = 2^{57} \pmod{467} = 313$
A	B
→	←
<i>compute shared keys</i>	
$K_{AB} = 313^{228} \pmod{467} = 206$	$K_{AB} = 394^{57} \pmod{467} = 206$

**HW8: Exercise 3.**Show that  $\alpha=4$  is not a primitive element in the field  $Z_{467}$ . $467$  is a prime so,  $\varphi(\varphi(467)) = 466$ 

Divisors of 466 = 1, 2, 233, 466

$$P = 467 \rightarrow |Z_{467}^*| = 466$$

Have to check if  $2^k \equiv 1 \pmod{467}$ 

$$4^1 = 4 \equiv 4 \pmod{467}$$

$$4^2 = 16 \equiv 16 \pmod{467}$$

$$4^{233} \equiv 1 \pmod{467} \rightarrow \text{because the result it's 1, this is the order of the element 4}$$

Thus,  $\text{ord}(4) = 233 \neq 466$  so 4 is not a primitive element of  $Z_{467}$

**HW8: Exercise 4.**

Compute the shared session key  $k_{AB}$  for DHKE parameters  $p=467$  and  $\alpha=4$  for

a)  $a = 400, b = 134$  and b)  $a = 167, b = 134$

You will find that both cases result in the same shared key. How is this possible?

a)  $a = 400, b = 134$

1	Alice's Key	$A = \alpha^a \pmod p = 4^{400} \pmod{467} = 89$
2	Bob's Key	$B = \alpha^b \pmod p = 4^{134} \pmod{467} = 51$
3	Alice and Bob exchange public keys	-----
4	Alice's shared key	$K_{AB} = B^a \pmod p = 236^{400} \pmod{467} = 161$
5	Bob's shared key	$K_{AB} = A^b \pmod p = 236^{134} \pmod{467} = 161$

So, the Alice's key is 89 and Bob's key is 51. The shared key  $K_{AB}$  is 38.

b)  $a = 167, b = 134$

1	Alice's Key	$A = \alpha^a \pmod p = 4^{167} \pmod{467} = 89$
2	Bob's Key	$B = \alpha^b \pmod p = 4^{134} \pmod{467} = 51$
3	Alice and Bob exchange public keys	-----
4	Alice's shared key	$K_{AB} = B^a \pmod p = 236^{167} \pmod{467} = 161$
5	Bob's shared key	$K_{AB} = A^b \pmod p = 272^{134} \pmod{467} = 161$

So, the Alice's key is 89 and Bob's key is 51. The shared key  $K_{AB}$  is 38.

Both cases result in the same shared key:

Even though the value for private exponent  $a$  is different, the public exponent  $A$  becomes the same as  $4^{400} \equiv 4^{167} \equiv 89 \pmod{467}$ . Why is that?

It's because 4 is not a primitive element of  $Z_{467}$  (this was verified in Problem 3). Instead, we know that  $4^{233} \equiv 1 \pmod{467}$ , hence

$$4^{400} = 4^{233+167} = 4^{233} \cdot 4^{167} \equiv 1 \cdot 4^{167} = 4^{167} \pmod{467}$$

**HW8: Exercise 5.**

In the DHKE protocol, the private keys are chosen from the set  $\{2, \dots, p-2\}$ . Discuss why the values 1 and  $p-1$  are excluded.

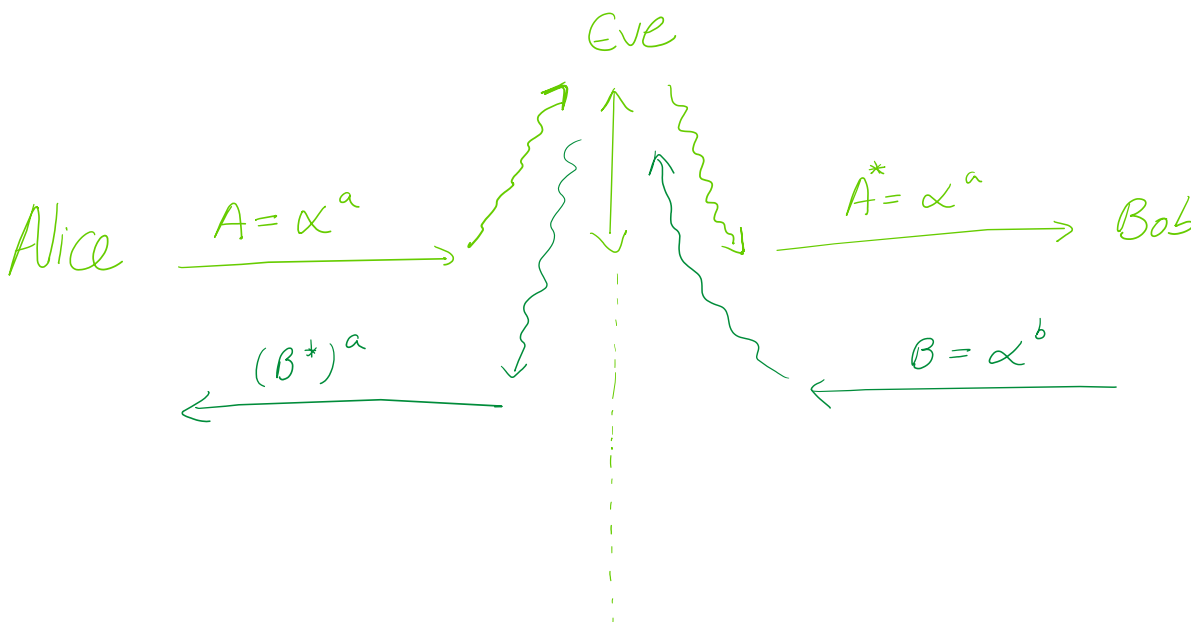
The public keys are  $A = \alpha^a$  and  $B = \alpha^b$  and the shared key is  $k_{AB} = \alpha^{ab}$ . Thus, if  $a = 1$  or  $b = 1$ , then either  $A = k_{AB}$  or  $B = k_{AB}$  and the shared key  $k_{AB}$  would be exposed during the protocol.

