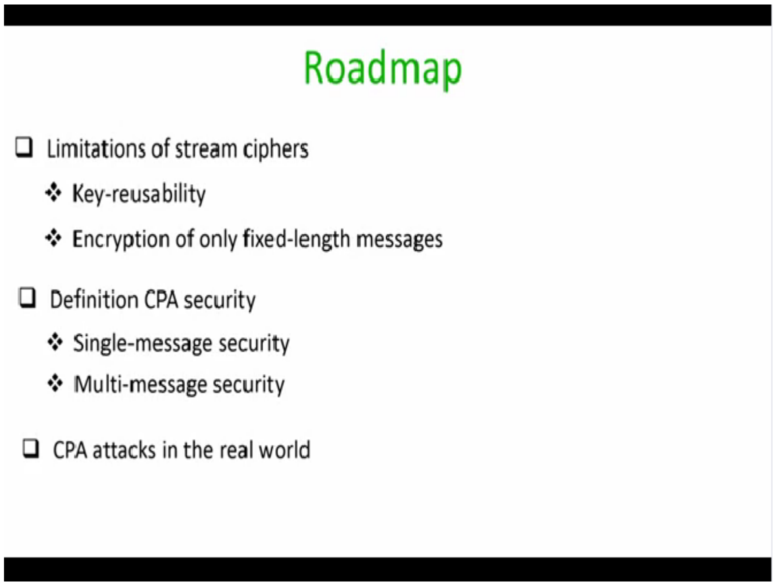
Foundations of Cryptography

Prof. Dr. Ashish Choudhury
(Former) Infosys Foundation Career Development Chair Professor
International Institute of Information Technology-Bangalore

Lecture-13
CPA Security

Hello everyone, welcome to lecture 12. The plan for this lecture is as follows.

(Refer Slide Time: 00:32)

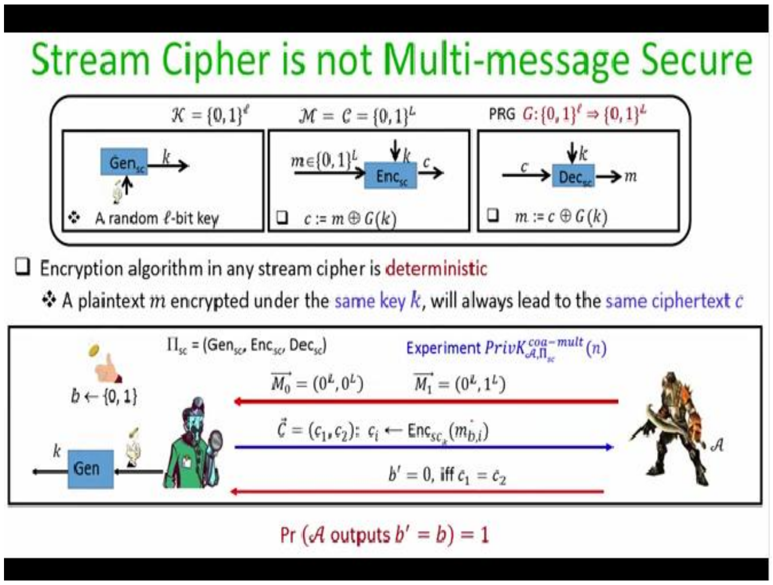


Roadmap

- ❑ Limitations of stream ciphers
  - ❖ Key-reusability
  - ❖ Encryption of only fixed-length messages
- ❑ Definition CPA security
  - ❖ Single-message security
  - ❖ Multi-message security
- ❑ CPA attacks in the real world

In this lecture we will discuss the limitations of stream ciphers. And we will introduce the concept of CPA security and we will see the motivations of CPA attacks in the real world scenario.

(Refer Slide Time: 00:46)


$\mathcal{K} = \{0,1\}^\ell$
$\mathcal{M} = \mathcal{C} = \{0,1\}^L$
PRG $G: \{0,1\}^\ell \Rightarrow \{0,1\}^L$
Gen_{sc}
k
A random ℓ-bit key
$m \in \{0,1\}^L$
Enc_{sc}
c
$c := m \oplus G(k)$
Dec_{sc}
$m := c \oplus G(k)$
$\Pi_{sc} = (\text{Gen}_{sc}, \text{Enc}_{sc}, \text{Dec}_{sc})$
Experiment $PrivK^{coa-mult}_{\mathcal{A},\Pi_{sc}}(n)$
$b \leftarrow \{0,1\}$
$\overrightarrow{M_0} = (0^L, 0^L)$
$\overrightarrow{M_1} = (0^L, 1^L)$
$\vec{C} = (c_1, c_2);\ c_i \leftarrow \text{Enc}_{sc_k}(m_{b,i})$
$b' = 0$, iff $c_1 = c_2$
Gen
$\mathcal{A}$

Stream Cipher is not Multi-message Secure

- Encryption algorithm in any stream cipher is deterministic
  - A plaintext m encrypted under the same key k, will always lead to the same ciphertext c

Pr ($\mathcal{A}$ outputs $b' = b$) = 1

So, let us recall the stream cipher. So in the stream cipher we have a pseudo random generator, right, which takes a random seed of l bits and gives you an output of L bits and outcome is pseudo random in the sense that output cannot be distinguished from the outcome of a truly random number generator. So the encryption algorithm of the stream cipher basically expands the seed for the encryption using the pseudo random generator by treating the key for the encryption algorithm as the seed for the pseudo random generator.

And the expanded output of the pseudo random generator is used as a pad to mask your message. And in the decryption algorithm, we just do the reverse operation. So that is a stream cipher. So the first restriction that is imposed by stream cipher is that it is not multi message secure. That means we can use stream cipher just to encrypt one message using a key, we cannot encrypt multiple messages using the same key.

And the main reason for that is that the encryption algorithm in any stream cipher is deterministic because inside the encryption algorithm, we will be expanding the key using pseudo random generator. That means if I want to encrypt a plain text m under the key k, it will lead to the same ciphertext c, right. So if I have a message says 00. And if I have a key k, and if I want to encrypt a message 00 using the same key multiple times, I will always obtain the same ciphertext because my encryption algorithm is not randomized.

