

Symmetric Encryption

Ahmed Y. Banihammd & Ihsan, ALTUNDAG

Mon November 5, 2007

Advanced Cryptography

1st Semester 2007-2008

University Joseph Fourier, Verimag

Master Of Information Security And Coding

Contents

1	Introduction	2
2	Block cipher modes	3
2.1	Electronic Code Book (ECB)	3
2.2	Cipher Block Chaining (CBC)	3
2.3	Cipher Feedback Mode (CFB)	4
2.4	Output Feedback Mode (OFB)	5
3	Attack on ECB mode	6
4	Hybrid Encryption	6
4.1	Security of Hybrid Encryption	7
5	Optimal Asymmetric Encryption Padding (OAEP)	7
5.1	The setting of the OAEP	7
5.2	Encryption	8
5.3	Decryption	8
6	Logical Attacks	8
7	Needham Schroeder	9
8	Conclusion	11
9	Reference	12

1 Introduction

This report is written as a summarization for the Symmetric Encryption lecture. Each section will show a general description with some details. The following subjects will be provided:

1. Block cipher modes
 - (a) ECB
 - (b) CBC
 - (c) CFB
 - (d) OFB
2. Attack on ECB
3. Hybrid Encryption
4. OAEP
5. Logical Attacks
6. Needham Schroeder
7. Conclusion

2 Block cipher modes

2.1 Electronic Code Book (ECB)

In ECB mode, the message is divided into blocks of size n and each block is encrypted separately. The following figure represents the encryption and the decryption:

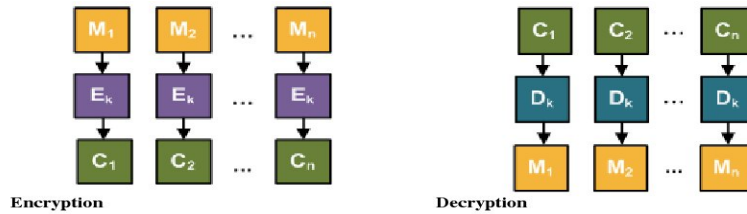


Figure 1: Illustrates the encryption and decryption of ECB mode

- What is useful about ECB?
 - Encryption in parallel.
 - Has the same speed as block cipher.
- Drawback
 - The main problems with this mode is that it has poor security, that the same input block is always encrypted as the same ciphertext and that an attacker can substitute blocks to alter part of a message (substituting ciphertext blocks from previous messages that use the same key).
 - An error in the cipher text will affect one complete block of plaintext.
 - Also, the inputs to the block cipher are never randomized because they are always exactly equal to the corresponding block of plaintext.

2.2 Cipher Block Chaining (CBC)

CBC mode divides the message into blocks of size n . The process starts with XORing the first block with random initial vector, then, encrypt the result and use it to XOR with the next block. The following figure represents the encryption and decryption using CBC:

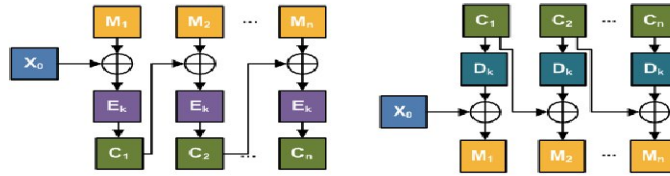


Figure 2: Illustrates the encryption and decryption of CBC mode

- What is useful about CBC?
 - It is clearly obvious that the plaintext is concealed using the XOR operation with the previous block.
 - Has the same speed as block cipher.
- Drawback
 - This mode of cipher does not provide a parallel decryption. One method to solve this problem is called “ciphertext stealing”. In this technique, we pass the last cipher text block through the cipher in ECB mode and XOR the output of that against the remaining message bytes. This avoids lengthening the message (beyond adding the Initial Vector).

2.3 Cipher Feedback Mode (CFB)

CFB mode is a way to turn a block cipher into a stream cipher. The description of the encryption algorithm is as follows. Encryption occurs by XOR’ing the keystream bytes with the plaintext bytes. The keystream is produced one block at a time, and it is always dependent on the previous keystream block as well as the plaintext data XOR’d with the previous keystream block. The following figure represents the encryption and decryption using CFB:

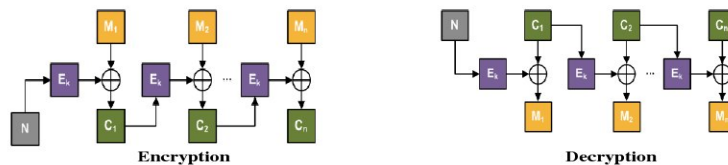


Figure 3: Illustrates the encryption and decryption of CFB mode

- What is useful about CFB?
 - If we use a 64-bit block ciphers, there is no nonce-size problems.