For  $a = p - 1$  or  $b = p - 1$ , the situation is similar because by Fermat's Little Theorem,  $\alpha^{p-1} \equiv 1 \pmod{p}$  for all  $p \in P$ .

**HW8: Exercise 6.**

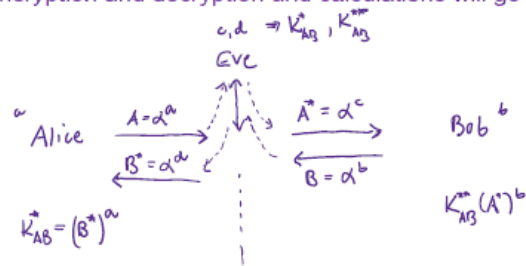
Public key algorithms without public key authentication are vulnerable to the so-called *man-in-the-middle* attack. Describe the attack on the basic DHKE protocol.

The man-in-the-middle attack is a type of attack that exploits the lack of public key authentication in the Diffie-Hellman Key Exchange (DHKE) protocol, a public key algorithm used for secure key exchange over an insecure communication channel. In this attack, the attacker intercepts the public keys exchanged between two parties and replaces them with his own public keys. The attacker then establishes separate keys with each party, posing as the other party. This allows the attacker to intercept and modify any messages exchanged between the two parties. To prevent this attack, it is essential to use public key authentication, such as digital signatures or certificates, to verify the identities of the parties involved in the communication.





DHKE is vulnerable to man-in-the-middle (MITM) attacks when the keys are exchanged. If Eve can manipulate the extension between Alice and Bob while they are exchanging keys, the security is broken. Alice sends  $\alpha^a \pmod{p}$ , Eve intercepts and send to Bob  $\alpha^c \pmod{p}$  instead. And similarly, Alice will receive  $\alpha^d \pmod{p}$  instead of  $\alpha^b \pmod{p}$  that Bob sent to her. Alice will compute a key  $\alpha^{ad}$  while Bob will compute a key  $\alpha^{bc}$ . Eve will be able to decrypt a message when it is sent and encrypt it again to send forward with corresponding keys to Alice's and Bob's keys. Alice and Bob won't even realize that something is wrong, since on their end encryption and decryption and calculations will go as expected.



**HW9: Exercise 1.**

Solve for  $x$  (if possible) the Discrete logarithm problem  $\alpha^x = \beta \pmod{p}$  when

a)  $\alpha = 5, \beta = 2, p = 7$

$$5^1 = 5 \neq 2 \pmod{7}$$

$$5^2 = 4 \neq 2 \pmod{7}$$

$$5^3 = 6 \neq 2 \pmod{7}$$

$$5^4 = 2 \pmod{7}$$

$$\left( \begin{array}{l} 5^5 = 3 \neq 2 \pmod{7} \\ 5^6 = 1 \neq 2 \pmod{7} \end{array} \right)$$

By brute force, we can know that  $x = 4$

b)  $\alpha = 2, \beta = 5, p = 7$

$$2^1 = 2 \neq 5 \pmod{7}$$

$$2^2 = 4 \neq 5 \pmod{7}$$

$$2^3 = 1 \neq 5 \pmod{7}$$

$$2^4 = 2 \neq 5 \pmod{7}$$

$$2^5 = 4 \neq 5 \pmod{7}$$

$$2^6 = 1 \neq 5 \pmod{7}$$

By brute force, we can assume that there is no integer value of  $x$  such that  $2^x = 5 \pmod{p}$  in  $\mathbb{Z}_7^*$

