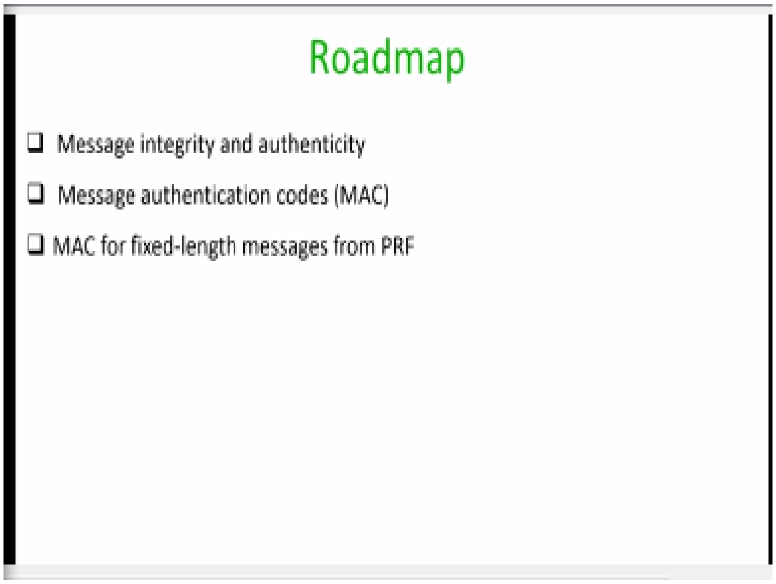
**Foundations of Cryptography**
**Prof. Dr. Ashish Choudhury**
**(Former) Infosys Foundation Career Development Chair Professor**
**Indian Institute of Technology-Bengaluru**

# Lecture-22
# Message Integrity and Authentication

Hello everyone, welcome to lecture 21, the plan for this lecture is as follows we will introduce the notion of message integrity and authenticity.

**(Refer Slide Time: 00:36)**

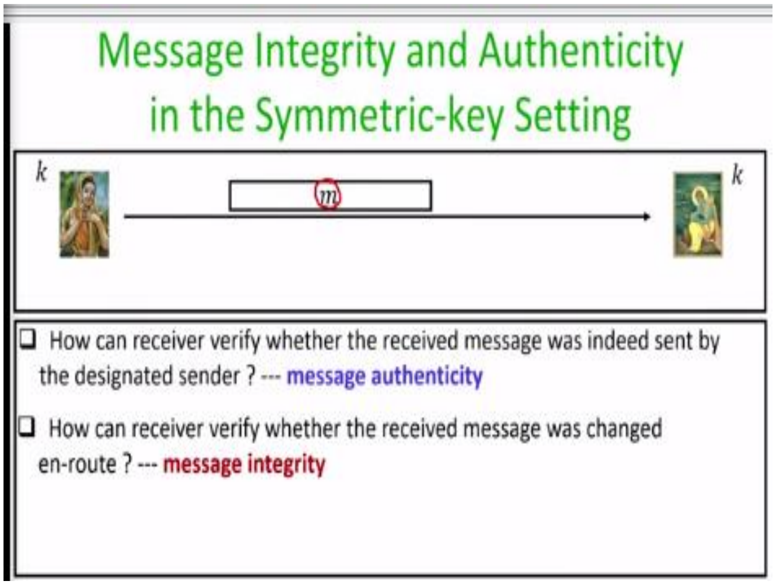


## Roadmap

- ❑ Message integrity and authenticity
- ❑ Message authentication codes (MAC)
- ❑ MAC for fixed-length messages from PRF

And we will discuss how to achieve these 2 notions using a cryptography primitive which we call us message authentication codes or MAC. And we will discuss the construction for max for fixed length messages using pseudo random functions.

**(Refer Slide Time: 00:53)**



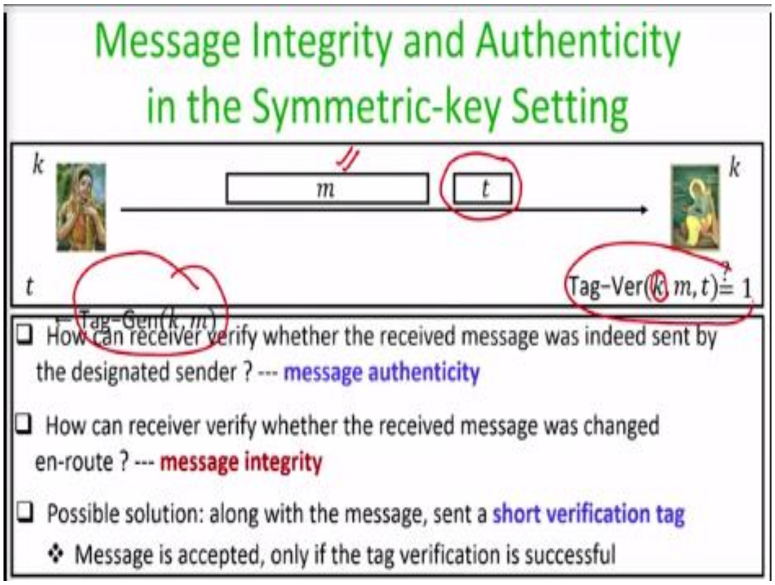
Message Integrity and Authenticity in the Symmetric-key Setting

- ❑ How can receiver verify whether the received message was indeed sent by the designated sender ? --- message authenticity
- ❑ How can receiver verify whether the received message was changed en-route ? --- message integrity

So, let us start with the definition of message integrity and authenticity in the symmetric key setting. So, the goal of the message integrity and authenticity in the symmetric key world is as follows: we assume that we have a sender and a receiver with pre shared random key known only to the sender and the receiver. And assume that there is a bit string, which sender has communicated over an insecure channel between the sender and a receiver. So, I stress here that *m* here denote just a bit string, it could be any bit string it need not be a cipher it is a just a bit string.