- Concealed plaintext.
 - Has the same speed as block cipher.
 - Random input to the block cipher.
- Drawback
 - Encryption is not parallel.
 - Integrity check is in need due to the fact that bit-flipping attacks can be easily occur. An error in the plaintext will effect not only the corresponding bit, but also the complete block.

2.4 Output Feedback Mode (OFB)

OFB mode turns a block cipher into a stream cipher. The encryption algorithm is done with the initialization vector by producing keystream, and combining the keystream with the plaintext via XOR. OFB produces keystream one block at a time. Here, each block of keystream is produced by encrypting the previous block of keystream, except for the first block, which is generated by encrypting the nonce. For the decryption the cipher text is XORed with the key stream to produce the plaintext, and the decipher operation is of course depends on the previous ones.

Note it is required that in every time a new IV must be use, otherwise the cipher stream of the any two messages that uses the same IV will be the same.

The following figure represents the encryption and decryption using OFB mode:

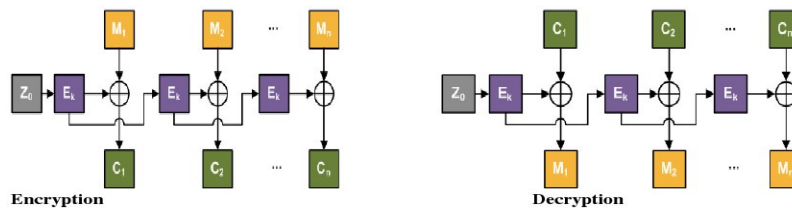


Figure 4: Illustrates the encryption and decryption of OFB mode

- What is useful about OFB?
 - If we use a 64-bit block cipher, there is not nonce-size problems.
 - Random ininput to the block cipher.
 - An error in the plaintext will affect only the specific bit and not the complete block.
 - Has the same speed as block cipher.
 - Keystreams can be precomputed.

- Conceal plaintext.
- Drawback
 - No parallel computation of the keystream.
 - Integrity check is in need due to the fact that bit-flipping attacks can be easily occur.
 - The same key can not be used, and it is better to avoid reusing a single key across multiple message or data streams to avoid problems.

3 Attack on ECB mode

During this section i will give the general idea of the attack for the reason that the details proof is provided in the lecture slides. The attack is called computational attack. During this attack we suppose a fixed block cipher size.

$$\mathcal{E} : \mathcal{K} \times \{0, 1\}^n \rightarrow \{0, 1\}^n$$

where n is the the size of the block.

The following adversary is build:

$$\mathcal{E}_K(LR(m_l, m_r, b)) = \begin{cases} \mathcal{E}_K(m_l) & \text{if } b = 1 \\ \mathcal{E}_K(m_r) & \text{if } b = 0 \end{cases}$$

Now, it is all depends on value of b. However, by the following adversary we will be able to detect the value of b.

Adversary $A^{\mathcal{E}_K(LR(\dots, b))}$
 $M_0 \leftarrow 0^n || 1^n;$
 $M_1 \leftarrow 0^{2n};$
 $C[1]C[2] \leftarrow \mathcal{E}_K(LR(M_0, M_1, b))$
 If $C[1] = C[2]$ then return 1 else return 0

As a conclusion, ECB encryption is NOT secure.

4 Hybrid Encryption

Hybrid encryption is an encryption method which holds a specific characteristic by combining two or more encryption schemes. It include both symmetric and asymmetric encryption to take advantage of the stringths of each type. The following figure represent the encryption and decryption scheme:

Encryption	Decryption
<p>Algorithm $\overline{\mathcal{E}}_{pk}(M)$ $K \leftarrow K^s$; $C^s \leftarrow \mathcal{E}_K^s(M)$; $C^a \leftarrow \mathcal{E}_{pk}^a(K)$; $C \leftarrow (C^a, C^s)$; Return C</p>	<p>Algorithm $\overline{\mathcal{D}}_{sk}(C)$ Parse C as (C^a, C^s); $K \leftarrow \mathcal{D}_{sk}^a(C^a)$; $M \leftarrow \mathcal{D}_K^s(C^s)$; Return M</p>

4.1 Security of Hybrid Encryption

IND-CCA KEM + IND-CCA SKE



IND-CCA Hybrid Encrypting

Note: Check the lecture slides to see the detailed proof.