There is no internal randomness as part of the stream cipher encryption process. And based on this idea, we can actually give an instantiation of the COA multi message security experiment where we can show an adversary who can actually identify what message is encrypted by the challenger in the experiment. So here is the instance of the experiment. So since this is a multi message COA security experiment, our adversary basically submits a pair of vector of messages where in the 0th vector we have 2 messages consisting of all 0s.

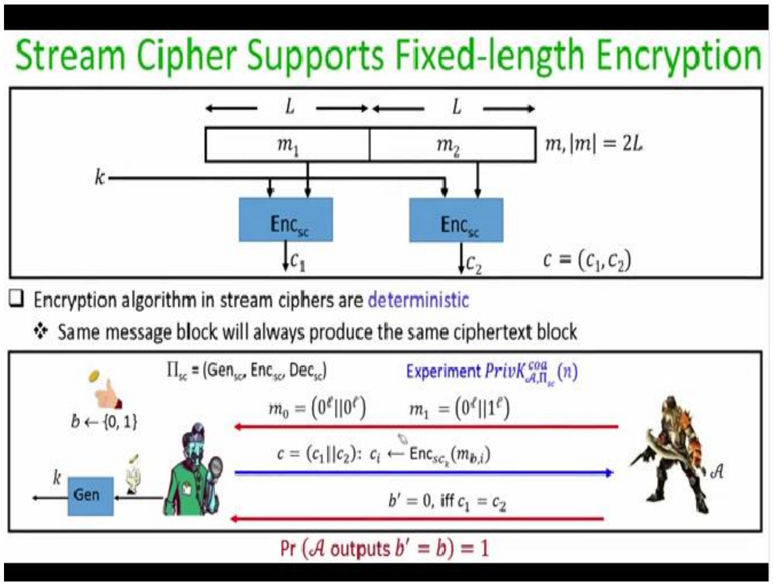
Whereas in the first vector m_1 the first message block is the first message is all 0s whereas the second message is all 1s. Now as per the rules of the experiment, the challenger here will run the key generation algorithm and it will randomly decide the index either m_0 or m_1 which it wants to encrypt. And accordingly it prepares the challenge ciphertext for the adversary where the challenge ciphertext vector is denoted by C vector and it will consist of 2 ciphertext right.

Now, notice that the way adversary has submitted the pair of challenge messages m_0 and m_1 ,adversary knows that if the challenge ciphertext vector is an encryption of the vector m_0, then both c_1 and c_2 will be identical because both of them would have been the encryption of the same message then all 0s. Whereas the adversary knows that if the challenge ciphertext is an encryption of the message vector m_1, then definitely the cipher text c_1 will be different from ciphertext c_2.

Because the message is in the first vector m_1 are different. So based on this intuition, our adversary is going to output as follows it will output the bit b**'** = 0 if and only f $c_1 = c_2$. And now it is easy to see that our adversary is going to output the right bit. Namely, it will correctly identify the challenge message which is encrypted in the side challenge ciphertext c vector with probability 1.

Because if indeed the 0th vector is encrypted in the vector C, then definitely $c_1 = c_2$. Whereas if the message vector m_1 would have been encrypted in the challenge, ciphertext C, then c_1 is not equal to c_2. So clearly our adversary is going to win this game with probability 1, and which shows that in a stream cipher we cannot encrypt multiple messages using the same key, at the most we can encrypt only a single message.

(Refer Slide Time: 04:52)



Stream Cipher Supports Fixed-length Encryption

L
L
m_1
m_2
$m, |m| = 2L$
k
Enc_{sc}
Enc_{sc}
c_1
c_2
$c = (c_1, c_2)$

- Encryption algorithm in stream ciphers are deterministic
  - Same message block will always produce the same ciphertext block

$\Pi_{sc} = (\text{Gen}_{sc}, \text{Enc}_{sc}, \text{Dec}_{sc})$
Experiment $PrivK^{coa}_{\mathcal{A},\Pi_{sc}}(n)$
$b \leftarrow \{0, 1\}$
$m_0 = (0^\ell || 0^\ell)$
$m_1 = (0^\ell || 1^\ell)$
$c = (c_1 || c_2)$: $c_i \leftarrow \text{Enc}_{sc_k}(m_{b,i})$
k
Gen
$b' = 0$, iff $c_1 = c_2$
$\mathcal{A}$

Pr ($\mathcal{A}$ outputs $b' = b$) = 1

The second restriction which is imposed by the stream cipher is that it supports encryption of only fixed length messages. So for instance imagine we have a stream cipher over the message space of L bits, right. So, imagine sender has a message which has size more than L bits. For instance, imagine that sender has a message which consists of 2 L bits. Now, a naive way to encrypt this message would have been to divide your message into 2 blocks each of size L bits, L bits.

And encrypt each of the message blocks m_1 and m_2 using the key k by running 2 instances of the stream cipher encryption algorithm right and the resultant ciphertext will be c_1, c_2. Now the question is, is this method of dividing your larger message into smaller blocks of size L bits and encrypting each individual message blocks is COA secured or not. And it turns out the answer is no. This is because of the problem that was stream cipher encryption algorithm is deterministic.

That means, if I have a larger message consisting of several blocks of L bits, and if I have repeating blocks in this larger message, then wherever that repeating block is appearing, I will obtain the same ciphertext blocks because the same k used to encrypt all those blocks, and there is no internal randomness as part of the encryption algorithm. So based on this weakness, we can actually create an instance of the COA single message security experiment, and show that indeed they are exists an adversary strategy, which can clearly identify what message has been encrypted by the challenger. So here is the instance of the COA single message security experiment. So, in this instance adversary submits 2 messages m_0 and m_1. And I stress that this is an instance of a single message COA experiment because our goal is to show that there is an adversary who can distinguish apart encryption of 2 larger messages where our encryption strategy is to divide the larger messages into small blocks and independently encrypt those blocks.