So, the problem of the message authenticity is as follows: when the receiver receives a bit string over this insecure channel, how can the receiver be sure that whether the received bit string was indeed sent by the designated sender? That is a problem of message authenticity. Namely, the goal here is for the receiver to verify whether the contents of the bit strings that it has received over the channel has indeed originated from the so called sender. That is a problem of message authenticity.

**(Refer Slide Time: 02:02)**



Message Integrity and Authenticity in the Symmetric-key Setting

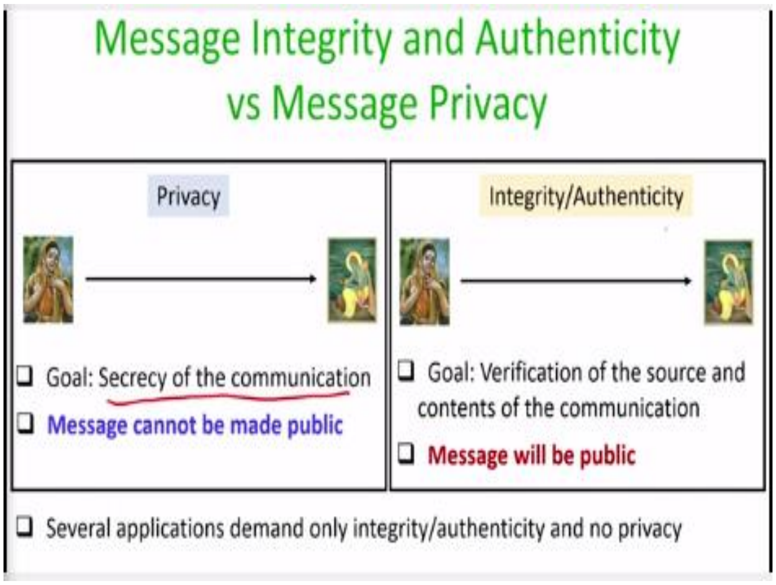
- ❑ How can receiver verify whether the received message was indeed sent by the designated sender ? --- message authenticity
- ❑ How can receiver verify whether the received message was changed en-route ? --- message integrity
- ❑ Possible solution: along with the message, sent a short verification tag
  - ❖ Message is accepted, only if the tag verification is successful

And a related problem is that of message integrity, where the goal for the receiver is to verify whether the received bit string was changed en-route or not. That means assume for the moment that receiver ensure that the received bit string has originated was sent by the designated sender. Now, it wants to verify whether that bit string which it has received was changed during it is course, during it is travel from, during it is journey from the sender to the receiver, that is a problem of message integrity.

And, I would like to stress here that there is no notion or no issue of secrecy here which we are interested to solve. We are just interested to solve the issue of authenticity and integrity here. So, potential cryptographic solution to solve the issue of authenticity and integrity is as follows: along with the bit string which sender would like to communicate to the receiver, sender can attach a short tag which is independent of the size of the message and which is computed as a function of the key and a message.

Once the tag is associated with the message when the receiver receives the message along with the tag, the receiver can run a tag verification algorithm with respect to the same k with which the sender has computed the tag. And then accordingly it can verify whether the tag verification algorithm output 0 or 1 and if it is outputs 1 that means the tag is successfully verified, then it can accept the message or else it can reject the message, so, that is the overall idea.

**(Refer Slide Time: 03:38)**



## Message Integrity and Authenticity vs Message Privacy

| Privacy | Integrity/Authenticity |
|---|---|
| ❑ Goal: Secrecy of the communication<br>❑ Message cannot be made public | ❑ Goal: Verification of the source and contents of the communication<br>❑ Message will be public |

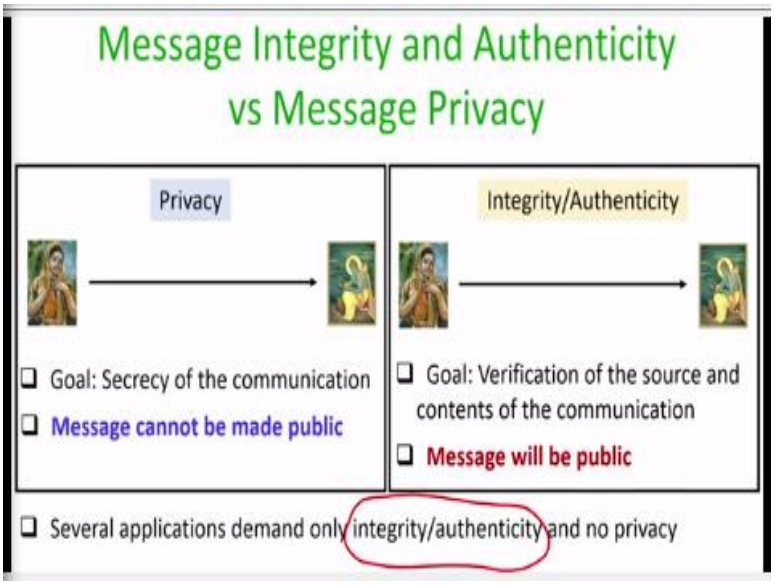
❑ Several applications demand only integrity/authenticity and no privacy

So, we will now discuss how exactly we go about designing such a tag generation algorithm and how do we design the tag verification, so that will be our next goal. But before going into that, let us discuss these 3 goals which we are trying to achieve using different cryptographic primitives. We have the problem of message integrity, message authenticity and message privacy.

