CSA E0 235: Cryptography	29th March, 2015
Lecture 4	
Instructor: Arpita Patra	*Submitted by: Cressida Hamlet*

1 Introduction

Last lecture we saw, why we need to capture CPA security notion. Now we will see, we also need to give decryption oracle service to the adversary so that we can represent the real world situations more accurately.

What we will see in this lecture:

- CCA security, stronger than CPA security
- Need for CCA security
- Introduction to MAC
- Security in MAC

2 Chosen-Ciphertext Attack security

In chosen-plaintext attack, we gave the adversary the ability to access encryption oracle service. A *chosen-ciphertext attack* is even more powerful. In a chosen-ciphertext attack, the adversary has the ability not only to obtain encryptions of messages of its choice, but also to obtain the decryption of ciphertexts of its choice. Formally, we give the adversary access to a decryption oracle in addition to an encryption oracle.

Consider the following experiment for any private-key encryption scheme $\pi = (\text{Gen}, \text{Enc}, \text{Dec})$, adversary $\mathcal{A}$, and value n for the security parameter.

The CCA indistinguishability experiment $\mathbf{PrivK}^{\mathbf{cca}}_{\mathcal{A},\pi}(n)$:

1. *A key k is generated by running* $\text{Gen}(1^n)$.
2. *Adversary $\mathcal{A}$ is given input* 1^n *and oracle access to* $\text{Enc}_k(\cdot)$ *and* $\text{Dec}_k(\cdot)$. *It outputs a pair of messages m_0, m_1 of the same length.*
3. *A uniform bit $b \in \{0,1\}$ is chosen, and then a ciphertext $c \leftarrow \text{Enc}_k(m_b)$ is computed and given to $\mathcal{A}$. We call c the* challenge ciphertext.
4. *The adversary A continues to have oracle access to* $\text{Enc}_k(\cdot)$ *and* $\text{Dec}_k(\cdot)$, *but is not allowed to query the latter on the challenge ciphertext itself. Eventually, $\mathcal{A}$ outputs a bit b'.*
5. *The output of the experiment is defined to be* 1 *if $b' = b$, and* 0 *otherwise. If the output of the experiment is* 1, *we say that $\mathcal{A}$ succeeds.*

DEFINITION 1. *A private-key encryption scheme π has* indistinguishable encryptions under a chosen-ciphertext attack*, or is* CCA-secure*, if for all probabilistic polynomial-time adversaries $\mathcal{A}$ there is a negligible function* negl *such that:*

$$\Pr[\text{PrivK}^{cca}_{\mathcal{A},\pi}(n) = 1] \leq \tfrac{1}{2} + \text{negl}(n),$$

where the probability is taken over all randomness used in the experiment.

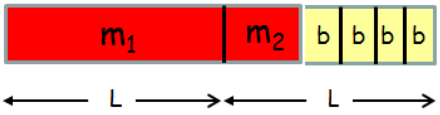
CCA Multiple message security:
Theorem: Any cipher that is CCA-secure is also CCA-secure for multiple encryptions.
Sufficient to prove CCA-security for single message; rest is for free.

3 Need for CCA security

Decryption oracle is more practical than getting an encryption oracle. Consider the case where a bank customer sends an encrypted message to transfer $100 to another account to the bank. After receiving the encrypted message bank may send a confirmation message/call to the customer asking if the customer really want to transfer the money. Any adversary seeing this will understand the previous encrypted message corresponds to transfering $100. A similar case is : An attacker sends an arbitrary ciphertext c(for an unknown message) to army headquarters and waits for the ciphertext to be decrypted and observes the behavior/movements of the army, which will give a hint what c corresponds to.

Even the knowledge of whether a modified ciphertext decrypted correctly or not can help an attacker to completely find the underlying plaintext !!

Padding oracle attack can be easily launched on several practically deployed ciphers. Consider a popular padding *PKCS#5 padding.* If we have a message(m), with length not a multiple of the block size, what can we do? Let b be the number of bytes need to be appended in the last block of message to make its length multiple of block size(L bytes). Append b bytes to the last block of m, each of them represending the integer value b as shown below.



Attack in CBC mode with PKCS#5 Padding:

Decryption in CBC mode with PKCS#5 Padding:
$m'_2 = F^{-1}(c_2) \oplus c_1$
$m_1 = F^{-1}(c_1) \oplus c_0$
Read the final byte value b. If the last b bytes of m_2 all have value b then strip-off the pad and output m, else output bad padding or request for re-transmission.

Now think what will happen if a adversary changes the first byte of c_1. It will affect the first byte of m'_2 calculated by the decryption algorithm. If the value of b is L, then the first byte of

m'_2 should have been b, so the change in the first byte c_1, will show a invalid ciphertext from the decryption algorithm. From this the adversary can understand the value of b is L. If the decrption algorithm says success, then the adversary can understand the value of b is less than L. Like this adversary can change in different bytes of c_1 from left to right, so the first success is in i^{th} byte then the value of $b = L - i + 1$. Now the value of b or $|m|$ is leaked to the adversary. Let B be the b+1^{th} byte from last. Consider what will happen if we change last b+1 bytes of c_1 by Δ_i, where $\Delta_i = (0, 0, 0, ...i, (b+1) \oplus b, (b+1) \oplus b, ..)$. Decryption oracle will return success only if B$\oplus i$ is same as $(b + 1)$. After atmost 256 attempts, adversary can find out the value of B. CPA secure CBC mode scheme is thus broken!

This shows that the adversary having control over what is decrypted will help the adversary to break the secrecy. CCA security is important.

4 Introduction to MAC

How the access to decryption oracle is useful for the adversary - when it is easy to manipulate known ciphertexts to obtain new ciphertexts so that the relation between the underlying messages are known to him and then get decrypted the changed ciphertext and get some information about the original message. The adversory here is malicious.