$\alpha=2$  is not a generator/primitive in  $\mathbb{Z}_7^*$

c)  $\alpha = 2, \beta = 4, p = 7$

$$2^1 = 2 \neq 4 \pmod{7}$$

$$2^2 = 4 \pmod{7}$$

$$2^3 = 1 \neq 4 \pmod{7}$$

$$2^4 = 2 \neq 4 \pmod{7}$$

$$2^5 = 4 \pmod{7}$$

$$2^6 = 1 \neq 4 \pmod{7}$$

$\beta=4$  is one of the elements that  $\alpha=2$  does generate in  $\mathbb{Z}_7^*$   
 $\Rightarrow$  we can find solutions  
 $x=2, x=5$

**HW9: Exercise 2.**

Learn about computational Diffie-Hellman assumption (CDH) and about decisional Diffie-Hellman assumption (DDH), and state them in a simple manner. What is the relationship between CDH and DDH?

The relationship between CDH and DDH is that CDH is a stronger assumption than DDH. If CDH is true, then DDH is also true. However, the converse is not necessarily true. In other words, if DDH is false, it does not necessarily mean that CDH is false. This is because the CDH assumption is a stronger requirement that requires not only the decisional hardness of the Diffie-Hellman problem, but also the computational hardness of it.

CDH states that in a cyclic group  $G$  with a primitive element  $\alpha$ , given a triple  $(\alpha, \alpha^a, \alpha^b)$ , where  $a, b$  are some random elements of  $G$ , it is computationally infeasible to find  $\alpha^{ab}$ .

DDH states that in a cyclic group  $G$  with a primitive element  $\alpha$ , the following 4-tuples are computationally indistinguishable:

$(\alpha, \alpha^a, \alpha^b, \alpha^{ab})$  where  $a, b$  are some random elements of  $G$ , and

$(\alpha, \alpha^a, \alpha^b, \alpha^c)$  where  $a, b, c$  are some random elements of  $G$

It is believed that CDH is a harder problem than DDH. If CDH was solved, then DDH would be trivial. But it may be that there are groups where DDH is solvable but still does not give away CDH.

**HW9: Exercise 3.**

Given an elliptic curve  $E: y^2 = x^3 + x + 7$  over  $Z_{17}$  and points  $P = (1,3)^{x_1,y_1}$  and  $Q = (2,0)^{x_2,y_2}$ , find  $P + Q$ .

$$S = \frac{y_2 - y_1}{x_2 - x_1} = \frac{0 - 3}{2 - 1} = \frac{-3}{1} = -3 \equiv 14 \pmod{17}$$

$$x_3 = S^2 - x_1 - x_2 = 14^2 - 1 - 2 = 193 \equiv 6 \pmod{17}$$

$$y_3 = S(x_1 - x_3) - y_1 = 14(1 - 6) - 3 = -73 \equiv 12 \pmod{17}$$

So,  $P + Q = (6, 12)$

To check id  $(6,12) \in Z_{17}$ :

$$12^2 \equiv 6^3 + 6 + 7 \pmod{17}$$

$$12^2 = 144 \equiv 8 \pmod{17} \text{ and } 6^3 + 6 + 7 = 229 \equiv 8 \pmod{17}$$

**HW9: Exercise 4.**

Consider the elliptic curve  $E: y^2 = x^3 + 4x^a + 20^b$  over  $GF(29^b)$ . Find

a) the inverse of  $P = (8,10)$

$$-P = (xp, p - yp) = (8, 29 - 10) = (8, 19)$$

To check if  $-P \in GF(29)$ :

$$19^2 = 8^3 + 4 \cdot 8 + 20 \pmod{29}$$

$$19^2 = 361 \equiv 13 \pmod{29} \text{ and } 8^3 + 4 \cdot 8 + 20 = 564 \equiv 13 \pmod{29}$$

b)  $2P$

$$S = \frac{3x_1^2 + a}{2y_1} = \frac{3 \cdot 8^2 + 4}{2 \cdot 10} = \frac{196}{20} = 196 \cdot 20^{-1} = 22 \cdot 16 = 352 \equiv 4 \pmod{29}$$

$$x_3 = S^2 - x_1 - x_2 = 4^2 - 8 - 8 = 0 \equiv 0 \pmod{29}$$

$$y_3 = S(x_1 - x_3) - y_1 = 4(8 - 0) - 10 \equiv 22 \pmod{29}$$

So,  $2P = (0, 22)$

To check id  $(0,22) \in Z_{29}$ :

$$22^2 \equiv 0^3 + 4 \cdot 0 + 20 \pmod{29}$$

$$22^2 = 484 \equiv 20 \pmod{29} \text{ and } 0^3 + 4 \cdot 0 + 20 \equiv 20 \pmod{29}$$

HW9: Exercise 5.