So, the message in the problem of message privacy, the goal is to achieve the secrecy of the communication. Namely, whatever plain text sender is interested to communicate to the receiver. The goal of the privacy is to ensure that the message is not linked to any third party. Whereas for the integrity and authenticity, in the integrity and authenticity problem, the goal is the verification of the source and the contents of the communication.

And it is fine if the privacy of the message is lost, that means we are not interested here to maintain the privacy of the contents. So, the issue of privacy and the issue of authenticity and integrity, they are orthogonal to each other.

**(Refer Slide Time: 04:49)**



Message Integrity and Authenticity vs Message Privacy

Privacy

□ Goal: Secrecy of the communication

□ Message cannot be made public

Integrity/Authenticity

□ Goal: Verification of the source and contents of the communication

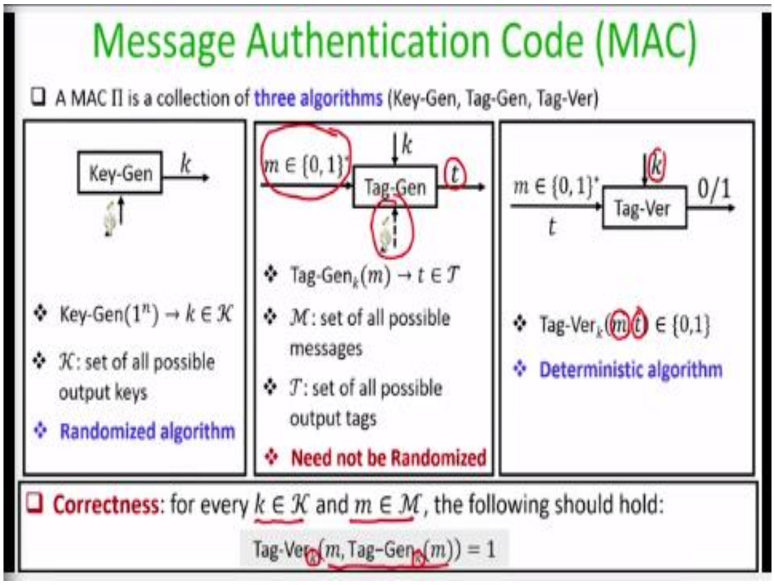
□ Message will be public

□ Several applications demand only integrity/authenticity and no privacy

And it turns out that there are several application scenarios where the requirement is only to ensure the integrity and authenticity and not the privacy. For example, if we consider an RFID application, where a user shows a smart card before entering a building then there the goal is to ensure only the integrity and authenticity. Namely, the goal has to ensure that only the legitimate users are allowed to enter the building. Whereas a user who is not supposed to enter the building should not give an access to the building.

So, like that there are several real-world applications where the goal is only to achieve integrity and authenticity and not the privacy. And that means we should now discuss how to design cryptographic primitives to solve the problems of integrity and authenticity.

**(Refer Slide Time: 05:34)**



Message Authentication Code (MAC)

❑ A MAC Π is a collection of three algorithms (Key-Gen, Tag-Gen, Tag-Ver)

- ❖ Key-Gen$(1^n) \rightarrow k \in \mathcal{K}$
- ❖ $\mathcal{K}$: set of all possible output keys
- ❖ Randomized algorithm

- ❖ Tag-Gen$_k(m) \rightarrow t \in \mathcal{T}$
- ❖ $\mathcal{M}$: set of all possible messages
- ❖ $\mathcal{T}$: set of all possible output tags
- ❖ Need not be Randomized

- ❖ Tag-Ver$_k(m, t) \in \{0,1\}$
- ❖ Deterministic algorithm

❑ Correctness: for every $k \in \mathcal{K}$ and $m \in \mathcal{M}$, the following should hold:

$$\text{Tag-Ver}_k(m, \text{Tag-Gen}_k(m)) = 1$$

And a common tool which solves both these problems is called as message authentication code. So, a message authentication code or MAC in short, is a symmetric primitive which consists of 3 algorithms. The key generation algorithms output a uniformly random key from the key space, and it has to be a randomized algorithm because if it is a deterministic algorithm, then any third party in the world will know the key.

The tag generation algorithm takes a message which sender would like to send to the receiver in an authenticated and a verifiable fashion. And it takes the key which is generated by the key generation algorithm available with the sender as well as with the receiver and it could be a potential randomized algorithm. So it could have an internal randomness, but it is not necessary that it has to be a randomized algorithm and we will discuss this fact later.

So unlike an encryption algorithm which has to be randomized to achieve any meaningful notion of secrecy, when we come to the message authentication code since our goal is not to achieve the secrecy, but rather to solve the problem of integrity and authenticity, it is not necessary that your tag generation algorithm should be randomized. So the tag generation algorithm is a function of the message to be authenticated and the key with an optional randomness.