5 Optimal Asymmetric Encryption Padding (OAEP)

General description of the OAEP scheme Before applying an encryption algorithm such as RSA, preprocessing of the message is necessary in order to prevent known attacks (in order also to break some mathematical structures inherent to the asymmetric cipher, such as the multiplicative structure in the case of RSA, and which could be useful for the attacker). The message is divided into blocks and then some formatting and/or padding mechanisms are performed. The OAEP mechanism can be used with any bijective trapdoor function.

5.1 The setting of the OAEP

- A trapdoor function $f : D \rightarrow D$ with $D \subset \{0, 1\}^n$ for a given n ,
- A deterministic pseudorandom bit generator $G : \{0, 1\}^k \rightarrow \{0, 1\}^l$ with $n = k + l$,
- A (cryptographic) hash function $h : \{0, 1\}^l \rightarrow \{0, 1\}^k$ (assuming in general that $k < l$).

Given a random seed $s \in \{0, 1\}^k$, G generates a pseudorandom bit sequence of length l .

5.2 Encryption

Encryption : To encrypt a message $m \in \{0, 1\}^l$, we proceed in three steps :

- (1) We choose a random bit string $r \in \{0, 1\}^k$,
- (2) We set $x = (m \oplus G(r)) || (r \oplus h(m \oplus G(r)))$. If $x \notin D$ we return to step 1.
- (3) We compute $c = f(x)$.

5.3 Decryption

Decryption : To decrypt a ciphertext c , we use the function f^{-1} , the same deterministic pseudorandom bit generator G and the same h as above :

- (1) Compute $f^{-1}(c) = a || b$ with $|a| = l$ and $|b| = k$.
- (2) Set $r = h(a) \oplus b$ and get $m = a \oplus G(r)$.

6 Logical Attacks

- **Man in the middle Attack:** a man-in-the-middle attack (MITM) is an attack in which an attacker is able to read, insert and modify at will, messages between two parties without either party knowing that the link between them has been compromised. The attacker must be able to observe and intercept messages going between the two victims. The MITM attack can work against public-key cryptography
- **Replay Attack:** a replay attack is a form of network attack in which a valid data transmission is maliciously or fraudulently repeated or delayed. This is carried out either by the originator or by an adversary who intercepts the data and retransmits it, possibly as part of a masquerade attack by IP packet substitution (such as stream cipher attack).

Example

Suppose Alice wants to prove her identity to Bob. Bob requests her password as proof of identity, which Alice dutifully provides (possibly after some transformation like a hash function); meanwhile, Eve is eavesdropping the conversation and keeps the password. After the interchange is over, Eve connects to Bob posing as Alice; when asked for a proof of identity, Eve sends Alice's password read from the last session, which Bob must accept.

- **Reflection Attack:** a reflection attack is a method of attacking a challenge-response authentication system that uses the same protocol in both directions. That is, the same challenge-response protocol is used by each side to authenticate the other side. The essential idea of the attack is to trick the target into providing the answer to its own challenge.

The general attack outline is as follows:

1. The attacker initiates a connection to a target.
2. The target attempts to authenticate the attacker by sending it a challenge.
3. The attacker opens another connection to the target, and sends the target this challenge as its own.
4. The target responds to the challenge.
5. The attacker sends that response back to the target on the original connection.

If the authentication protocol is not carefully designed, the target will accept that response as valid, thereby leaving the attacker with one fully-authenticated channel connection (the other one is simply abandoned).

- **Oracle attack:** take advantage of normal protocol responses as encryption and decryption “services”.
- **Type flaw (confusion) attack:** substitute a different type of message field (e.g. a key vs. a name).

7 Needham Schroeder

Needham-Schroeder Public Key Protocol

The Needham-Schroeder public key protocol is another example that has been widely examined by protocol researchers. Users A and B own public keys, K_a and K_b respectively, which, it is assumed here, are authentically known by the other. In the following, N_a and N_b are random values chosen by A and B respectively, while $\{X\}_k$ denotes encryption of the value X with the public key K.

1. $A \rightarrow B: \{N_a, A\}_{K_b}$
2. $B \rightarrow A: \{N_a, N_b\}_{K_a}$
3. $A \rightarrow B: \{N_b\}_{K_b}$

Although this protocol is nearly 20 years old it has aroused quite some interest recently. An attack of Lowe shows that B cannot be sure that the final message came from A. Gollman points out that the protocol fails, because of this, to achieve his goal Gol2. Notice that A has never explicitly declared her intention to converse with B.