Consider the elliptic curve  $E: y^2 = x^3 + ax + b$  over  $GF(29)$ .  
For  $P = (8,10)$ , use the Double-and-add algorithm to find  $9P$ . (Answer:  $9P = (4,10)$ ).  
(You can use an EC point calculator to perform point computations.)

Parameters

x-point (Px):  
8

n:  
9P

a: 4

b: 20

Prime: 29

a= 4  
b= 20  
p= 29  
n= 9  
x-point= 8

P (8,10) Point is on curve  
2P (0,22) Point is on curve  
3P (16,2) Point is on curve  
4P (6,17) Point is on curve  
5P (20,3) Point is on curve  
6P (10,25) Point is on curve  
7P (2,6) Point is on curve  
8P (13,6) Point is on curve  
9P (4,10) Point is on curve

HW9: Exercise 6.

Let  $a = 2, b = 9, p = 17$  and  $P = (4,8)$  be the domain parameters for the ECDH protocol.  
Alice and Bob will use the ECDH to agree on a joint session key  $T_{AB}$ .  
Their respective choices for private keys are  $a = 10, b = 13$ .  
Show the steps of the protocol.  
(You can use an EC point calculator to perform point computations.)

	Alice	Bob
private keys	$a = 10$	$b = 13$
public keys	Compute $A = 10 \cdot P$	Compute $B = 13 \cdot P$
	$A (2,15) \rightarrow B (9,12)$	
	Compute $T_{AB} = 10B = (5,5)$	Compute $T_{AB} = 13A = (5, 12)$

Parameters

x-point (Px):

n:

a:

b:

Prime:

Determine

```

a= 2
b= 9
p= 17
n= 13
x-point= 4
P      (4,8)      Point is on curve
2P     (10,3)     Point is on curve
3P     (7,3)      Point is on curve
4P     (5,5)      Point is on curve
5P     (0,14)     Point is on curve
6P     (11,11)    Point is on curve
7P     (6,13)     Point is on curve
8P     (9,5)      Point is on curve
9P     (3,5)      Point is on curve
10P    (2,15)     Point is on curve
11P    (2,2)      Point is on curve
12P    (3,12)     Point is on curve
13P    (9,12)     Point is on curve

```

Parameters

x-point (Px):

n:

a:

b:

Prime:

Determine

```

a= 2
b= 9
p= 17
n= 10
x-point= 9
P      (9,12)     Point is on curve
2P     (0,14)     Point is on curve
3P     (7,14)     Point is on curve
4P     (2,15)     Point is on curve
5P     (10,3)     Point is on curve
6P     (11,6)     Point is on curve
7P     (6,13)     Point is on curve
8P     (4,9)      Point is on curve
9P     (3,12)     Point is on curve
10P    (5,5)      Point is on curve

```

Parameters

x-point (Px):

n:

a:

b:

Prime:

Determine

```

a= 2
b= 9
p= 17
n= 13
x-point= 2
P      (2,2)      Point is on curve
2P     (4,8)      Point is on curve
3P     (3,12)     Point is on curve
4P     (10,3)     Point is on curve
5P     (9,12)     Point is on curve
6P     (7,3)      Point is on curve
7P     (6,4)      Point is on curve
8P     (5,5)      Point is on curve
9P     (11,6)     Point is on curve
10P    (0,14)     Point is on curve
11P    (0,3)      Point is on curve
12P    (11,11)    Point is on curve
13P    (5,12)     Point is on curve

```

**HW10: Exercise 1.**

Given an RSA signature scheme with the public key ( $n=9797, e=131$ ), which of the following signatures are valid?

*Proof of correctness :*  $s^e \equiv x \pmod{n}$

a) ( $x=123, s=6292$ )

$$x' = s^e = 6292^{131} \pmod{9797} \equiv 123$$

$x \equiv x' \leftrightarrow 123 \equiv 123$  ✓ is valid

b) ( $x=4333, s=4768$ )

$$x' = s^e = 4768^{131} \pmod{9797} \equiv 9644$$

$x \equiv x' \leftrightarrow 4333 \equiv 9644$  ✗ is not valid

c) ( $x=4333, s=1424$ )

$$x' = s^e = 1424^{131} \pmod{9797} \equiv 4333$$

$x \equiv x' \leftrightarrow 4333 \equiv 4333$  ✓ is valid

**HW10: Exercise 2.**

Bob wants to sign message  $x=12345$  and send it to Alice.

Bob's public key is  $(n_B, e_B) = (86609, 17)$  with set-up parameters  $p=257, q=337$ .