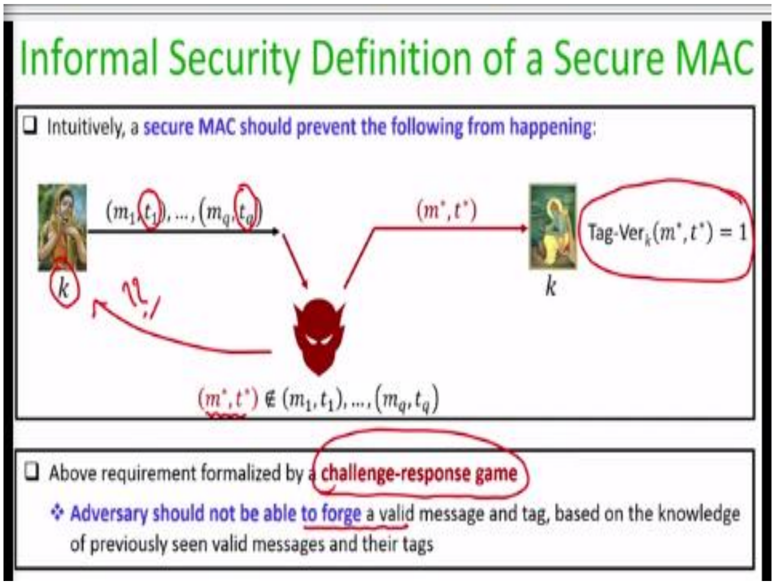
And it outputs the tag from tag space, and preferably the size of the tag should be independent of the size of the message. The tag verification algorithm takes 2 inputs namely it takes the message to be verified along with the corresponding tag and a key is used which is the same key with which should be preferably the same key with which the tag has been generated.

And a tag verification algorithm either output 0 or 1, 0 means it rejects the message that means the tag is not satisfying with respect to the message. Whereas, the output 1 means the tag verification is successful and hence accept the message. And the tag verification algorithm is always a deterministic algorithm because we want the tag verification to be unambiguous.

The correctness requirement from any message authentication code is as follows: we require that for every k which is obtained by running the key generation algorithm and any message which has been authenticated using the tag generation algorithm, the output of the tag verification algorithm with the message and the corresponding tag generated by running the tag generation algorithm where the same k has been used for the tag generation and for the tag verification, the output of the tag verification should always be successful.

So, this is analogous to the correctness requirement of the decryption process in any encryption scheme, so the correctness requirement here is straightforward.

**(Refer Slide Time: 08:22)**



## Informal Security Definition of a Secure MAC

❑ Intuitively, a secure MAC should prevent the following from happening:

$(m_1,t_1),\ldots,(m_q,t_q)$
$k$
??
$(m^*,t^*)$
$k$
Tag-Ver$_k(m^*,t^*) = 1$
$(m^*,t^*) \notin (m_1,t_1),\ldots,(m_q,t_q)$

❑ Above requirement formalized by a challenge-response game

❖ Adversary should not be able to forge a valid message and tag, based on the knowledge of previously seen valid messages and their tags

Now, let us proceed to the security definition of MAC, what exactly is the security property we require. So, before going into the formal definition, let us first try to intuitively understand what exactly we want to achieve using a secure MAC.

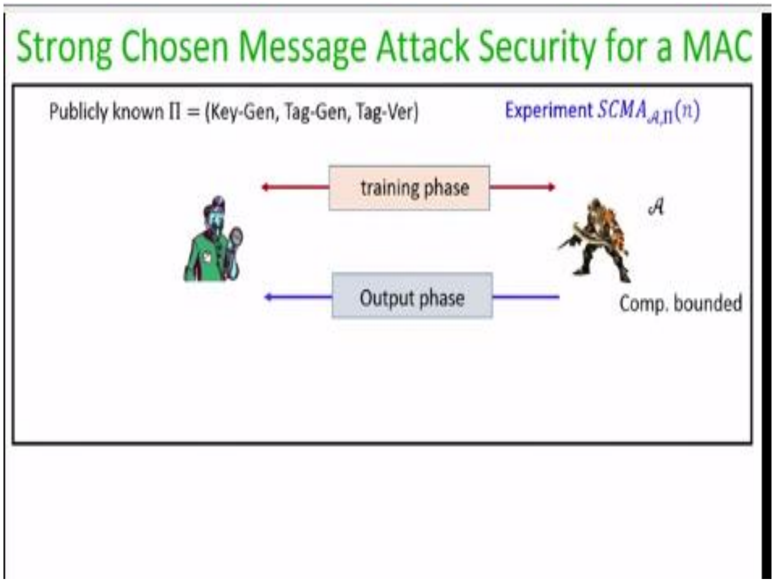
So, intuitively a secure MAC should prevent the following from happening: imagine we have a sender and a receiver who have shared a key which has been obtained by running the key generation algorithm. And say the sender has communicated several messages and it has attached the corresponding tags computed as per the tag generation algorithm with respect to the key which is known only to the sender and only to the receiver. And now we have a malicious adversary which intercepts all this (message, tag) pairs, but it does not know the value of key. The key is unknown for this attacker.

The goal of the attacker is to produce a new message from the plain text space or from the message space and the corresponding tag such that the (message, tag) pair is different from all the (message, tag) pair that it has intercepted or which has been communicated by the sender. Such that when it forwards the new (message, tag) pair to the receiver, the tag verification outputs 1, that means the tag verification is successful.