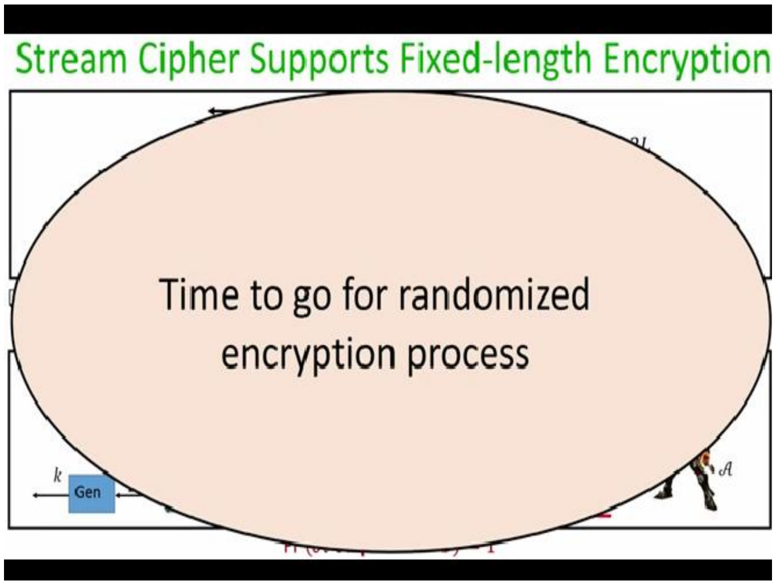
So, in this case basically adversary is submitting 2 large messages, m_0 and m_1, where m_0 basically consist of 2 blocks of all 0s and m_1 would consist of the first block being all 0s and the second block being all 1s. Now, the challenger prepares the challenge ciphertext and to do that, basically challenger runs the key generation algorithm of the stream cipher and obtains a random key and it randomly decides either the message m_0 or m_1 for encryption right.

So based upon the index b which is randomly chosen by the challenger, the challenger prepares the challenge ciphertext consisting of 2 ciphertext block c_1 and c_2. And notice that here adversary knows that if the challenge ciphertext is an encryption of m_0, then definitely c_1 and c_2 are going to be same because our encryption process is a deterministic algorithm. On the other hand our adversary knows that if the challenge ciphertext C is an encryption of the message m_1, then definitely the cipher text block c_1 will be different from the cipher text block c_2.

Because the internal blocks in the message m_1 are different. So, based upon this strategy, the adversary is going to produce his output as follows it outputs b' = 0 if and only if $c_1 = c_2$. And it is easy to see that the adversary strategy in this case indeed correctly identifies what message has been encrypted in the challenge ciphertext c with probability 1, that means that 100% chance the adversary is going to distinguish apart the encryption of m_0 from an encryption of m_1.

And hence this process of dividing your larger message into smaller blocks of L bits, L bits and encrypting each block by running an instance of stream cipher but with the same key k is not a secure mechanism right.

(Refer Slide Time: 08:58)

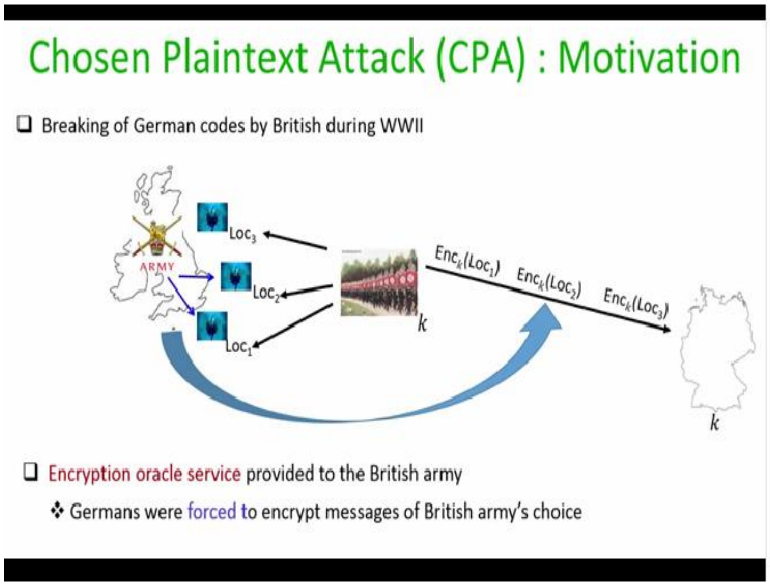


Stream Cipher Supports Fixed-length Encryption

Time to go for randomized encryption process

So, as you can see that both the weaknesses that we have pointed out in the context of stream cipher is because of the fact that the encryption algorithm in the stream cipher is a deterministic algorithm. And hence, it is high time that we have to now go for randomized encryption process to get rid of these 2 shortcomings.

(Refer Slide Time: 09:16)



Chosen Plaintext Attack (CPA) : Motivation

- Breaking of German codes by British during WWII

- Encryption oracle service provided to the British army
    - Germans were forced to encrypt messages of British army's choice

So what we are now going to do next is we are going to introduce more stronger attack model, namely chosen plaintext attack model, which we also call a CPA. And we will later see that when we construct the encryption process, which are CPA secure, we can actually get rid of both the problems or the shortcomings that we encountered in the context of stream cipher. So before going into the formal details of CPA attack model, let us see some real world applications which we want to actually capture in our attack model.

So this is an example taken in the context of second world war. So this example basically demonstrates how basically the British army broke the German codes by actually getting an encryption oracle service from the German army. So what the British army did is the following. They purposely planted sea mines at known locations which I denote by say location 1, location 2 and location 3 and so on right.

And what the German army did basically is whenever they actually found a sea mine being planted at a specific location, they were sending back the identity of those locations back to their headquarters in an encrypted fashion, right. So here we have the sender, the German army which are actually finding out the locations of the sea mines and headquarter is acting as the receiver. And there is a common key random key established between the sender and the receiver.

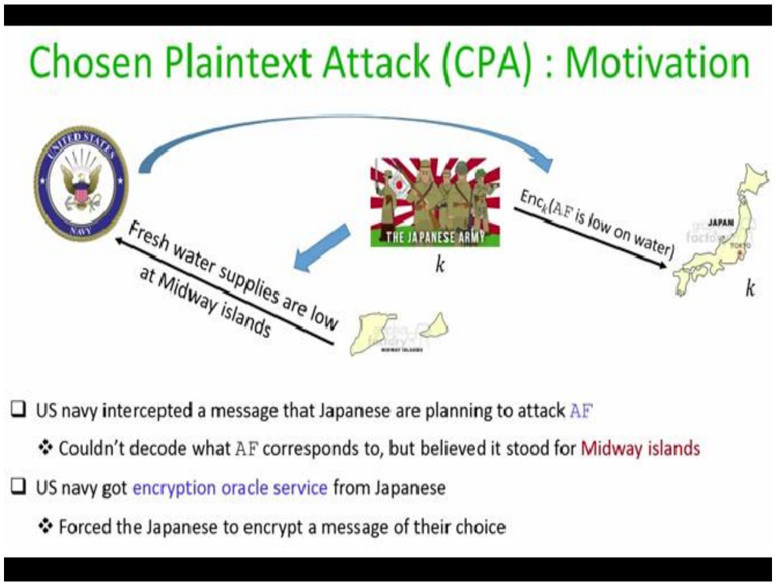
And whenever the sender namely in this context, the German army were identifying the locations of the sea mine, they were encrypting the identity of those locations under an unknown key k and sending it back to the receiver, namely, the headquarters. And when this encrypted communication was happening, right, what the British army did is basically they intercepted this encrypted communication.