Find Bob's private key and show the computations and steps of interaction between Alice and Bob in the RSA signature protocol.

$$\varphi(n) = (p-1)(q-1) = 257 - 337 = 86016$$

$$ed \equiv e^{-1} \pmod{\varphi(n)}$$

$$\leftrightarrow d = e^{-1} \pmod{\varphi(n)}$$

$$\leftrightarrow d = 17^{-1} \pmod{\varphi(86016)} = 65777$$

Bob's private key  $(n_B, d_B) = (86609, 65777)$

Alice		Bob
	$\leftarrow (n, e) = (86609, 17)$	
	$\leftarrow (x, s) = (12345, 5361)$	$s = 12345^{65777} \equiv 5361 \pmod{86609}$

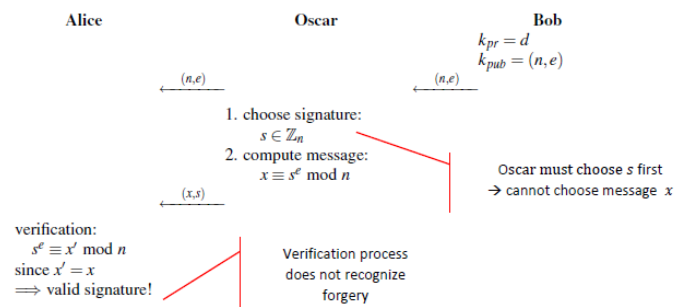
Verification  $x' = 5361^{17} \equiv 12345 \pmod{86609}$

If  $x = x'$  it's valid so, it's valid because  $x = 12345$  and  $x' = 12345$

**HW10: Exercise 3.**

Given an RSA signature scheme with the public key  $(n=9797, e=131)$ , show how Oscar can perform an existential forgery attack by providing an example of one for the parameters of the RSA digital signature scheme.

Existential forgery for RSA signature protocol:



If Oscar can control the message  $x$ , he has access to  $x^e \pmod{n}$ .

Also, if he has access to  $e$  and  $n$  values, he knows Bob's public key and can this way send messages to Alice that would seem valid.

Bob's public key:  $x = 9797, e = 131$

Oscar's fake signature:  $s = 5051$

Oscar compute a matching "message"  $x$ :  $x \equiv 5051$

(...)

**HW10: Exercise 4.**

What are the current recommendations for bit lengths for DSA parameters  $p$  and  $q$ ?

What are the corresponding signature lengths?

Lengths for  $p$  and  $q$  are  $L$  and  $N$ . Possible values:

- 1024 and 160 should not be used anymore.
- 2048 and 224 should not be used to generate new signatures, but signatures can be verified.
- 2048 and 256 can be used.
- 3072 and 256 can be used.

Length of signature is  $L/4$  bytes, so for the used lengths that would be  $256/4 = 64$  bytes = 512 bits.

Source: <https://pycryptodome.readthedocs.io/en/latest/src/signature/dsa.html>

For $2^{80}$	$p = 1024, q = 160$ signature length = $160 + 160 = 320$
For $2^{112}$	$p = 2048, q = 224$ signature length = $224 + 224 = 448$
For $2^{128}$	$p = 3072, q = 256$ signature length = $256 + 256 = 512$

**HW10: Exercise 5.**

In the example for DSA key generation, signature and verification process, verify that the chosen parameter  $\alpha=3$  has the proper order with respect to chosen parameters  $p$  and  $q$ .

$p = 59$

$|Z_{59}^*| = 58$  and 58 has the factors 1, 2, 29

Check if  $3^1, 3^2, 3^{29} \pmod{59} \rightarrow \text{ord}(\alpha) = 29 = q$

$3^{29} = 1 \pmod{59}$

$\text{ord}(3) = 29 \rightarrow$  so, it's true

**HW10: Exercise 6.**

Given DSA parameters  $p=59$ ,  $q=29$ ,  $\alpha=3$  and Bob's private key  $d=23$ , show the steps of the signing process (Bob) and verification (Alice) for hash value  $h(x)=17$  and ephemeral key  $k_E=25$ .