So basically, the goal of the adversary here is to create a forgery. Basically, based on the several legitimate (message, tag) pairs which it might have seen in the previous sessions, which have been exchanged between the sender and a receiver under an unknown key. The goal of the attacker is to produce a new (message, tag) pair such that that new (message, tag) pair was never communicated by the sender. And forward that new (message, tag) pair to the receiver such that when verified at the receiving end, the verification is successful.

If that is possible, then our MAC is not considered to be secure. That is a intuitive goal which we want to capture through a formal definition. And as we have done for all the cryptographic primitives in this course, this intuitive requirement is going to be captured formally through a challenge response game.

**(Refer Slide Time: 10:34)**



Strong Chosen Message Attack Security for a MAC

Publicly known Π = (Key-Gen, Tag-Gen, Tag-Ver)

Experiment $SCMA_{\mathcal{A},\Pi}(n)$

training phase

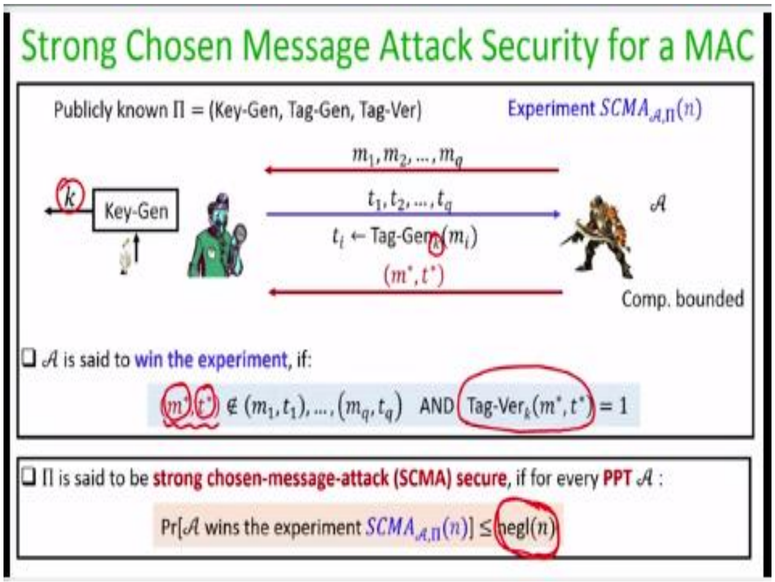
Output phase

$\mathcal{A}$

Comp. bounded

Where, we have 2 kinds of security notions, so the first experiment corresponds to a stronger security guarantee which we call us strong chosen message attack security or SCMA in short. We have a computationally bounded adversary in this experiment, and hypothetical verifier and experiment basically consist of training phase and an output phase.

**(Refer Slide Time: 10:58)**



Strong Chosen Message Attack Security for a MAC

Publicly known Π = (Key-Gen, Tag-Gen, Tag-Ver)

Experiment $SCMA_{\mathcal{A},\Pi}(n)$

$k$ Key-Gen

$m_1, m_2, \dots, m_q$

$t_1, t_2, \dots, t_q$

$t_i \leftarrow \text{Tag-Gen}_k(m_i)$

$(m^*, t^*)$

$\mathcal{A}$

Comp. bounded

□ $\mathcal{A}$ is said to **win the experiment**, if:

$(m^*, t^*) \notin (m_1, t_1), \dots, (m_q, t_q)$ AND $\text{Tag-Ver}_k(m^*, t^*) = 1$

□ Π is said to be **strong chosen-message-attack (SCMA) secure**, if for every **PPT** $\mathcal{A}$ :

Pr[$\mathcal{A}$ wins the experiment $SCMA_{\mathcal{A},\Pi}(n)$] ≤ negl($n$)

In the training phase, the adversary adaptively submits several messages of it is choice and ask for the tags on those messages from the experiment. So, this corresponds to the fact the real-world scenario where our adversary might have seen several legitimate (message, tag) pairs communicated between the sender and a receiver in the previous sessions.

So here to model that, we give the adversary the chance to train itself where we allow the adversary to submit any message of it is choice from the message space. And see the tags on those messages under an unknown key k which is chosen by the experiment of the verifier. And the response from the verifier is the tags on those messages as per the tag generation algorithm and an unknown random key k which is not known to the attacker. So the number of queries for which the adversary can ask the tag is upper bounded by some polynomial function in the security parameter.

Once the adversary is trained, the goal of the adversary is to output of forgery, namely the goal of the adversary is to output a (message, tag) pair. And we say that adversary has won the experiment, if $(m^*, t^*) \notin (m_1, t_1), \ \ldots, (m_q, t_q)$ and Tag-verf$_k$ $k(m^*, t^*) = 1$

that means that tag verification is successful.