And they knew what message has been actually encrypted, right. So what is not known to the British army here is the key k, that is not known to the British army, but what the British army knows in this particular example is what message has been encrypted and communicated to the receiver. So in some sense, we can imagine that in this example, the British army, it is acting as an adversary, right.

Because it is sitting in between the sender and the receiver and it is somehow getting an encryption oracle service, in the sense that the British army were forcing the Germans to encrypt messages of British army’s choice. And that too without letting the sender or the receiver namely the German army, aware of this fact, right. The reason we are calling it as an oracle services that the sender who is actually sending the encrypted messages is not aware of the fact that it is influenced to encrypt messages of adversary’s choice.

So that is the kind of attack scenario which has actually happened in the real world context. And if you see our cipher text only attack model, definitely this kind of scenario is not captured in our cipher text only attack model, because this is a more powerful attack, which can happen and which is not captured properly, or not at all captured in the ciphertext only attack model.

(Refer Slide Time: 12:38)



Chosen Plaintext Attack (CPA) : Motivation

Enc_k(AF is low on water)
THE JAPANESE ARMY
k
k
Fresh water supplies are low at Midway islands
JAPAN

- ❑ US navy intercepted a message that Japanese are planning to attack AF
  - ❖ Couldn’t decode what AF corresponds to, but believed it stood for Midway islands
- ❑ US navy got encryption oracle service from Japanese
  - ❖ Forced the Japanese to encrypt a message of their choice

Now let us see another example which motivates the chosen plaintext attack model. And this is again in the context of world war. And here the scenario is the following the Japanese army they were actually keeping track of the movements of the US Navy and it is acting as a sender and their headquarters back Japan is the receiver and they have a shared key k. And somehow, during the world war, US Navy intercepted a message that Japanese are planning to attack a location called AF.

So AF was kind of encoded location. So, after spending significant amount of time, the US Navy were not able to confirm what exactly the encoded location AF corresponds to. But they strongly believed that it stands for midway islands which is a small island in the US. But they were not sure whether exactly AF corresponds to the midway Island or not.

So what to confirm their intuition or understanding basically what US army did is they got an encryption oracle service from the Japanese, basically the forced the Japanese to encrypt a message of their choice and without letting the Japanese army knowing about this fact. So the way the US Navy did, this is as follows. So what they basically did is they simply, they instructed the force which was located at the midway Island to communicate a message to the US headquarters stating that freshwater supplies are low at midway islands.

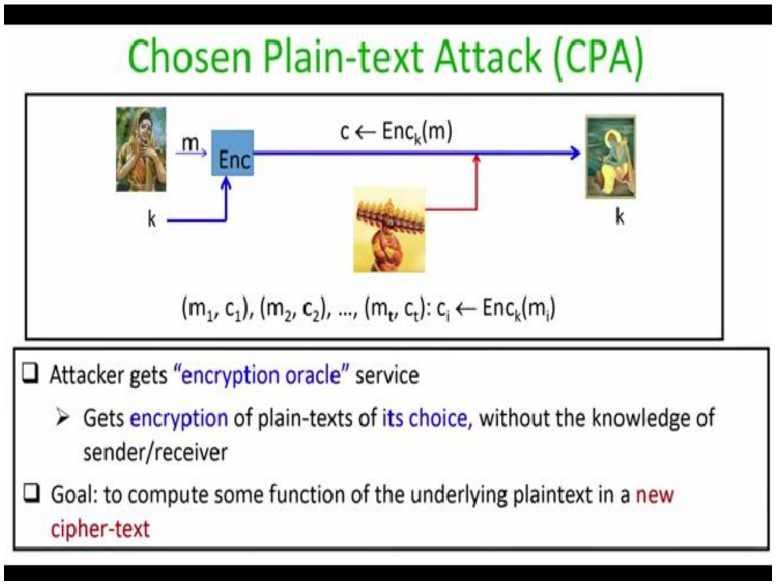
And this message was sent in clear, and they actually wanted the Japanese to intercept this message. And this message was intercepted by the Japanese army. And as soon as the Japanese army actually intercepted this message, they did an encrypted communication back to their headquarter under the key k which was not known to the US army and encrypted message was basically corresponding to the message that AF is low on water.

And is encrypted communication was then intercepted by the United States Navy. And now you can clearly see that as soon as this encrypted message was intercepted, the US Navy was confirmed that indeed AF corresponds to the midway Island. So now you can see here, the US Navy basically acted as a adversary sitting between the sender and the receiver, and they somehow got an encryption oracle service from the sender, namely the Japanese army in this context.

And once they got confirmed that AF corresponds to the midway Island, they basically ended up deploying additional troops in the Midway Island. And the Japanese thought that indeed, Midway islands actually having a low water supply, which was not actually the case. So they were actually targeting the midway Island. They were not anticipating that the troops have been strengthened in the midway Island.

And actually the Japanese army ended up getting suffering heavy casualties right. So again, this is an example of an attack real world attack scenario which cannot be captured properly in the ciphertext only attack model.

(Refer Slide Time: 15:42)



Chosen Plain-text Attack (CPA)

m
Enc
$c \leftarrow Enc_k(m)$
k
k
$(m_1, c_1), (m_2, c_2), \ldots, (m_t, c_t): c_i \leftarrow Enc_k(m_i)$

- Attacker gets "encryption oracle" service
  - Gets encryption of plain-texts of its choice, without the knowledge of sender/receiver
- Goal: to compute some function of the underlying plaintext in a new cipher-text

That means now we have to go to a more stronger attack model which we call as the CPA attack model. And the scenario in the CPA attack model is as follows. We have a sender and a receiver and a common random key is somehow agreed upon between the 2 entities. And here the adversary is assumed to get encryption oracle service either from the sender or the receiver for simplicity as assume is getting the encryption oracle service from the sender.