Alice		Bob
		choose $p = 59$ choose $q = 29$ choose $\alpha = 3$ choose private key $d = 23$ $\beta = \alpha^d \equiv 45 \pmod{59}$
	$\leftarrow$ $(p, q, \alpha, \beta)$ $=$ $(59, 29, 3, 45)$	
		Sign:  Compute hash of message $h(x) = 17$  Choose ephemeral key $k_E = 25$ $r = (3^{25} \pmod{59}) \equiv 22 \pmod{29}$ $s = (17 + 23 \cdot 22) \cdot 7$ $\equiv 7 \pmod{29}$
	$\leftarrow$ $(x, (r, s))$ $=$ $(x, (22, 7))$	
Verify: $w = 7^{-1} \pmod{29} \equiv 7^{29-2}$ $\pmod{29} \equiv 25 \pmod{29}$ $u_1 = 25 \cdot 17 \equiv 19 \pmod{29}$ $u_2 = 25 \cdot 22 \equiv 28 \pmod{29}$ $v = (3^{19} \cdot 45^{28} \pmod{59}) \pmod{29}$ $29 \equiv 22 \pmod{29}$ $v \equiv r \rightarrow \text{valid signature}$		



**HW11: Exercise 1.**

True or false?

a) A hash function is a mathematical function that converts a variable-sized message into a fixed-sized output.

**True →** A hash function transforms a message of any size into a fixed-sized output, known as a hash value. It is designed to be computationally infeasible to derive the original message from the hash value or to find two different messages that produce the same hash value, making it useful for data integrity verification and cryptographic applications.

b) A hash function can be used to encrypt a message so that it cannot be read by unauthorized parties.

**False →** Hash functions are used for data integrity verification. Hashing does not provide confidentiality, meaning it does not make the message unreadable to unauthorized parties. Instead, it produces a fixed-sized output that can be used to verify that the original message has not been tampered with or altered.

c) Hash functions can be used for digital signatures, message authentication codes, and password storage.

**True →** Hash functions are used in digital signatures to ensure the integrity and authenticity of a message. They are also used in message authentication codes (MACs) to provide message authentication and integrity. Finally, hash functions are commonly used in password storage to protect user passwords by hashing them before storing them in a database.

d) A hash function can be used to generate random numbers.

**False →** Hash functions are not designed for generating random numbers. They are deterministic functions that always produce the same output for the same input. While the output of a hash function may appear random, it is still derived from the input, and therefore not truly random. To generate random numbers, other techniques such as using a random number generator, or a hardware-based random source should be used.

e) A hash function can be used to check the integrity of a file by comparing the hash value of the original file with the hash value of the downloaded file.

**True →** Hash functions can be used to verify the integrity of a file by generating a hash value of the original file and comparing it with the hash value of the downloaded file. If the two hash values match, it indicates that the file has not been modified or corrupted during the download process.

**HW11: Exercise 2.**

Shortly explain the meaning of the following terms with respect to hashing:

- a) a collision: A collision occurs when two different input messages produce the same hash value or digest. This is an undesirable property for a hash function because it can compromise the integrity and security of the hashed data.
- b) clustering: Clustering refers to the phenomenon in which many input messages produce hash values that are located close together in the hash space. This can make it easier for an attacker to find collisions or perform other attacks on the hash function.
- c) resolving collisions: Resolving collisions refers to the process of dealing with collisions in a hash function. This can be done using various techniques, such as chaining or open addressing, which aim to ensure that every input message is mapped to a unique hash value. The goal is to minimize the likelihood of collisions occurring and to maintain the security and integrity of the hashed data.

**HW11: Exercise 3.** There are two main approaches to resolving collisions: *chaining* and *open addressing*. Shortly explain the basic principle of each approach.

Chaining involves creating a linked list of values that hash to the same bucket, while open addressing involves searching for the next available bucket in the hash table when a collision occurs. The goal of both approaches is to ensure that every input message is mapped to a unique hash value, even in the presence of collisions.

**HW11: Exercise 4.** Consider a hypothetical hash function with output length 160 bits. Find the approximate number of hash values needed to find a collision with success rate

a)  $\lambda=0.5$

$$t \approx 2^{\frac{n+1}{2}} \sqrt{\ln\left(\frac{1}{1-\lambda}\right)} = 2^{\frac{160+1}{2}} \sqrt{\ln\left(\frac{1}{1-0.5}\right)} \simeq 1.42 * 10^{24} \simeq 8^{80}$$

b)  $\lambda=0.9$

$$t \approx 2^{\frac{n+1}{2}} \sqrt{\ln\left(\frac{1}{1-\lambda}\right)} = 2^{\frac{160+1}{2}} \sqrt{\ln\left(\frac{1}{1-0.9}\right)} \simeq 2.6 * 10^{24} \simeq 8^{81}$$

**HW11: Exercise 5.** What are (one-way) compression functions and what is their role in building hash functions? Name some popular compression functions.

They transform two fixed-length inputs into a fixed-length output. "one-way" means in this context that given output, it's difficult to compute the inputs that compress to the output.

Pros include ease of computing, collision resistance (should be hard to find any 2 inputs that compress into same output) and preimage-resistance if an attacker only knows the output, it should be infeasible to calculate an input.