In order to fix the protocol against his attack, Lowe proposed the following variation which simply includes the identifier of B in the second message.

1. $A \rightarrow B: \{Na, A\}K_b$
2. $B \rightarrow A: \{Na, Nb, B\}K_a$
3. $A \rightarrow B: \{Nb\}K_b$

Let us consider whether the extensional goals for entity authentication proposed above are satisfied. Consider the point of view of A . When she receives the message 2 she wants to use it to verify that B wishes to communicate with her. Immediately we run into a problem here because it is not clear what she can tell from receiving an encrypted message. What she really requires is an authenticated message, but encryption with her public key may not provide this. Indeed, encryption is a way to provide the confidentiality service to the message sender. It seems that the protocol designers (and most analysers) have assumed that inclusion of the nonce Na , which A sent confidentially to B , is sufficient to provide authentication. Unfortunately this need not be the case, even with the most well known public key encryption algorithms.

To see the point, consider the Blum-Goldwasser public key encryption algorithm. In this algorithm the plaintext is added modulo 2 to a string formed by iterative squaring. Consequently it is trivial for an attacker to change the known parts of the ciphertext, which are the identifiers in message 1 (and message 2 in Lowe's fix). This results in a simple attack on the protocol, or on Lowe's fix.

In practice, use of the Blum-Goldwasser algorithm is not very reasonable, since it is known to be vulnerable to a chosen ciphertext attack which is eminently possible in the protocol. On the other hand, there is a reasonable scenario in which use of the well-known RSA algorithm is insecure. Suppose the attacker knows (or chooses) an identifier C such that C is the same bit string as identifier A shifted left one place. Then the attacker can capture $\{Na, A\}K_b$ and multiply it by $\{2\}K_b$. The effect of this is to change the encrypted message by shifting it to the left, due to the well-known multiplicative property of RSA, so that it becomes $\{2^*(Na, A)\}K_b = \{2^*Na, C\}K_b$. By this process, A 's name changes into C , even though the attacker has no knowledge of Na . With the collaboration of C (or we may assume the attacker is C) this allows a run of the protocol, or Lowe's fix, where B only ever wanted to communicate with C , but A believes B wants to communicate with A . An attacking run on Lowe's fix is as follows.

1. $A \rightarrow C_b: \{Na, A\}K_b$
 $C_b \rightarrow B: \{2^*Na, C\}K_b$
2. $B \rightarrow C: \{2^*Na, Nb, B\}K_c$
 $C_b \rightarrow A: \{Na, Nb, B\}K_a$
3. $A \rightarrow B: \{Nb\}K_b$

In order to allow the receivers of messages 2 and 3 to authenticate the messages the encryption functions needs to act like a message authentication code (MAC) which guarantees that the message was written with knowledge of a shared secret (Na or Nb in this case). This leads to the following alternative protocol.

1. $A \rightarrow B: \{Na\}K_b, A$
2. $B \rightarrow A: \{Nb\}K_a, h(Na, B)$

3. $A \longrightarrow B: h(Nb, A, B)$

Here $h(K, \cdot)$ may be a keyed one-way hash function such that need K is needed to calculate it, and it does not give away K . Several constructions for such functions exist in the literature. Further considerations on protocol design for entity authentication are discussed below.

It should be noted that the above attack is probably prevented by including strict formatting in the RSA encrypted messages, such as is recommended by the RSA Encryption Standard PKCS #1 or by Bellare and Rogaway. Such formatting is intended to ensure that the encrypting agent must be aware of the whole plaintext in order to form a valid ciphertext. In other words encryption of a shared secret (such as the nonce N_a or N_b in the Needham-Schroeder protocol) gives the ciphertext the essential property of a MAC. Notice also that the attack is not a ‘typing’ attack, since inclusion of, say, one bit typing tags would still allow the attack to succeed with high probability.

8 Conclusion

In this report we covered seven subjects. Starting from defining and describing about different block cipher modes, then, define an attack on ECB. After, we explained about hybrid encryption and OAEP. More, we defined some logical attacks and provided some examples. Finally, we introduced Needham Schroeder.

9 Reference

1. www.wikipedia.com
2. <http://dimacs.rutgers.edu/Workshops/Security/program2/boyd/node14.html>
3. <http://www-fourier.ujf-grenoble.fr/~pev/scci>

login: scci2008

passwd: scci2008