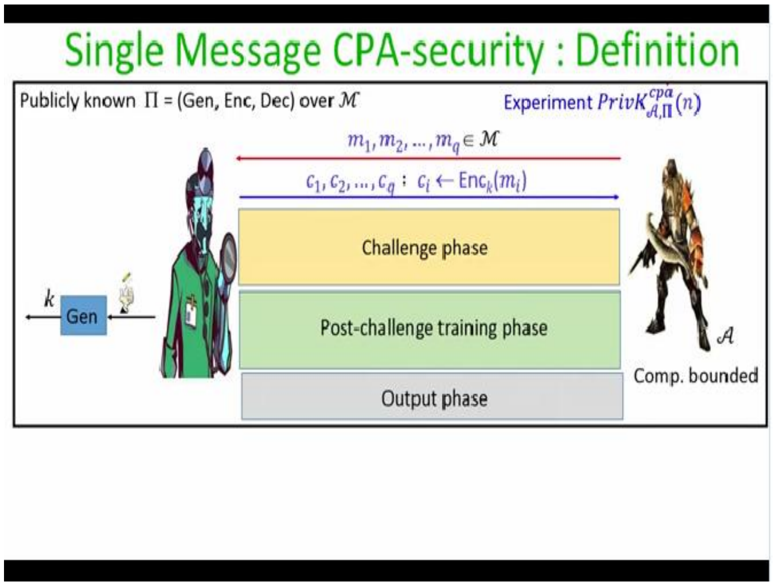
So what I mean by encryption oracle services basically is that the adversary somehow gets access to the encryption "box" (in quotes). And basically the adversary can influence the sender to encrypt whatever plaintext adversary wants to get encrypted without actually letting the sender know or be aware of the fact that the sender is actually encrypting messages of adversary's choice.

And this can happen for any number of queries as long as the running time of the attacker is feasible or computationally bounded. So in this case, for example, the adversary can first get an encryption of the message m_1 and then based on the encryption of the message m_1 it can decide next what message it should ask for the encryption from the encryption oracle. So for example, m_2 and like that it can adaptively query the encryption oracle, right.

So what I mean by adaptive queries is that this adversary does not submit all its encryption queries in a single shot, our adversary could be a smarter adversary. And based on the responses that the adversary has seen from the previous interaction with the encryption oracle, it can decide what to ask from the encryption oracle in the subsequent queries. In that sense our adversary is an adaptive adversary.

And once the adversary has got access to the encryption oracle and it has prepared a database of several (messages, ciphertext) pairs, we are all those corresponding ciphertext are the encryptions of the corresponding plain text under the same unknown key k. The goal of the adversary is as follows : a fresh message is encrypted, and it is intercepted by our adversary, and the goal of the adversary is basically to compute some function of the underlying plain text in this fresh ciphertext. I stress that it might be the case that a fresh message which has been encrypted and now is intercepted by the adversary might already belong to the list of the messages for which the adversary would have got the encryption oracle query. When we say that we want to construct a CPA secure scheme, our scheme should take care of this scenario as well right.

(Refer Slide Time: 18:07)



Single Message CPA-security : Definition

Publicly known Π = (Gen, Enc, Dec) over $\mathcal{M}$

Experiment $PrivK^{cpa}_{\mathcal{A},\Pi}(n)$

$m_1, m_2, \ldots, m_q \in \mathcal{M}$
$c_1, c_2, \ldots, c_q$: $c_i \leftarrow \mathrm{Enc}_k(m_i)$
Challenge phase
Post-challenge training phase
Output phase
k
Gen
$\mathcal{A}$
Comp. bounded

So now let us make the definition of CPA security a little bit more formal through our experiment. So we have a publicly known encryption process over some known message space and we have a computationally bounded adversary. And nomenclature of the experiment is as follows. We call the experiment as $PrivK^{cpa}_{\mathcal{A},\Pi}$ with respect to the adversary A placed against a scheme Π and this is a single message CPA security.

We are basically the goal of the attacker is to identify what message has been encrypted basically, it has to distinguish apart encryption between encryption of 2 messages based on the encryption oracle service, and n here is the security parameter right. So on a very high level basically the game consists of 4 stages. We have a pre-challenge training phase, we have a challenge phase, we have a post challenge training phase and we have an output phase.

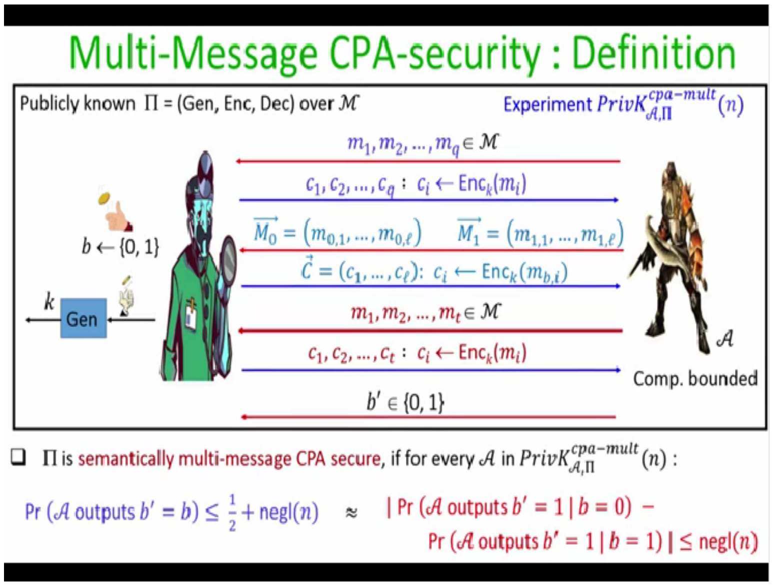
If we compare this experiment with our COA security game, then the challenge phase and output phase remains the same as in the COA game. The new things here are the pre-challenge training phase and the post challenge training phase. So let us go a little bit deeper into what exactly this pre-challenge and post challenge training phases are. So in the pre-challenge training phase, basically our adversary gets encryption oracle service.