Notable functions:

Davies-Meyer (block cipher, XOR input + encryption + prev hash as key)

Matyas-Meyer-Oseas (similar to above but now input is the key)

Miyaguchi-Preneel (output = XOR input + prev hash val + encryption of input with prev hash as key)

**HW12: Exercise 1.**

True or false?

a) MACs are used to provide message authentication and integrity. **True**

MACs are used to provide message authentication and integrity by generating a fixed-size output that can be used to verify the integrity of a message and authenticate its sender.

b) MACs can be used to provide confidentiality to a message. **False**

MACs do not provide confidentiality to a message as they only ensure message authentication and integrity, not encryption.

c) MACs can be used to verify the identity of the sender of a message to the receiver. **True**

MACs can be used to verify the identity of the sender of a message to the receiver by generating a MAC that can be verified using a shared secret key between the sender and receiver.

d) MACs can be used to verify the identity of the sender of a message to a third party. **False**

MACs are typically used for authentication and integrity within a specific communication channel between a sender and receiver, and are not intended for verifying the identity of the sender to a third party.

**HW12: Exercise 2.**

Explain shortly:

a) the difference between a MAC and a digital signature:

The main difference between a MAC and a digital signature is that a MAC uses a shared secret key between the sender and receiver to generate a fixed-size output that verifies the integrity of a message and authenticates its sender, while a digital signature uses the sender's private key to generate a unique digital signature that can be verified using the sender's public key, providing non-repudiation in addition to integrity and authentication.

b) the difference between a hash function and a MAC:

The main difference between a hash function and a MAC is that a hash function is a one-way function that takes an input and produces a fixed-size output (hash value) that is unique to the input, used for data integrity checks, while a MAC is a cryptographic technique that uses a secret key and a hash function to generate a fixed-size output that can verify the integrity of a message and authenticate its sender, providing message authentication in addition to data integrity.

**HW12: Exercise 3.**

Consider a MAC protocol where the sender performs the following operation:

$$y = e_{k1}[x || h(k2 || x)]$$

here  $x$  is the message,  $h$  is the hash function,  $e$  is a symmetric encryption algorithm, and " $||$ " means concatenation.  $k1$  and  $k2$  are secret keys shared by the sender and the receiver. Sender will transmit  $y$  to receiver. Give a step-by-step description of what the receiver will do upon receiving  $y$  to verify the MAC.

**How the receiver verifies the MAC in the given protocol:**

1. The receiver splits the received value into two parts:  $y1$  and  $y2$ .
2. The receiver uses a secret key  $k1$  to decrypt  $y1$  and obtain  $x'$ .
3. The receiver computes  $h(k2 || x')$  by combining  $k2$  and  $x'$ , and applying a hash function.
4. The receiver compares the computed  $h(k2 || x')$  with  $y2$ .
5. If the computed value matches  $y2$ , the MAC is considered valid, and the message is accepted as authentic. If not, the MAC is considered invalid, and the message is rejected as potentially tampered with.

Optionally, the receiver may perform additional checks, such as checking freshness or integrity of the message using other means.

**HW12: Exercise 4.**

How is CMAC similar to CBC-MAC, and how do they differ from each other?

CMAC (Cipher-based Message Authentication Code) and CBC-MAC (Cipher Block Chaining Message Authentication Code) are both cryptographic techniques used to generate message authentication codes, but they differ in their construction and usage.

**Similarities:**

- Both CMAC and CBC-MAC use a symmetric key-based block cipher to generate the authentication code.
- Both techniques provide message authentication and integrity, allowing the receiver to verify that a message has not been tampered with during transmission.

**Differences:**

- CMAC uses a double-length key, derived from the original key, and applies a block cipher in a modified way, allowing for more efficient and secure message authentication compared to CBC-MAC.
- CBC-MAC operates in a chaining mode, where each block of the message is XORed with the previous block's ciphertext before encryption, while CMAC uses a combination of XORing and bit manipulation operations to achieve improved security.
- CMAC supports variable-length messages, while CBC-MAC typically requires padding or truncation of the message to fit into fixed-size blocks.
- CMAC has been standardized, while CBC-MAC is a more general concept that can be implemented in various ways.

In summary, CMAC and CBC-MAC are similar in their purpose of generating message authentication codes, but differ in their construction, usage, and security features. CMAC is a more modern and widely used technique, known for its efficiency and security, while CBC-MAC is a more general concept that can be implemented in different ways.

**HW12: Exercise 5.**

Explain shortly, what is GMAC and how does it compare to other types of MACs in terms of efficiency and security.

GMAC (Galois/Counter Mode) is a cryptographic technique used to generate message authentication codes (MACs) that provide both message authentication and confidentiality. GMAC is typically used in combination with a block cipher operating in counter mode (CTR), such as AES-CTR, to provide authenticated encryption.