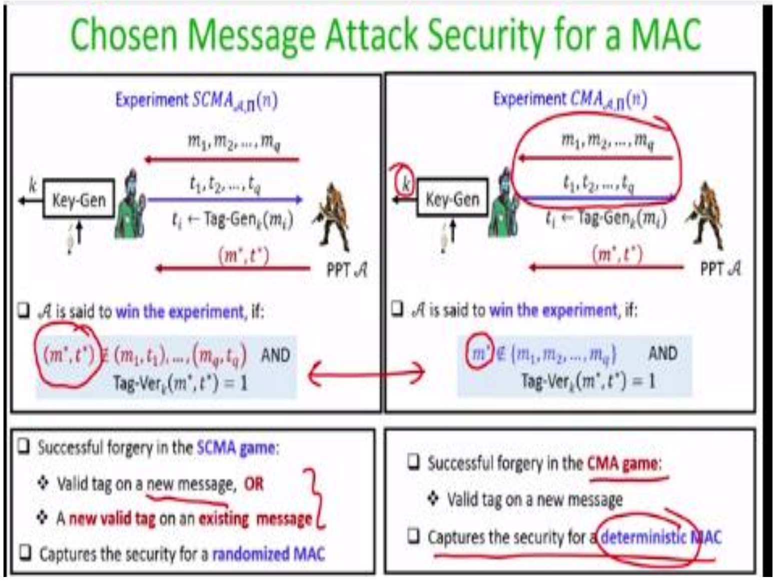
If this happens, then we say that adversary is able to win the experiment or adversary is able to produce a legitimate forgery even without knowing the value of k. And our security definition is we say that our message authentication code is strong chosen message attack secured or SCMA secure.

If for every polynomial time adversary participating in this experiment, the probability of adversary winning the experiment is Pr[*A* wins the experiment $SCMA_{\mathcal{A}, \Pi}(n)$] ≤ *negl(n)* where the probabilities over the random choice of the key generation algorithm and the queries for which the adversary has ask for the tag service.

So notice that unlike the previous security notions where we were actually requiring that success probability of that adversary should be upper bounded by half plus negligible, there is no quantity like 1/2 here. Because the goal of message authentication code is to prevent forgery not to achieve the privacy. That namely in this experiment there is nothing for which the adversary should be able to distinguish. The goal of the adversary is basically to create a forgery. And there is always a guessing attack where the adversary can just guess a random message and a random tag.

Because it knows the description of the message space and the tag space and there is always a nonzero probability that the guessed message and the guessed tag indeed constitutes a legitimate (message, tag) or a legitimate forgery. So that is why we can never demand in this security experiment that a success probability of the adversary should be 0. The maximum of the best that we can hope for is that the forgery probability of the adversary is negligible function in the security parameter.

**(Refer Slide Time: 14:13)**



Chosen Message Attack Security for a MAC

❑ $\mathcal{A}$ is said to win the experiment, if:

$(m^*, t^*) \notin (m_1, t_1), \ldots, (m_q, t_q)$ AND $\text{Tag-Ver}_k(m^*, t^*) = 1$

❑ $\mathcal{A}$ is said to win the experiment, if:

$m^* \notin \{m_1, m_2, \ldots, m_q\}$ AND $\text{Tag-Ver}_k(m^*, t^*) = 1$

❑ Successful forgery in the SCMA game:

- ❖ Valid tag on a new message, OR
- ❖ A new valid tag on an existing message

❑ Captures the security for a randomized MAC

❑ Successful forgery in the CMA game:

- ❖ Valid tag on a new message

❑ Captures the security for a deterministic MAC

So that is a definition of strong CMA security. Now, there is a related notion of security for the MAC, which we call just a CMA security. And here also the experiment basically consists of a training phase and the challenge phase wherein the training phase adversary submits several messages of it is choice. And sees the tags on those messages under an unknown key k and the goal of the attacker is to submit a forgery.

But the rule of the experiment here or the way we define the output of the experiment here is different. We say in this experiment CMA, that adversary has won the experiment if $m^* \notin \{m_1, m_2, \ \ldots, m_q\}$. So if you see these 2 notions of security, they differ in a very, very certain way.

The successful forgery in the SCMA game or the strong CMA game means that either the adversary has produced a valid tag on a new message or it has produced a new valid tag on an existing message. Because there the requirement is that the overall forgery namely the combined message and the tag when taken together, it should be different from all the previous (message, tag) pairs that the adversary has seen.

And this condition that (m*, t*) is different from all the previous ($m_i$, $t_i$) can occur in any of these 2 ways. Either the $m^* \notin \{m_1, m_2, \ldots, m_q\}$ a completely new message, or it might be possible that m* is one of the existing messages. But the tag t* is different from any of the previous tag on the same message.

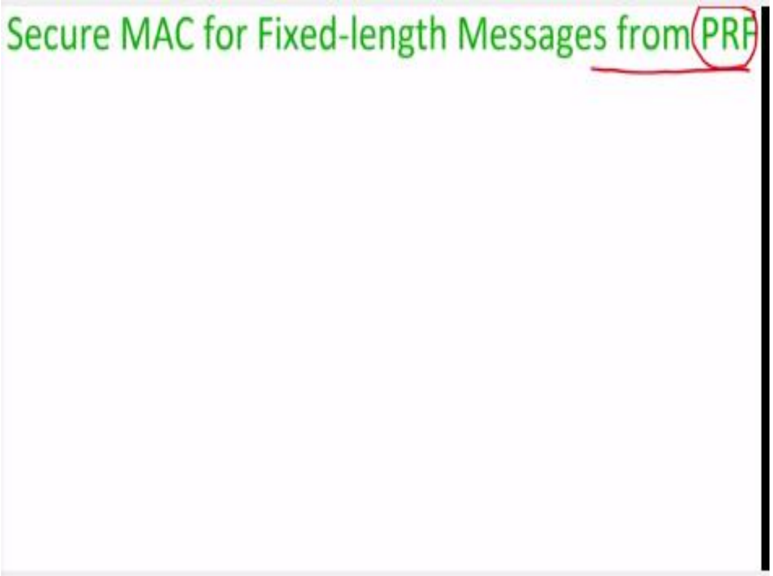
And this is possible only if your underlying tag generation algorithm is a randomize tag generation algorithm. And that is why when we discuss the syntax of the tag generation algorithm, I said it is not necessary that your tag generation algorithm should be randomized. It could be potentially randomized or it may not be a randomized algorithm. So, if your tag generation algorithm is a randomized MAC, then the strong CMA security gives you the guarantee.

