Homework 1- Principles of Cybersecurity

Exercise 1

Cipher text: **dxjblsbo**

This cipher can be solved by trying different combinations of rotating the alphabet (aka Caesar cipher). Meaning if the cipher would have ROT1 “a” would become “b” and so on. For the current cipher text, the solution is ROT3.

Plain text: **gameover**

Exercise 2

A one-time pad is used to decrypt plain text by performing an XOR operation between the letters. Meaning that the one-time pad characters and the plain text characters are converted to binary which then has the XOR operation performed on them. The resulting text becomes the cipher.

A) Plain text “here” and Cipher text “kite”:

Plain Text (here)	010	000	110	000
Cipher (kite)	100	011	111	000
One time pad (key)	110	011	001	000

One time pad: “**rife**”

B) Plain text “file” and Cipher text “kite”:

Plain text (file)	001	011	101	000
Cipher (kite)	100	011	111	000
One time pad (key)	101	000	010	000

One time pad: “**lehe**”

Exercise 3

The question consists of two different types of result; expected time and worst-case time. The expected time is worst case divided by 2.

A)

$$\frac{2^{80}}{2^{30}} = 2^{50}\ seconds\ worst\ case \rightarrow \frac{2^{50}}{2} = 2^{49}\ seconds\ average\ case, 60 * 60 * 24 * 365$$
$$= 31536000\ seconds\ per\ year, \frac{2^{49}}{31536000} \approx\ 17\,851\,025.9\ years$$
$$\approx 1.79 * 10^{7} years$$

B)

$$\frac{2^{100}}{2^{30}} = 2^{70}\ seconds\ worst\ case \rightarrow \frac{2^{70}}{2} = 2^{69} seconds\ average\ case, \frac{2^{69}}{31536000}$$
$$\approx 1.87 * 10^{13} years$$

C)

$$\frac{2^{256}}{2^{30}} = 2^{226}\ seconds, \frac{2^{226}}{31536000} = 3.41 * 10^{60}\ years$$

Exercise 4

A)

As recommended when calculating and listing all the inputs and outputs of the "majority voting" the register Y will step close to around 75% of the time. The reason for this is because in each decision if Y will step is calculated using the majority.

$$m = maj(x, y, z)$$

If you list all different possibilities for the bits, the outcome will be:

(x, y, z) 000 001 010 011 100 101 110 111

In this situation register Y will step $\frac{6}{8} = 0.75 = 75\%$ of the time. Meaning that 75% of the time Y end up on the majority side (on average).

B)

Two registers will always step which means that on average one register till step 0% of the time.

C)

There are two possibilities where all registers will jump. When they all are 000 or 111.

$$\frac{6}{8} = 75\%\ of\ the\ time$$

D)

Using the bit calculations above on average the registers X and Y will step $\frac{4}{8} = 0.5 = 50\%$ of the time.

Exercise 5

Represent C with L0, R0, K0, K1, K2, and K3.

A)

	Left	**Right**
Round 0	L0	R0
Round 1	R0	L0
Round 2	L0	R0
Round 3	R0	L0

Round 4	L0	R0

Cipher=(L0, R0)

B)

	Left	**Right**
Round 0	L0	R0
Round 1	R0	L0⊕R0
Round 2	L0⊕R0	R0⊕(L0⊕R0)
Round 3	L0	L0⊕(L0⊕R0)
Round 4	R0	L0⊕R0

Cipher=(R0, L0⊕R0)

C)

	Left	**Right**
Round 0	L0	R0
Round 1	R0	L0⊕K0
Round 2	L0⊕K0	R0⊕K1
Round 3	R0⊕K1	K2⊕(L0⊕K0)
Round 4	K2⊕(L0⊕K0)	K3⊕(R0⊕K1)

Cipher=(K2⊕(L0⊕K0), K3⊕(R0⊕K1))

D)

	Left	**Right**
Round 0	L0	R0
Round 1	R0	L0⊕(R0⊕K0)
Round 2	L0⊕(R0⊕K0)	R0⊕(R1⊕K1)
Round 3	R0⊕(R1⊕K1)	(L0⊕(R0⊕K0))⊕(R2⊕K2)
Round 4	(L0⊕(R0⊕K0))⊕(R2⊕K2)	(R0⊕(R1⊕K1)) ⊕(R3⊕K3)

The answer above consists of R1,R2,R3. The actual answer is:

Left:

$(L0 \oplus (R0 \oplus K0)) \oplus ((R0 \oplus ((L0 \oplus (R0 \oplus K0)) \oplus K1)) \oplus K2)$

Right:

$(R0 \oplus ((\, L0 \oplus (R0 \oplus K0)) \oplus K1)) \oplus (((L0 \oplus (R0 \oplus K0)) \oplus ((R0 \oplus ((L0 \oplus (R0 \oplus K0)) \oplus K1)) \oplus K2)) \oplus K3)$

Cipher=$\Bigg(\big(L0 \oplus (R0 \oplus K0)\big) \oplus \Big(\Big(R0 \oplus \big(\big(L0 \oplus (R0 \oplus K0)\big) \oplus K1\big)\Big) \oplus K2\Big), \Big(R0 \oplus \big(\big(\, L0 \oplus (R0 \oplus K0)\big) \oplus K1\big)\Big) \oplus \Bigg(\Bigg(\big(L0 \oplus (R0 \oplus K0)\big) \oplus \Big(\Big(R0 \oplus \big(\big(L0 \oplus (R0 \oplus K0)\big) \oplus K1\big)\Big) \oplus K2\Big)\Bigg) \oplus K3\Bigg)\Bigg)$

Exercise 6

The main difference between ECB and CBC is that ECB encrypts each block **independently** while CBC **chains** the blocks together.

ECB works just as stated above. Data is divided into blocks which then with the help of a key are encrypted. A problem with this is that if two blocks are identical their ciphers will also be. For example, $P_i = P_j \rightarrow C_i = C_j$.

This means when trying to encrypt an image the original and the encrypted one will have a lot of similarities.

CBC works by starting with an initialization vector (IV). The encryption method then XORs the plaintext block with the IV and then encrypts it with the key. The next block will then take the ciphered text from block one and use it as an IV value. Doing this means that $P_i = P_j \rightarrow C_i \neq C_j$.

If an ECB encrypted block would have tampered with the rest of the encrypted data would not be harmed because each block is encrypted independently. But this also means that ECB is more open to attacks like “cut and paste”. Because CBC blocks are chained together if a block would be tampered with would result in that the rest of the data would also be affected when decoding the message.

A way for computers to detect if encrypted data has been tampered with is using the MAC address. For example, in CBC the MAC address of the receiving end is stored as the last part of the cipher text. If the MAC address is invalid than the encrypted message has been tampered with.

Exercise 7

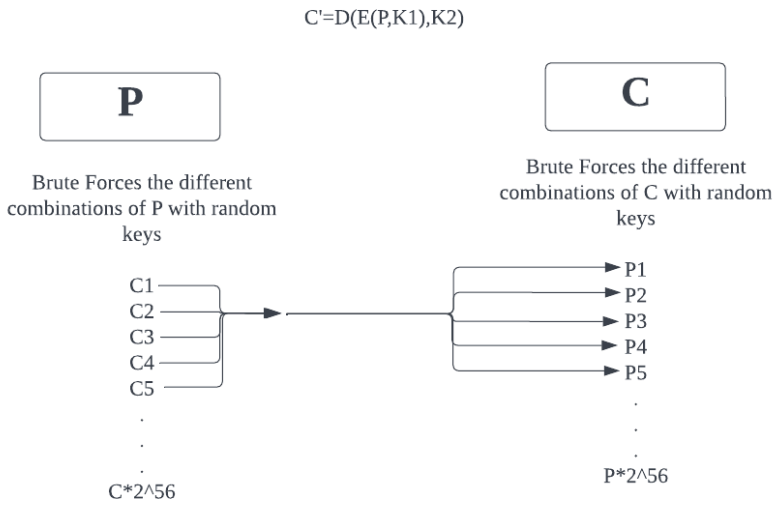
A)

DES encrypts blocks of data of 64 bits in size. The key is also 64 bits, but 8 bits are used as parity bits, meaning that the actual bits used of the DES key when encrypting is 56 bits.

B)

This type of DES (2-DES) is vulnerable against attacks like meet-in-the-middle. Even though the keys are 56 bits long the search space will not be $2^{(56+56)}$ but rather $2^{56} * 2 = 2^{57}$.

The attack uses fundamentality where it works from both sides of the function. Meaning that it in this case decrypts the cipher text and encrypts the plain text. The attacker will then try every single different combination of the key until it finds two different which matches up together.



