CS2107 Tutorial 2 (Encryption: One-Time Pad & Block Ciphers)

School of Computing, NUS

30 August – 3 September 2021

1. *Attackable OTP? (Mid-Term Quiz S1 AY2019/20):*
   Bob really likes the One-Time Pad (OTP), an encryption scheme that does achieve perfect security. Bob thinks that he should be able to use the OTP by itself for a *secure message communication*, and not just for preserving confidentiality.

   Suppose Bob's OTP keys are random and always fresh as required. His plaintexts, however, always start with "`From: Bob`" string, and this is known by Mallory. Mallory is a man-in-the-middle, who can intercept Bob's ciphertexts, modify them, and then relay the modified ciphertexts to the respective receivers.

   Suppose now Mallory wants to modify all Bob's OTP ciphertexts so that, when decrypted by their respective receivers using correct keys, the recovered plaintexts start with "`From: Bot`" instead. What should Mallory *turn each OTP ciphertext from Bob into*? Explain briefly why your attack works.

   (Note: Suppose the two relevant characters are encoded using their following ASCII-based binary strings: '`b`' $\rightarrow$ 0110 0010, '`t`' $\rightarrow$ 0111 0100.)

Solution

Let's consider that the ciphertext (c) and key (k) byte strings below are zero based. To launch her attack, Mallory can just replace the 8-th byte of Bob's OTP ciphertexts, $c[8]$, with $c'[8] = c[8] \oplus b \oplus t$. In other words, Mallory can just XOR $c[8]$ with: $b \oplus t = 01100010 \oplus 01110100 = 00010110$.
To see how this attack works, let's consider the decryption done by a receiver. Notice that the receiver of an authentic ciphertext c from Bob will perform the following using its key k: $c[8] \oplus k[8] = b$. As such, $k[8] = c[8] \oplus b$. Therefore, when the receiver decrypts c' containing the modified $c'[8]$, the recovered $m'[8]$ is: $c'[8] \oplus k[8] = (c[8] \oplus b \oplus t) \oplus (c[8] \oplus b) = t$ as intended.

Alternatively, notice that XOR-ing a bit with 1 does flip the bit, i.e. $1 \rightarrow 0$ and $0 \rightarrow 1$. Performing $c'[8] = c[8] \oplus 000\mathbf{1}0\mathbf{11}0$ above is thus equivalent to flipping the 3 bits of $c[8]$ at index 1, 2 and 4 (with respect to the least significant bit). When a particular bit of an OTP ciphertext gets flipped, the corresponding decrypted bit will get flip as well. Hence, you can also see that, by performing the 3-bit flipping, the '`b`' (011**0** 0**01**0) will become '`t`' (011**1** 0**10**0) in the recovered plaintext, i.e. $m'[8] = t$.

2. *Block Cipher with a Small Block Size:*
Bob is designing a block cipher that performs complex operations similar to those in AES. He believes that he can combine the strengths of both block cipher (e.g. high confusion and diffusion) with that of stream cipher (e.g. lower latency) if he makes the block size rather small. Hence, he sets the size of the input and output blocks of his cipher to **16 bits** only. Alice, however, warns Bob that his block cipher can be attacked due to its small block size.

   (a) Consider a *known-plaintext attack* scenario, where an attacker can learn a number of plaintext and ciphertext pairs encrypted using the same key. Suppose the attacker wants to implement a *codebook attack* on Bob's Cipher, which is a block cipher with a small block size, by compiling a lookup table of all plaintext-ciphertext pairs observed under the same key. How much storage will the attacker need to comprehensively store *all the input and output blocks* in his table? Express your answer in MB (megabyte) or GB (gigabyte).
   Note: 1MB = 2^{20}, 1GB = 2^{30}.

   Solution

   Storing 1 input block requires 16 bits = 2 bytes.
   Storing 1 output block requires 16 bits = 2 bytes.
   Hence, storing 1 pair of input and output blocks require 4 bytes.
   Storing all the pairs therefore requires:
   $2^{16}.4$ bytes $= 2^{16}.2^2/2^{20}$ MB $= 2^{-2}$ MB $= 0.25$ MB.

   (b) Given Alice's warning and posssible codebook attack, Bob agrees to increase the size of the input and output blocks of his cipher to **48 bits**. Using the same codebook attack, how much storage will the attacker now need to store *all the input and output blocks* in his lookup table? Express your answer in MB (megabyte), GB (gigabyte), TB (terabyte), or PB (petabyte).
   Note: 1T = 2^{40}, 1P = 2^{50}.

   Solution

   Storing 1 input block requires 48 bits = 6 bytes.
   Storing 1 output block requires 48 bits = 6 bytes.
   Hence, storing 1 pair of input and output blocks require 12 bytes.
   Storing all the pairs now requires:
   $2^{48}.12$ bytes $= 2^{48}.2^2.3$ bytes $= 2^{48}.2^2.3/2^{50}$ PB $= 3$ PB.

3. *Mode-of-Operation (Mid-Term Quiz S1 AY2018/19):*
Cipher Block Chaining (CBC) mode-of-operation is commonly used to encrypt a plaintext longer than a cipher's block. In CBC, each plaintext block is XOR-ed with the previous ciphertext block before being encrypted. An IV is used in encrypting the first plaintext block.