That even if the adversary wants to come up with a forgery on an existing message, the forgery should be with respect to producing a new tag for the existing message. Whereas if we see the forgery in the CMA game, we say a forgery is successful only if the forgery is on a new message which is completely different from all the messages for which the adversary has seen the tag in the previous session.

This captures the security for deterministic MAC because if your tag generation algorithm is deterministic, then for every message there is a unique tag. And in that case, the forgery is possible only if the forged message is different from all the messages for which the adversary has already seen the tag. So, that implies that if our tag generation algorithm is a deterministic process, then the strong CMA security as well as CMA security are same.

Because if our tag generation algorithm is deterministic, then $(m^*, t^*) \notin (m_1, t_1),\ \ldots, (m_q, t_q)$ can happen only if $m^* \notin \{m_1, m_2,\ \ldots, m_q\}$. In that case, the strong CMA security game becomes same as the CMA game. But if we are using a deterministic message authentication code, then this notion of strong CMA and CMA are completely different.

**(Refer Slide Time: 17:49)**



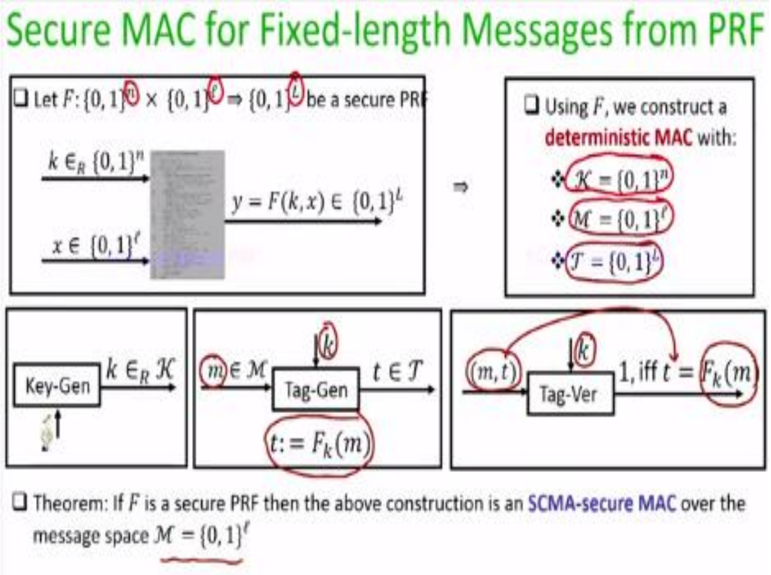
Secure MAC for Fixed-length Messages from PRF

So now we have the definition of strong CMA security and CMA security. So, let us see how we can construct a message authentication code for fixed length messages. In the next lecture, we will see that how we can construct message authentication codes for arbitrary long messages. And interestingly to compute a message authentication code for fixed length message, we will use our old symmetric key friend namely pseudo random function, and that shows the significance of pseudo random function we had already seen.

If you want to design CPA secure scheme, then you can use pseudo random functions in any mode of operation. And we have also seen that if you want to instantiate this PRF in practice, then you can always replace them with triple DES or any practically known secure block cipher.

So, this pseudo random functions are very, very significant primitive because they are not only used for designing cryptographic tools to solve the privacy problem. We are now going to see that how they can be used to solve designing message authentication codes to solve the problem of message authenticity and integrity.

**(Refer Slide Time: 18:59)**



Secure MAC for Fixed-length Messages from PRF

❑ Let $F: \{0,1\}^n \times \{0,1\}^\ell \Rightarrow \{0,1\}^L$ be a secure PRF

❑ Using $F$, we construct a deterministic MAC with:

- ❖ $\mathcal{K} = \{0,1\}^n$
- ❖ $\mathcal{M} = \{0,1\}^\ell$
- ❖ $\mathcal{T} = \{0,1\}^L$

❑ Theorem: If $F$ is a secure PRF then the above construction is an SCMA-secure MAC over the message space $\mathcal{M} = \{0,1\}^\ell$

