Comp 3334 Computer System Security
Assignment 1
Zhu Jin Shun 22101071d
2025/3/1

Question 1:

Task 1:

(1) 10001101 ∧ 01110110 =00000100

∧-AND Operation

Bit Position	1st Operand	2nd Operand	Result Calculation	Result
1	1	0	1∧0=0	0
2	0	1	0∧1=0	0
3	0	1	0∧1=0	0
4	0	1	0∧1=0	0
5	1	0	1∧0=0	0
6	1	1	1∧1=0	1
7	0	1	0∧1=0	0
8	1	0	1∧0=0	0

(2) 10111001∨01100111 =11111111

∨-OR Operation

Bit Position	1st Operand	2nd Operand	Result Calculation	Result
1	1	0	1∨0=1	1
2	0	1	0∨1=1	1
3	1	1	1∨1=1	1
4	1	0	1∨0=1	1
5	1	0	1∨0=1	1
6	0	1	0∨1=1	1
7	0	1	0∨1=1	1
8	1	1	1∨1=1	1

(3) 11100011 ⊕ 01110111 =10010100

⊕-XOR Operation

Bit Position	1st Operand	2nd Operand	Result Calculation	Result
1	1	0	1⊕0=1	1
2	1	1	1⊕1=0	0
3	1	1	1⊕1=0	0
4	0	1	0⊕1=1	1
5	0	0	0⊕0=0	0
6	0	1	0⊕1=1	1
7	1	1	1⊕1=0	0

8	1	1	1⊕1=0	0

Task 2:

Confidentiality

Requirement: The personal information of users must be kept secure and only accessible to authorized entities. This means that sensitive data like identity card numbers and phone numbers should be protected from unauthorized access.

Example: The system should employ encryption methods to encrypt user data both at rest and in transit . This ensures that even if data is intercepted, it cannot be read without the appropriate decryption keys.

Integrity

Requirement: The personal information stored in the profile must remain accurate and unaltered by unauthorized users. Any changes to the data should be tracked and verified to prevent tampering.

Example: The system should implement checksums or hash functions to validate the integrity of user data. If a user's profile information is updated, the system can compare the hash of the new data with the previous hash to ensure that the data has not been altered in an unauthorized manner.

Availability

Requirement: The online identity system must be accessible to authorized users when needed. This means ensuring that the service remains operational and that users can retrieve their profiles without significant downtime.

Example: The system should employ redundancy and load balancing to handle high traffic and prevent downtime. For instance, using multiple servers and a cloud service can ensure that if one server fails, others can take over, maintaining access to user profiles.

Task 3:

The attack method I have selected will deliberately overflow the password buffer to overwrite the return address with the address of special_action().

If the system uses big-endian byte order, I'll input: 0000 0000 0000 0000 0000 0000 0000 0000 00000000 000025C2

If the system uses little-endian byte order, I'll input: 0000 0000 0000 0000 0000 0000 0000 0000 00000000 C2250000

This works because my input will fill the memory for password[16] with zeros, overwrite the EBP with zeros, and then replace the return address with the address of special_action() (which is 0x000025C2).

When the authorize() function returns, instead of going back to the normal execution flow where password_exam() would verify my password, the program will jump directly to special_action(). This completely bypasses the password verification process, giving me unauthorized access to the protected functionality.

Task 4:

Trust in Endpoint 2: No, the user does not trust Endpoint 2 because there is no direct trust path from CA 2 to Endpoint 2.

Trust in Endpoint 1: Yes, the user trusts Endpoint 1 because it is directly signed by CA 2, which the user trusts.

Question 2:

Task 1:

Authentication is the process of verifying the identity of a user, device, or entity, usually through credentials such as passwords, biometric data, or tokens.

Authorization occurs after authentication and determines whether the authenticated user has permission to access specific resources or perform certain actions based on assigned roles and permissions.

Task 2:

When Eve acts as a passive attacker, she is more likely to collect information from the message without altering the content or interacting with the other users. Her primary goal is to gather information for future exploitation. For example, if Alice sends a private message containing her personal details, such as her passport number and ID number, to Bob for visa purposes, Eve might intercept this message and sell it to another company for benefit. In this case, she uses the intercepted information for illegal benefits. Here, the confidentiality property of the CIA triad is compromised, as sensitive information is exposed to an unauthorized party.

When Eve is an active attacker, she is more likely to disrupt or destroy data and engage in direct interaction with the other users. For instance, if Alice sends a simple "hello" message to Bob, and Eve, acting as an active attacker, intercepts it, she might destroy the content of the message before it reaches Bob. In this scenario, the confidentiality, integrity, and availability property of the CIA triad is broken, as the accuracy and reliability of the original message have been altered, and Bob may not be able to receive the message.