So it submits several messages of its choice adaptively and asks for the encryption of those messages. So, for the simplicity purpose here, I have represented all the encryption oracle queries being submitted in one go. But that may not be the case, our adversary can submit any message of its choice based on the response that he has seen earlier. So once this encryption queries are submitted to our challenger, what the challenger does is it has to respond to these queries.

And this models basically the adversary gets access to the encryption oracle service where it can influence the sender to encrypt whatever messages the adversary feels like, and to respond to the encryption oracle queries, what the challenger does is, it runs the key generation algorithm or obtains a random key and encrypt the encryption oracle messages m_1, m_2, …, m_q as per the encryption algorithm of the scheme Π right.

So there is no restriction on the number of queries, what kind of queries the adversary can submit for the encryption oracle queries and so on. The only restriction is that the running time of the adversary should be computationally bounded namely it should be some polynomial function of our security parameter that automatically enforces a restriction on the queue, namely the number of queries that the adversary can ask for. So that is a pre-challenge training phase.

(Refer Slide Time: 20:59)



Multi-Message CPA-security : Definition

Publicly known Π = (Gen, Enc, Dec) over $\mathcal{M}$

Experiment $PrivK^{cpa-mult}_{\mathcal{A},\Pi}(n)$

$b \leftarrow \{0,1\}$
k
Gen
$m_1, m_2, \ldots, m_q \in \mathcal{M}$
$c_1, c_2, \ldots, c_q$: $c_i \leftarrow \text{Enc}_k(m_i)$
$\overrightarrow{M_0} = (m_{0,1}, \ldots, m_{0,\ell})$ $\overrightarrow{M_1} = (m_{1,1}, \ldots, m_{1,\ell})$
$\vec{C} = (c_1, \ldots, c_\ell)$: $c_i \leftarrow \text{Enc}_k(m_{b,i})$
$m_1, m_2, \ldots, m_t \in \mathcal{M}$
$c_1, c_2, \ldots, c_t$: $c_i \leftarrow \text{Enc}_k(m_i)$
$b' \in \{0,1\}$
$\mathcal{A}$
Comp. bounded

- Π is semantically multi-message CPA secure, if for every $\mathcal{A}$ in $PrivK^{cpa-mult}_{\mathcal{A},\Pi}(n)$:

$$\Pr(\mathcal{A} \text{ outputs } b' = b) \leq \frac{1}{2} + \text{negl}(n) \quad \approx \quad |\Pr(\mathcal{A} \text{ outputs } b' = 1 \mid b = 0) - \Pr(\mathcal{A} \text{ outputs } b' = 1 \mid b = 1)| \leq \text{negl}(n)$$

Now in the challenge phase, adversary challenges our challenger, and it submits a pair of messages from the plain text space. Since this is a single message security experiment, the only restriction on the pair of challenge messages is that the lengths should be the same. Apart from that there is absolutely no restriction that means the message is m_0 m_1 could be any of the messages that the adversary has already queried as part of the encryption oracle queries.

So to distinguish the messages m_0 and m_1 which was submitted as part of the challenge phase, I am using a different color here. But value wise there could be any of the messages which adversary has already queried in the past in the pre-challenge training phase. Now, the challenger has to prepare the challenge ciphertext and to do that what the challenger does is it at randomly decides whether to encrypt the message m_0 or whether to encrypt a message m_1.

And once it has decided what message it has to encrypt, it encrypts that particular message. So the encrypted message is denoted by m_b and with probability half m_b could be m_0 and with probability half m_b could be m_1. And now the challenge for the adversary is to identify what exactly has been encrypted in c* whether it is m_0 or whether it is m_1. Now, what we do here is we actually give the attacker more power here right.

So, once the adversary sees the challenge ciphertext we again let the adversary get access to the encryption oracle service and this basically models the fact that in the real world session between the sender and the receiver could consist of several messages and adversary might be interested to identify or break the security of only one particular message in that session based on whatever encryption oracle query it could get before that ciphertext is communicated and after that ciphertext is communicated.

So to model that, basically we are giving the adversary a post challenge encryption oracle service, where again it can ask for encryption of any message of its choice including the challenge messages, m_0 or m_1. There is absolutely no restriction. It can adaptively submit encryption oracle query and in response, it will learn the encryption of those messages. The only restriction is that the number of queries should be upper bounded by some polynomial function of the security parameter l.

Now, once the adversary gets that pre-challenge training phase, post challenge training phase, based on the responses it has obtained, it has to identify what has been encrypted in the challenge ciphertext c*, namely it has to identify whether it is m_0 or whether it is m_1 which has been encrypted in c*. So, that is an instance of a single message a CPA security experiment.

Now, the definition of single message CPA security is that we say that our scheme Π is single message CPA secure or semantically single message CPA secure, if for every probabilistic polynomial time algorithm participating in this experiment, there is some negligible function negl(n), such that the probability of adversary winning this experiment, which we denote as $P(\mathcal{A} \text{ outputs } b' = b) \leq \frac{1}{2} + negl(n)$ meaning the probability that A outputs or identifies the right message which has been encrypted in c*, is upper bounded by half plus that negligible function, where the probability is actually taken over the randomness of our challenger and the randomness of our adversary. So remember that all these instances of experiment are randomized experiment, because the verifier is going to use some random coins to decide what message to encrypt, what is the key generation algorithm going to output and the internal randomness which is going to be used as part of the encryption process. That is why the probability is over the randomness over the challenger.

And in the same way adversary might be asking for encryption oracle queries, where the nature of the queries can be determined randomly based on the internal coins of the adversary right. So that is our definition of single message CPA security. There is another way to put this definition. The second way to put this definition is that irrespective of whether $b = 0$ or whether $b = 1$, the adversary's response should be the same except with negligible probability.

That means, it does not matter whether c^* is an encryption of m_0 or whether c^* is an encryption of m_1. In both the cases from the viewpoint of the adversary, it should look like as if c^* is equally an encryption of m_0 as well as equally as an encryption of m_1 except with negligible probability that means the behavior or the output of the adversary should not be significantly apart, it should be almost the same irrespective of whether it is m_0 that is encrypted or whether it is m_1 is encrypted.