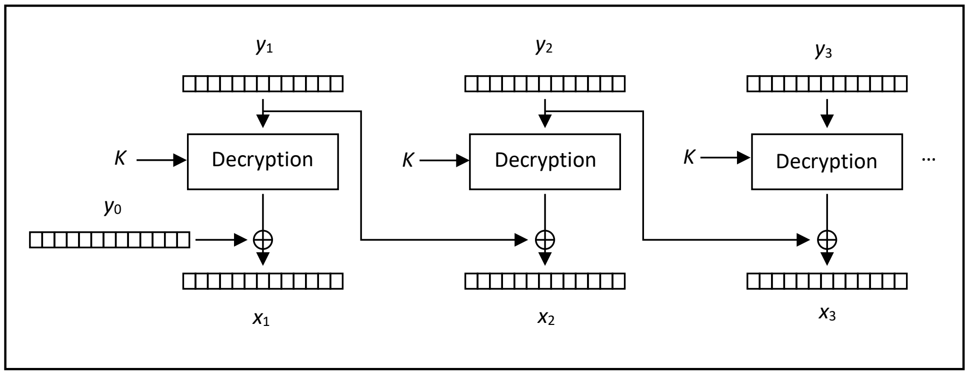
Mathematically, the encryption can thus be expressed as follows:
Given a n-block plaintext message $x_1, x_2, x_3, \ldots, x_n$, a secret key K, and an initial value IV, CBC outputs $(n{+}1)$-block ciphertext message $y_0, y_1, y_2, \ldots, y_n$, where:

- $y_0 = IV$;
- $y_k = Enc_K(x_k \oplus y_{k-1})$, for $k = 1, 2, 3, \ldots, n$.

Given the definition above, answer the following questions:

- Your lecture notes show a diagram depicting how a CBC-based encryption is done. Draw a diagram of the corresponding CBC-based *decryption*.

  Solution:



- How is decryption affected if the first ciphertext block y_0 is *removed* from the ciphertext?

Solution

The plaintext block x_1 cannot be correctly recovered.
Other plaintext blocks x_2, x_3, $\ldots$, x_n, however, still can be recovered correctly.

- Can the encryption processes of different blocks belonging to a plaintext run *in parallel*? How about the decryption of a ciphertext's different blocks?

Solution

The encryption process *cannot* run in parallel.
This is since the encryption at round i to produce the ciphertext block y_i does take as its input the ciphertext block y_{i-1} that is generated only in the previous round i-1.

The decryption process *can* run in parallel.
This is because the decryption at round i to recover the plaintext block x_i depends only on the ciphertext blocks y_i and y_{i-1}, which are both readily available from the sent ciphertext.

4. *Insecure Use of DES (Mid-Term Quiz S1 AY2018/19):*

(a) Bob knows that DES has a rather short key size/length of 56 bits. He, however, still wants to employ DES due to its widespread availability. Bob thinks that he has found a good way of addressing the limited key length of DES by randomly selecting three different keys K_1, K_2 and K_3. Bob then performs his DES encryption as follows:

$$C = E_{K_1 \oplus K_2 \oplus K_3}(P).$$

Decryption process is then performed using $K_1 \oplus K_2 \oplus K_3$ as its key. Bob argues that his method significantly increases the key space size. Is Bob's argument correct? Argue concisely by comparing the key space size of using one and three keys above.

Solution

Bob's argument is incorrect. The new key $K_1 \oplus K_2 \oplus K_3$ has the same length as those of K_1 and K_2, namely 56 bits. This is since the XOR operation of three 56-bit strings is also a 56-bit string. Hence, the key space size of Bob's new method remains 2^{56}.

(b) Bob now uses only two secret keys K_1 and K_2. However, he modifies his encryption to implement 2DES as follows:

$$C = E_{K_2}(E_{K_1}(P)).$$

Bob now believes that his double-encryption method indeed *doubles* the key space size to $2^{2.56} = 2^{112}$, and brute-forcing correspondingly requires 2^{112} cryptographic operations. How can you tell Bob that, under the **known-plaintext attack**, there is a way to find his two keys by performing $2.2^{56} = 2^{56+1} = 2^{57}$ cryptographic operations only?

Solution

Given a pair of plaintext P_1 and its corresponding ciphertext C_1, an attacker can perform the following steps:

i. Decrypt the ciphertext C_1 using all 2^{56} possible key values, and store all the recovered plaintexts in set V.

ii. Encrypt the plaintext P_1 using all 2^{56} possible key values, and store all the recovered plaintexts in set U.

iii. For a match (common element) x in the two sets, we can obtain the two keys as follows. The key that is used to generate x in Step ii is the key K_1 used by Bob. The key that is used to generate x in Step i is the key K_2 used by Bob.

The total no of cryptographic operations (i.e. encryptions and decryptions) in the steps above is $2.2^{56} = 2^{56+1} = 2^{57}$, and not 2^{112}. This attack technique is also known as the *meet-in-the-middle attack*, which can be mounted on ciphers that perform multiple encryption operations in sequence. Notice that the extra level/round of encryption has given only 1 extra bit of security (not 56).

Note that, in Step iii, it is possible to find i multiple entries $x_1, x_2, \ldots, x_i$, that exist in both sets. Correspondingly, we do have i possible pairs of K_1 and K_2. In this case, the attacker just needs to test all these key-pair candidates using other available pairs of plaintext and ciphertext until there is only one unique applicable pair of K_1 and K_2.

5. *3DES Encryption Options:*
Your lecture notes have mentioned two 3DES encryption options, namely:

- $E_{k_1}(E_{k_2}(E_{k_1}(x)))$; and
- $E_{k_1}(D_{k_2}(E_{k_1}(x)))$.

The latter is quite popular due to its extra benefit. It can provide a backward compatibility with the (single) DES. Explain succinctly how one can use 3DES to be compatible with, or simulate, DES.

Solution

Simply set $k_2 = k_1$, i.e., use one key k_1 only.
The output of $E_{k_1}(D_{k_2}(E_{k_1}(x)))$ thus becomes $E_{k_1}(D_{k_1}(E_{k_1}(x))) = E_{k_1}(x)$ as in DES.

— End of Tutorial —