Task 3:

Plaintext String:

P- 15 O- 14 L- 11 Y- 24 U- 20 C- 2 O- 14 M- 12 P- 15

Key: 7

En(x) = (x + key) mod 26

Therefore:

P=(15+7) mod 26= 22 mod 26 = 22 → W

O=(14+7) mod 26=21 mod 26 = 21 → V

L=(11+7) mod 26=18 mod 26 = 18 → S

Y=(24+7) mod 26=31 mod 26 = 5 → F

U=(20+7) mod 26=27 mod 26 = 1 → B

C=(2+7) mod 26=9 mod 26 = 9 → J

O=(14+7) mod 26=21 mod 26 = 21 → V

M=(12+7) mod 26=19 mod 26 = 19 → T

P=(15+7) mod 26= 22 mod 26 = 22 → W

So, the ciphertext by Caesar Cipher for plaintext string POLYUCOMP if the key is 7 is WVSFBJVTW.

Task 4:

Truly random numbers are generated from unpredictable physical processes and are not repeatable or predictable to ensure

unpredictability.

Pseudorandom numbers are generated by algorithms and are deterministic, meaning they can be reproduced if the same seed is used.

Question 3:

First Step:

Perform byte substitution for each input of the last round:

Substitution Table

0a→67	1b→af	5d→4c	2c→71
1a→a2	4f→84	e2→98	c7→c6
a9→d3	b3→6d	bc→65	d3→66
ca→74	ad→95	d9→35	c2→25

Second Step

Shift Rows

Row 0 remains unchanged.

Row 1 shift one position to the left.

Row 2 shifts two positions to the left.

Row 3 shifts three positions to the left.

Shifted Table:

67	af	4c	71

84	98	c6	a2
65	66	d3	6d
25	74	95	35

Third Step:

Add Round Key, perform XOR of the resulting matrix and the round key:

67⊕b6	af⊕8f	4c⊕48	71⊕23
84⊕fd	98⊕9f	c6⊕0a	a2⊕f6
65⊕cf	66⊕78	d3⊕6b	6d⊕12
25⊕6b	74⊕96	95⊕23	35⊕e9

67⊕b6=01100111⊕10110110=11010001=d1

af⊕8f=10101111⊕10001111=00100000=20

4c⊕48=01001100⊕01001000=00000100=04

71⊕23=01110001⊕00100011=01010010=52

84⊕fd=10000100⊕11111101=01111001=79

98⊕9f=10011000⊕10011111=00000111=07

c6⊕0a=11000110⊕00001010=11001100=cc

a2⊕f6=10100010⊕11110110=01010100=54

65⊕cf=01100101⊕11001111=10101010=aa

66⊕78=01100110⊕01111000=00011110=1e

d3⊕6b=11010011⊕01101011=10111000=b8

6d⊕12=01101101⊕00010010=01111111=7f

25⊕6b=00100101⊕01101011=01001110=4e

74⊕96=01110100⊕10010110=11100010=e2

95⊕23=10010101⊕00100011=10110110=b6

35⊕e9=00110101⊕11101001=11011100=dc

Therefore, the final resulting matrix after the last round of AES is:

d1	20	04	52
79	07	cc	54
aa	1e	b8	7f
4e	e2	b6	dc

Question 4:

Task 1:

For the hash function with an output of 200 bits, the expected possible hash values will be 2^{200} possible hash values. As there is a 50% chance of finding a collision, the number of possible trials would be about $2^{200/2} = 2^{100}$ to find the collision.

The hash function does not meet the collision resistance requirement because 2^{100} trials are lower than 2^{112} which means it does not meet the requirement of computational infeasibility This makes the function not secure enough against collision attacks.

Task 2:

The reason is given that the number of outputs generated by hash functions is significantly smaller than the number of possible inputs, the pigeonhole principle dictates that there must be pairs of distinct inputs that produce identical outputs which lead to collision existing in the hash function.

Task 3:

MD5 (Message Digest Algorithm 5) is deprecated due to significant security vulnerabilities, including its susceptibility to collision attacks, which allow attackers to find different inputs that produce the same hash. Additionally, with a 128-bit output, it lacks sufficient security margins compared to modern hash functions like SHA-256 and SHA-3, as MD5 would only require y $(2^{128})^{1/2} = 2^{64}$ trials to find a strong collision which is feasible for attackers.

Task 4:

Block size = 4 bytes

The current length of the message = 29 bytes.

The next multiple of 4 that is greater than 29 is 32 bytes (since 32=4×8).

Padding needed

= Total length needed - Current length

= 32−29

= 3 bytes.

Using the Zero Length method, we will need to append three zero bytes (0x00) to the end of the message. Thus, the final padded data consists of the original 29 bytes followed by three 0x00 bytes, ensuring the total length is now 32 bytes, aligned with the block size. Therefore, the padded message will be the original message adding 00 00 00 at the end.

Task 5:

m: message

k: secret key

First step: concatenate the secret key k with the message m

$$x=k||m$$

Second step: apply the SHA-3 hash function to the concatenated input x to produce the MAC.

$$MAC(m,k)=SHA3(x)=SHA3(k||m)$$

As the output of the SHA-3 computation is the MAC value.

The final resulting function is MAC(m,k)=SHA3(k||m)