And again, we can formally prove that if we have a candidate encryption process, which is CPA secure as per this first condition, then it also implies that it is CPA secure as per the second condition. And in the same way we can prove that if we have a candidate encryption scheme is secure as per the second condition, then it is also secure as per this first condition. So depending upon our convenience, we can use any of these definitions.

And that is what we are going to do here. The first definition you can imagine as if the adversary has to identify correctly what has been encrypted. The second definition we imagine that we can interpret it as if the distinguishing advantage of the adversary to distinguish apart the 2 instances of the experiment is upper bounded by some negligible function. Now, you might be wondering here that why we are upper bounded, or why we required to upper bound the success probability of the adversary to half plus negl(n), why not 0.

Because there is always a naive adversary algorithm, namely a guessing adversary, which without actually requiring any kind of encryption oracle service can just guess what is encrypted in c^*. And the success probability of that adversary is always half we can never prevent that kind of attack. The additional negligible probability we are allowing here because remember, we are in the computationally secure world.

And as we discussed, during our one of our earlier presentation that one of the necessary evils associated with computational security is that we should be willing to let the adversary break the scheme, but that the chances of breaking the scheme should be so small that we should be willing to ignore it apart. So that is why we are willing to give that adversary an additional negligible advantage of breaking the scheme.

So that is the definition of single message CPA security. Now let us see the definition of multi message CPA security, where the sender would basically like to encrypt multiple messages using the same key and would like to maintain the security even in the presence of an adversary who has got access to the encryption oracle service. So on a very high level, the experiment is almost identical to the single message CPA security.

The differences are as follows. First of all, the name of the experiment is different, $PrivK_{\mathcal{A},\Pi}^{cpa-mult}(n)$ meaning now we are calling it CPA-mult instead of just CPA and like in the previous experiment adversary gets access to the pre-challenge training phase. So this is your pre-challenge training phase where adversary submits encryption oracle queries and get their responses, get the response from the encryption oracle service.

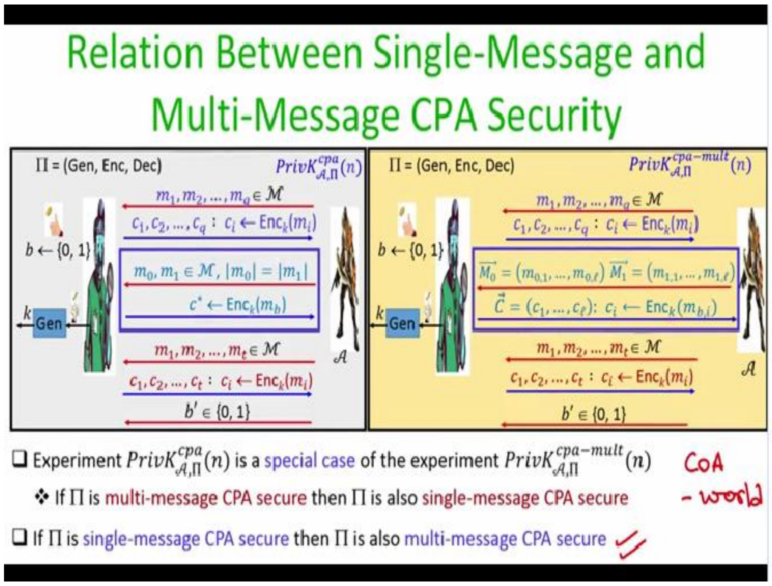
And now instead of submitting a pair of messages it basically submits a pair of vector of messages consisting of same number of messages l number of messages, and our restriction here is that the first message in the 0^{th} block and the first message in the first vector should be of the same length, in the same way the second message in the 0^{th} vector and the second message in the first vector should be of the same length.

And like that l^{th} message in the 0^{th} vector and l^{th} message in the first vector should be of the same length. So apart from the restriction on the length part, there is absolutely no restriction on the pair of the challenge vectors that the adversary is allowed to submit and now our challenger has to prepare the challenge vector of challenge ciphertext where it randomly decides with probability 1/2 to either encrypt the vector of messages in m$_0$, or to encrypt the vector of messages m$_1$.

And the goal of the adversary is basically to identify which vector of messages has been encrypted. Now, like in the previous experiment, we also give the adversary access to the post challenge training phase where after seeing the encrypted vector of ciphertext adversary can again ask for encryption of whatever messages it feels like. And then finally, there is an output phase where the adversary outputs what vector has been encrypted in the challenge ciphertext vector.

Now our definition is for the multi message CPA security is that we say our scheme Π is multi message CPA security for every probabilistic polynomial time adversary, participating in this experiment, there exists some negligible function such that the probability of our adversary identifying what vector has been encrypted in C is upper bounded by half plus negligible, or equivalently the distinguishing advantage of our adversary is upper bounded by negligible function. Both these conditions are equal. So that is our definition of our multi message CPA security.

(Refer Slide Time: 30:12)



Relation Between Single-Message and Multi-Message CPA Security

Π = (Gen, Enc, Dec)
$Priv K^{cpa}_{\mathcal{A},\Pi}(n)$
$m_1, m_2, \ldots, m_q \in \mathcal{M}$
$c_1, c_2, \ldots, c_q : c_i \leftarrow \mathsf{Enc}_k(m_i)$
$b \leftarrow \{0, 1\}$
$m_0, m_1 \in \mathcal{M}, |m_0| = |m_1|$
$c^* \leftarrow \mathsf{Enc}_k(m_b)$
k
Gen
$m_1, m_2, \ldots, m_t \in \mathcal{M}$
$c_1, c_2, \ldots, c_t : c_i \leftarrow \mathsf{Enc}_k(m_i)$
$b' \in \{0, 1\}$
$\mathcal{A}$
Π = (Gen, Enc, Dec)
$Priv K^{cpa-mult}_{\mathcal{A},\Pi}(n)$
$m_1, m_2, \ldots, m_q \in \mathcal{M}$
$c_1, c_2, \ldots, c_q : c_i \leftarrow \mathsf{Enc}_k(m_i)$
$b \leftarrow \{0, 1\}$
$\overrightarrow{M_0} = (m_{0,1}, \ldots, m_{0,\ell})$ $\overrightarrow{M_1} = (m_{1,1}, \ldots, m_{1,\ell})$
$\vec{C} = (c_1, \ldots, c_\ell): c_i \leftarrow \mathsf{Enc}_k(m_{b,i})$
k
Gen
$m_1, m_2, \ldots, m_t \in \mathcal{M}$
$c_1, c_2, \ldots, c_t : c_i \leftarrow \mathsf{Enc}_k(m_i)$
$b' \in \{0, 1\}$
$\mathcal{A}$