How to prevent this attack:

- Creating a new ciphertext will be nearly impossible
- Changing a ciphertext should either result in an incorrect ciphertext or should decrypt to a plaintext which is unrelated to the original plaintext
- The above two, together makes DO useless to the adversary.

Message Authentication Codes(MAC) help us to get such a cipher.

Message Integrity and Authentication
In secure-communication, is it enough to keep privacy of the message? We need to guarantee that a message received indeed originated from the correct sender not an intruder - *issue of message authentication.* Even it is confirmed that the message is authenticated, is there any guarentee that the message content is unaltered - *issue of message integrity*
Encryption scheme doesnot help to get MI and MA unless designed for it.

Now lets define formally message authentication code, which help us to obtain MI and MA.

DEFINITION 2. *A* Message Authentication Code *or* MAC *consists of three probabilistic polynomial-time algorithms* (Gen, Mac, Vrfy) *such that:*

1. *The* key-generation algorithm Gen *takes as input the security parameter* 1^n *and outputs a key* k *with* $|k| \geq n$.
2. *The* tag=generation algorithm Mac *takes as input a key* k *and a message* $m \in \{0,1\}^*$*, and outputs a tag* t*. Since this algorithm may be randomized, we write this as* $t \leftarrow \text{Mac}_k(m)$.

3. *The deterministic* verification algorithm Vrfy *takes as input a key k, a message m, and a tag t, It outputs a bit b, with b =1 meaning* valid *and b = 0 meaning* invalid*. We write this as b:=*$\text{Vrfy}_k(m,t)$*.*

5 Security in MAC

To know what we need to secure in Mac we first need to know the threats and breaks.

Threat:
Attacks an attacker can mount :
Chosen Message Attack(CMA) - in spirit of CPA; models the fact that adversary can influence the honest parties to authenticates a message of its choice.
Chosen Message and Verification Attack(CMVA) - in spirit of CCA models the fact that the adversary can influence the honest parties to authenticate messages and verify tag, message pair of its choice.

Break:
New (m,t) pair such that adversary has not seen a tag on m for CMA
New (m,t) such that adversary has not seen (m,t) before - stronger notion for CMVA

With this in mind we can formally define message authentication experiment with message authentification code π = (Gen, Mac, Vrfy), an adversary $\mathcal{A}$, and value n for the security parameter:

The message authentication experiment Mac-forge$_{\mathcal{A},\pi}(n)$:

1. *A key k is generated by running* $\text{Gen}(1^n)$*.*
2. *The adversary* $\mathcal{A}$ *is given input* 1^n *and oracle access to* $\text{Mac}_k(\cdot)$*. The adversary eventually outputs (m, t). Let* $\mathcal{Q}$ *denote the set of all queries that* $\mathcal{A}$ *asked its oracle.*
3. $\mathcal{A}$ *succeeds iff (1)* $\text{Vrfy}_k(m,t) = 1$ *and (2)* $m \notin \mathcal{Q}$*. In that case the output of the experiment is defined to be* 1*.*

A MAC is secure if no efficient adversary can succeed in the above experment with non-negligible probability:

DEFINITION 3. *A message authentication code* π = (Gen,Mac, Vrfy) *is* existentially unforgeable under an adaptive chosen-message attack*, or just* secure*, if for all probabilistic polynomial-time adversaries* $\mathcal{A}$*, there is a negligible function* negl *such that:*

$$\Pr[\text{Mac-forge}_{\mathcal{A},\pi}(n) = 1] \leq \text{negl}(n).$$

What is not captured in MAC security definition

- If $\mathcal{A}$ returns (m,t) for a already queried message, we don't consider that as a break - what happens if an attacker just sends the message many times at a later point of time. What if the message is to transfer $100 to some account? - *Replay Attack*.
- Why Replay attack is not taken care in Mac definition? - Whether this attack is a concern depends upon the actual application scenario. So it is better to deal with this in the outer protocol (to avoid this attack we can use timestamp, counters. etc...).

Strong MAC:
As defined, a secure MAC ensures that an adversary cannot generate a valid tag on a new message that was never previously authenticated. But it does not rule out the possibility that an attacker might be able to generate a *new tag* on a previously authenticated message. That is, a MAC guarantees that if an attacker learns tags $t_1, ...$ on messages $m_1, ...$, then it will not be able to forge a valid tag t on any message $m \notin \{m_1, ...\}$. However, it may be possible for an adversary to forge a different valid tag $t'_1 \neq t_1$ on the message m_1. In general, this type of adversarial behavior is not a concern. Nevertheless, in some settings it is useful to consider a stronger definition of security for MACs where such behavior is ruled out.

Formally, we consider a modified experiment *Mac-sforge* that is defined in exactly the same way as Mac-forge, except that now the set $\mathcal{Q}$ contains pairs of oracle queries and their associated responses. (That is, $(m, t) \in \mathcal{Q}$ if A queried $\text{Mac}_k(m)$ and received in response the tag t.) The adversary $\mathcal{A}$ succeeds (and experiment Mac-sforge evaluates to 1) if and only if $\mathcal{A}$ outputs (m, t) such that $\text{Vrfy}_k(m, t) = 1$ and $(m, t) \notin \mathcal{Q}$.

DEFINITION 4. *A message authentication code* π = (Gen,Mac, Vrfy) *is* strongly secure *or a* strong MAC*, if for all probabilistic polynomial-time adversaries* $\mathcal{A}$*, there is a negligible function* negl *such that:*

$$\Pr[\text{Mac-sforge}_{\mathcal{A},\pi}(n) = 1] \leq \text{negl}(n).$$