**Efficiency:**

- GMAC is known for its efficiency as it can be highly parallelized, allowing for fast processing of large data streams.
- GMAC does not require padding of the message, which can result in more efficient computation compared to other MACs that operate on fixed-size blocks.
- GMAC can be implemented using hardware-accelerated instructions available in modern processors, further enhancing its efficiency.

**Security:**

- GMAC provides strong security, as it is based on the Galois field multiplication, which is highly resistant to various attacks.

- GMAC uses a nonce (a unique number) combined with a counter to generate a unique input for each block, providing protection against replay attacks.
- GMAC has been thoroughly analyzed and proven to be secure when used correctly in authenticated encryption schemes.

In comparison to other MACs, GMAC is generally considered to be efficient and secure. It provides both message authentication and confidentiality, making it suitable for applications where data integrity and privacy are important, such as secure communication over networks or storage of sensitive data. However, it is important to implement GMAC correctly, including proper key management and nonce handling, to ensure its security properties are maintained.

**HW13: Exercise 1.** True or false?

- a) In symmetric-key cryptography, key establishment typically involves the use of a key distribution center (KDC) to manage the distribution of secret keys to users. **True**
- b) In asymmetric-key cryptography, key establishment typically involves the use of public-key cryptography to securely exchange public keys between two or more parties. **True.**
- c) Key establishment is not necessary for secure communication in a cryptographic system. **False.** Key establishment is necessary for secure communication in a cryptographic system, as it involves securely generating, distributing, and exchanging keys among parties to ensure confidentiality, integrity, and authentication of the communication.
- d) The key derivation function (KDF) is used to generate new cryptographic keys from a shared secret or other input. **True.**

**HW13: Exercise 2.** True or false?

- a) Password-Based Key Derivation Function 2 (PBKDF2) is a type of KDF that is commonly used to derive cryptographic keys from passwords. **True.**
- b) In a public key infrastructure (PKI), the root CA issues certificates directly to end-users. **False.**  
In a PKI, the root CA issues certificates to intermediate CAs, which in turn issue certificates to end-users, creating a hierarchical trust model.
- c) The Diffie-Hellman key exchange is an example of a key agreement protocol used in asymmetric-key cryptography. **True.**
- d) In a public-key infrastructure (PKI), a digital certificate contains a public key and other identifying information about the owner of the certificate. **True.**

**HW13: Exercise 3.**

Explain the Man-in-the-Middle attack against the Diffie-Hellman Key Exchange protocol, if the public keys are not authenticated.

In a Man-in-the-Middle (MITM) attack against the Diffie-Hellman Key Exchange protocol, if the public keys are not authenticated, an attacker can intercept the communication between two parties and secretly establish separate connections with each party using their own public keys. The attacker can then intercept and modify the messages exchanged between the parties, effectively decrypting, reading, and possibly modifying the communication without either party's knowledge, because the parties are unknowingly using the attacker's public key instead of each other's authenticated public keys. This allows the attacker to eavesdrop on the communication and potentially manipulate it to their advantage, compromising the confidentiality and integrity of the exchanged data.

**HW13: Exercise 4.**

Give some basic information about the Finnish national CA.

The Finnish national CA, also known as the Certification Authority of Finland, is a trusted entity responsible for issuing digital certificates within Finland. These digital certificates are used in Public Key Infrastructure (PKI) to verify the authenticity of digital identities, such as websites, email addresses, and individuals or organizations. The Finnish national CA may establish policies and procedures for certificate issuance, renewal, and revocation, and ensure the security and integrity of the PKI infrastructure within Finland. The certificates issued by the Finnish national CA typically contain information, including a public key and other identifying details, that are used for authentication, encryption, and digital signature purposes, enhancing the security of online communications and transactions.

**HW13: Exercise 5.**

Find the meaning of the following terms and shortly explain their purpose in key establishment:

a) key wrap algorithm

A key wrap algorithm is a cryptographic algorithm used in key establishment to securely wrap (encrypt) a secret key using another key, often referred to as a wrapping key or key encryption key (KEK). The purpose of a key wrap algorithm is to protect the confidentiality and integrity of the secret key during its transportation or storage, ensuring that only authorized parties with the appropriate wrapping key can unwrap (decrypt) and use the secret key.

b) perfect forward secrecy

Perfect forward secrecy: Perfect forward secrecy, also known as forward secrecy, is a property in key establishment protocols where the compromise of long-term keys does not compromise the confidentiality of past or future communication. In other words, even if an attacker gains access to the long-term keys used in a key exchange, they should not be able to decrypt previously captured or future communications, as each key exchange generates a unique session key that is used only for that session. This enhances the security of the communication by minimizing the impact of key compromise, as it prevents the retroactive decryption of previously intercepted data even if the long-term keys are compromised later.