In the figure above the different C are then compared to the different computed P. If any of these matches the two possible keys have been found.

C)

A way to improve the security of the cipher without adding another key is by adding the keys repeatedly to the encryption and decryption stage. By doing this it would take the attacker a lot more computations to decrypt the cipher.

For example:

$C = E(D(E(P, K1), K2), K1)$

Exercise 8

A)

Public key $(N, e) = (91,17)$

$N = p * q = 91 = 7 * 13$

$e * d = 1 mod (p-1)(q-1) \rightarrow 17 * d \; mod \; (p-1)(q-1) = 1 \rightarrow 17 * d \; mod 72 = 1 \rightarrow d = 17$

$\boldsymbol{Private\ key\ (d) = 17}$

B)

In the example above it was extremely easy to recover the private key. But in a more real-world example the p and q primes would be a much larger prime number. Different types of RSA encryption programs use different lengths of primes but some use 1024 bits. When a prime is that large it becomes near impossible to calculate the private key from trying to guess and find p and q.

C)

Encryption

$M = 19, N = 91, e = 17 \rightarrow Encryption: C = M^e \ mod \ N \rightarrow C = 19^{17} mod \ 91 = 80$

Decryption

$M = C^d \ mod \ N \rightarrow 80^{17} mod \ 91 = 19$

D)

$S = M^d mod(n) \rightarrow S = 25^{17} mod 91 = 51$

The process works as follow: Alice signs the message using her private key. Bob can than with the help of Alice public key and the signed message and the actual message verify that it was Alice who signed the message.

$M = S^e mod(n) = 51^{17} mod(91) = 25$

Exercise 9

A)

Double encryption does not increase the security compared to a single RSA encryption.

B)

The first step an attacker would take when trying to decrypt and find the private key is factoring N (finding p and q). The next step would be to compute the two different "d". Computing both will not take a lot more time than just computing one of them. This is because p and q have already been found. Meaning that having the same N but different but two different exponents doesn't help towards security.

Exercise 10

A)

The modified Diffie-Hellman method is secure from a man-in-the-middle attack.

B)

The reason why a normal Diffie-Hellman key exchange is not secure for a man-in-the-middle attack is that the primary use case for this type of key exchange is that Alice and Bob without any knowledge or share information between them should be able to encrypt information on a public channel. When both Alice and Bob have their initial conversation where they choose their own private keys (A1, B1) the attack could swap these values for another value that only the attacker knows (A2, B2). By doing this the attacker is able to decrypt all the future messages that are being sent between Alice and Bob.

But with the modified version of Diffie-Hellman where Alice and Bob have shared a 4-digit PIN number outside the public channel makes it so that it becomes impossible for the attacker to decrypt the initial message and future messages. The only way for the attacker to perform a man-in-the-middle attack is by figuring out the shared 4-digit PIN number. Meaning that using this modified Diffie-Hellman key exchange does make the it more secure against a man-in-the-middle attack.

But doing this kind of defeats the purpose of the Diffie-Hellman key exchange. The purpose of the key exchange is to establish a shared secret key over a public channel without sharing anything private.

Exercise 11

Hashing is the process of transforming information into a certain type of key with a fixed length using a hash function. Hashing is most commonly used to validate authentication for example passwords. By using hashing the provider can avoid storing the plaintext passwords in a database. The hash functions are one-way meaning that it is impossible to retrieve the original information from the hash key.

The only bad thing is that if a value would be hashed it will give a unique value for the original value. But that also means that if users have the same password it would lead to it giving the same hash key. A way to solve this would be to use salt where the purpose is to add random data to the original information before it is hashed.

Encryption is the process of transforming a piece of plain text information into a non-readable message using some sort of encryption key. For someone to either encrypt or decrypt a message the user needs to know what the encryption algorithm is and the key to decrypt it. There are different types of encryption protocols for example public key system (RSA) or DES. All these systems use different methods for how keys are exchanged between the sender and receiver of the information.

Meaning that the **biggest difference** between them is that hashing is a one-way function to hide information while encryption is a two-way meaning that the encrypted data can be decrypted again. Generally speaking, hashing is more secure than encryption due to it being a one-way function where it is impossible to “decrypt” a hashed value. Hashing is also important because it provides a fixed-length hash value which makes it perfect for storing values in a database for example passwords where encrypting passwords would not.