❑ Experiment $Priv K^{cpa}_{\mathcal{A},\Pi}(n)$ is a special case of the experiment $Priv K^{cpa-mult}_{\mathcal{A},\Pi}(n)$ CoA - world

❖ If Π is multi-message CPA secure then Π is also single-message CPA secure

❑ If Π is single-message CPA secure then Π is also multi-message CPA secure ✓✓

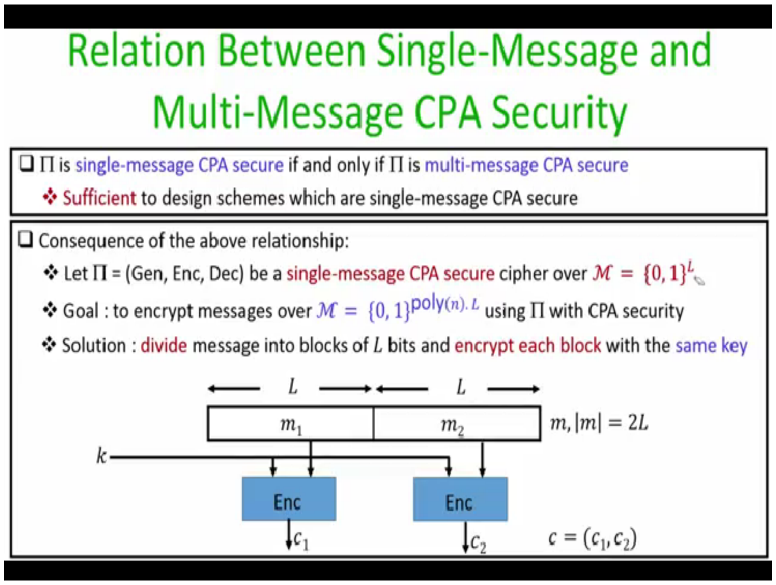
So you have here the left hand side part here is the single message CPA security. On the right hand side part of the experiment is the multi message CPA security. And now we would like to explore the relationship between these 2 notions of security. So, if you look closely into the 2 experiments, the only difference is in the challenge phase. Apart from that, you have the pre-challenge training phase, post challenge training phase in both the versions of the experiment.

And if you see the challenge phase here, you can easily see here that the single message CPA security is basically a special case of the multi message CPA security, because if in the multi message CPA security adversary submits the vectors of size 1 each, then it basically becomes an instance of the single message CPA security. So it is simple to conclude that if you have an encryption process, which has multiple message CPA secure, then of course it is also single message CPA secure. Interestingly, it turns out that in the CPA world, we can prove the other way around as well. That means if we have a candidate encryption scheme, which is single message CPA secure, then it is also multi message CPA secure. And the basic intuition for that is, since our encryption process is CPA secure it to be randomized, because the same keys used is going to be used throughout the experiment for encrypting the encryption oracle queries.

The challenge messages, challenge plain text and again, the for the post challenge training phase. That means each instance of the CPA encryption process is going to use internal randomness. So it does not matter whether the adversary submits a pair of messages or a pair of vectors, from the challenge ciphertext adversary cannot conclude what exactly has been encrypted. That is basically the intuition of the underlying proof.

I would not go through the full proof if you are interested to see the full proof, you can refer to the book by Katz and Lindel. Also notice that this implication of single message CPA security implying multi message CPA security is completely different from what we had seen in the COA world. Because in the COA world, we had seen instances where we have encryption process which are only single message COA secure. But as soon as we try to encrypt multiple messages using those encryption process, it no longer provides a security for multiple messages. That means in the COA world a single message security and multi message security are completely different, one does not imply the other, but in the CPA world the beautiful fact is that both the security notions are equivalent to each other.

(Refer Slide Time: 32:47)



Relation Between Single-Message and Multi-Message CPA Security

- ❑ Π is single-message CPA secure if and only if Π is multi-message CPA secure
  - ❖ Sufficient to design schemes which are single-message CPA secure
- ❑ Consequence of the above relationship:
  - ❖ Let Π = (Gen, Enc, Dec) be a single-message CPA secure cipher over $\mathcal{M} = \{0,1\}^L$
  - ❖ Goal : to encrypt messages over $\mathcal{M} = \{0,1\}^{\text{poly}(n) \cdot L}$ using Π with CPA security
  - ❖ Solution : divide message into blocks of L bits and encrypt each block with the same key

As a result what we can do is that we can just focus to design CPA secure schemes for fixed length messages, because once we have a CPA secure scheme for fixed length messages. Suppose for instance we have a candidate scheme, which provides you CPA secure to encrypt messages of size, L bits. Then using that encryption process as a building block, we can design an encryption process to encrypt messages of larger size say messages of size poly times L namely a message consisting of many blocks of L bits as follows.

For example, if you have a message consisting of 2 blocks of L bits what we can do is, we can divide the message into 2 individual blocks and encrypt each block independently by running an instance of the CPA secure encryption process for encrypting L bits using the same key k. And the resultant cipher text basically will be the concatenation of the 2 ciphertext block. And what this implication guarantees is that this process of dividing the larger messages into small blocks and encrypting each individual block by an instance of the scheme Π is guaranteed to provide you CPA security.

So that is why for the rest of our discussion, we will focus on designing CPA secure scheme only for fixed length messages. So that brings me to the end of this lecture, just let me summarize what we had seen in this lecture, we discussed the 2 shortcomings of any stream cipher. The first shortcoming is that we cannot encrypt multiple messages, namely, key reusability is a big issue. And the second problem is that we can encrypt messages of only fixed length.

We cannot use an encryption process which is COA secure for a fixed length message to encrypt arbitrary long messages by dividing the arbitrary long messages into small individual blocks and encrypting each block using the same key. That is not going to give you a COA security guarantee. We also introduced the notion of CPA security and we had seen some real world motivation or real world examples where adversary can actually launch a CPA attack. And we had seen that in the CPA world, single message security and multiple message security are equivalent. Thank you.