Md. Safiqul Islam
School of Information and Communication Technology
KTH-ID: 851221-A619
islam3@kth.se

1. Alice has designed a pseudo-random number generator using the following linear congruential generator algorithm:
$R_{i+1} = (69 \times R_i + 113)$ *mod* 256
She uses the output from the algorithm to encrypt a message by a bitwise XOR operation of the message and the output of the generator. Assume that Alice wants transmit the ASCII string "Safe" as plaintext message to Bob, and uses Ro = 43 as the initial value for the generator.

a) What is the resulting ciphertext? (in hexadecimal, "Safe" corresponds to the sequence 53_{16} 41_{16} 46_{16} 45_{16})

Answer:

Given, $R_0 = 43 = 00101011$
$R_1 = (69 * 43 + 113) \bmod 256 = 8 = 00001000$
$R_2 = (69 * 8 + 113) \bmod 256 = 153 = 10011001$
$R_3 = (69 * 153 + 113) \bmod 256 = 174 = 10101110$

Key value will be = $R_0\ R_1\ R_2\ R_3$ = 00101011 00001000 10011001 10101110
Plain text in binary = 01010011 01000001 01000110 01000101

Cipher text = Plain text $\oplus$ Key
= 01111000 01001001 11011111 11101011
= 78_{16} 49_{16} DF_{16} EB_{16}

b) This is symmetric encryption using a stream cipher. What is the key Alice is using for encryption?

Answer: Alice is using the key, $R_0 = 43$.

c) Assume that Alice wants to use the stream cipher to authenticate Bob, in the following way:
Alice encrypts a random number, a "nonce", with the secret key and transmits it to Bob. Bob decrypts the nonce and sends it back in clear text to Alice. Since Bob was able to decrypt the nonce, Bob is authenticated (since only Bob has the correct secret key to decrypt the nonce). What is the problem with this scenario?

Answers:

A man in the middle or bad guy can easily get the information and pretend that he is Bob or Alice.

2. Describe briefly five different block encryption algorithms besides DES. Write down the main characteristics of the algorithms and compare them (in a table, for instance). Summarize how the algorithms are used, and what their strengths and weaknesses are.

Answer:

The table below shows the characteristics and strength and weaknesses of five different block encryptions:

	Algorithm	**Strength**	**Weakness**
Electronic Code Book (ECB)	It takes the message as input and break it into 64 bit block and encrypt each block with same secret key.	- Easy encryption method. - Less expensive.	- It produce the same cipher blocks for each repeated plain text blocks. - Man in the middle can easily gain information by looking at the consecutive repeated cipher block and rearrange and modify it.
Cipher Block Chaining (CBC)	Break the message into 64 bit block and use initial random number (IV) and XOR'ed with first plaintext block and encrypt it with secret key to get C1. And this C1 is used as IV for the next 2^{nd} plain block. And rest of the process same as previous.	- No identical cipher blocks for the the repeated plain blocks.	-If any plain text block m2 garbled that will garbled cipher block C2…..Cn. -If Any cipher block garbled/(c2) that will garbled relevant Plaintext block(m2) and next plain text block(m3).

K-Bit Output Feedback Mode (OFB)	It generate one time pad (IV) and encrypt with Key and XOR'ed with message m1 to encrypt and encrypted k bits (before the XOR operation) is used with IV for the next block encryption and so on.	- Any arbitrary size of the plaintext can be encrypted. - One time pad can be generated in advance.	-If plaintext and cipher text is known by bad guy, the plain text can be modified to any desired message.
k-Bit Cipher Feedback mode (CFB)	It generate one time pad (IV) and encrypt with Key and XOR'ed with message m1 to encrypt and encrypted k bits from cipher block is used with IV for the next block encryption and so on.	- Any arbitrary size of the plaintext can be encrypted.	- Every byte of input require the DES operation.
Counter Mode (CTR)	It generate one time pad (IV) and encrypt with Key and XOR'ed with message m1 to encrypt and for the next block encryption it increments the IV and so on.	-One time pad can be generated in advance. - Decryption of message can be done from any point like CBC.	-Message can be altered if plain text and cipher text is known.

3. In a chained block cipher such as CBC, the encryption of one block is a function of decryption of previous blocks. Still, it is a desirable property of a chained block cipher that it is *self-synchronizing*, so that the entire message is not destroyed by a single error. In other words, if there is an error during the transmission of a block, after a few corrupted blocks the decryption will work correctly again. For instance, in Fig. 4-5 and 4-6 in Kaufman, if there is an error in transmission of block $c2$, clearly $m2$ will be incorrect

after decryption. However, it should still be possible to decrypt some of the following blocks *m*3 … *m*6 correctly.

a) If block *c*2 is corrupted between encryption and decryption, which blocks in the decrypted plain text will be affected by the error?

Answer:

If block C_2 is corrupted between encryption and decryption, Plain text block m_2 and m_3 will be affected by the error. As C_2 is decrypted and doing XOR with C_1 we get M_2 and C_2 is used as a input to XOR with decrypted C_3 to retrieve M_3. So, M_2 will be totally corrupted and partial of M_3 will be corrupted.

b) Suppose instead that the error occurs in the encryption logic, so that *m*2 is corrupted before it is encrypted. Which blocks in the decrypted plaintext will be affected by the error?

Answer:

If M_2 is corrupted then after encryption c_2, c_3, …c_n will be corrupted. Since, c_2, c_3, …c_n is decrypted to retrieve m_2,m_3,……m_n. So, decrypted plain text m_2,m_3,……$m_{n<}$ will be affected by the error.

4. A secure hash function gives a condensed version of a message (it is a "lossy" compression function).

a) What are the most important properties of a secure hash function for message authentication?

Answer:

There are two important properties of a secure hash function :

i. Collision resistance: It is impossible to find two messages with the same digest.
ii. One way function: Attacker will not be able to compute the input from the digest.

b) What is the problem with computing a message digest by hashing the message concatenated with a cryptographic key?

Answer:

There will be a problem if the secret key is concatenated at the first of the message. Because message digest algorithm work by dividing the messages into n bit chunks. Then calculate the hash for the first chunk and then use this first chunk to calculate the hash for

the next chunk. And, the result of the final chunk is the message digest. So, if anyone wants to add some message with the Message then he will use the digest and calculate hash for his appended message. Then he can send the appended message with newly computed digest. Here, Key added in the first is not providing the proper security. But, If key is added at the last then no one will be able to append the message.

c) Explain why HMAC is considered secure, and under what conditions it is considered secure?

Answer:

HMAC is secure because it eliminates the problem of adding key at the first. It added the key with the data then digest it and then add the key with the result and digest it again. So, no one can easily append the key in this scenario.

It is considered to be secure if the Message digest compression function is secure.

d) What does it typically mean when cryptologists say that a hash algorithm is "insecure"?

Answer:

If anyone can modify the message by modifying the computed digest, then the cryptologists say that a hash algorithm is "insecure".