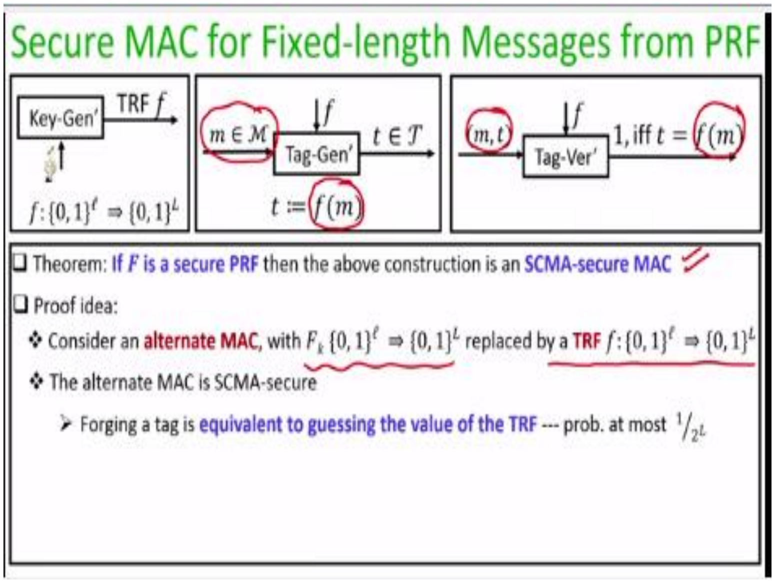
Assume you have a keyed pseudo random function with n-bit key, *l*-bit block size and then output of size L bits. It is a secure PRF as per the definition of indistinguishability-based game. And using this keyed pseudo random function, we are going to now construct a deterministic message authentication code where $\mathcal{K} = \{0,\ 1\}^n$, $\mathcal{M} = \{0,\ 1\}^\ell$ and $\mathcal{T} = \{0,\ 1\}^L$.

The idea here is very simple, the key generation algorithm of the MAC that we are constructing is going to do the following: it will output a uniformly random key for the underlying PRF. That’s the key generation algorithm. The tag generation algorithm is a deterministic algorithm where to authenticate a message m of size *l*-bits under the key k, $t := F_k(m)$.

The tag verification algorithm is simple, it takes the (message, tag) pair and the key k and it just recomputes the tag on the message m and compares it with the tag component t which it takes as the input, namely it recompute the tag of the message part under the key k and outputs $1$,iff $t = F_k(m)$.

So, in a sense what basically this tag generation and tag verification algorithm is doing is the tag is basically the output of the pseudo random function, getting messages as the input. And we can formally prove that if the underlying pseudo random function is a secure pseudo random function, then this MAC construction that we have seen is a strong CMA secure MAC, for authenticating messages of size *l*-bit strings. Why it is strong CMA secure? Because we are constructing a deterministic tag generation algorithm. And as we have argued that if our tag generation algorithm is deterministic, then the notion of CMA security and strong CMA security are equivalent.

**(Refer Slide Time: 21:30)**



Secure MAC for Fixed-length Messages from PRF

❑ Theorem: If $F$ is a secure PRF then the above construction is an SCMA-secure MAC

❑ Proof idea:

❖ Consider an alternate MAC, with $F_k \{0,1\}^{\ell} \Rightarrow \{0,1\}^{L}$ replaced by a TRF $f: \{0,1\}^{\ell} \Rightarrow \{0,1\}^{L}$

❖ The alternate MAC is SCMA-secure

➢ Forging a tag is equivalent to guessing the value of the TRF --- prob. at most $1/2^{L}$

So this is the theorem statement that we want to prove and I would not go into the full formal details of the security proof here. I will just give you an intuitive argument why exactly this theorem holds. And I leave it as an exercise for you to convert this intuitive argument into a formal reduction-based argument.

So, for the moment, you consider an alternate message authentication code where all the instances of the keyed PRF both in the tag generation process as well as in the tag verification process are replaced by an instance of a keyed TRF $f: \{0,\ 1\}^{\ell} \Rightarrow \{0,\ 1\}^{L}$. That means, in the alternate message authentication code, the key generation algorithm outputs a truly random function which will be available both with the sender and a receiver. Of course, this is an inefficient key generation algorithm.

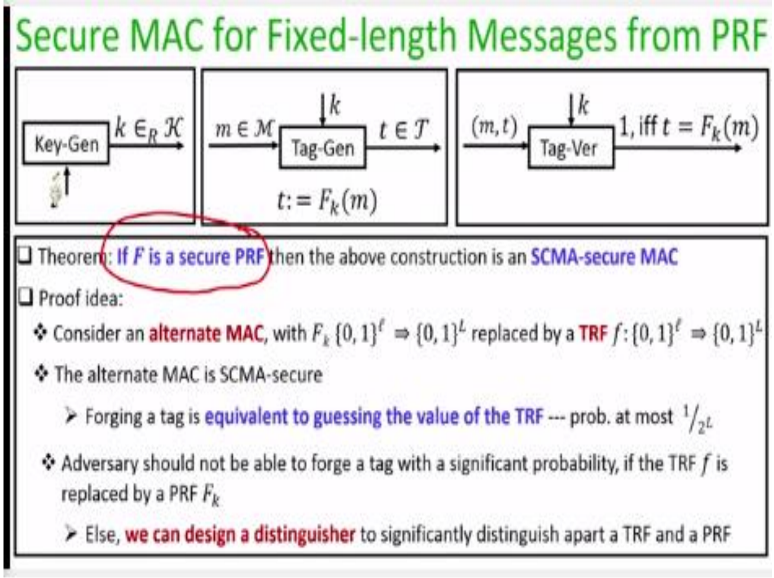
But we are not interested in the efficiency aspect here. The reason I am introducing this truly random function-based message authentication code is that it will help us to understand the underlying intuition behind a proof of this theorem, which we want to prove here. So that is a key generation algorithm for the truly random function-based message authentication code.

The tag generation algorithm, for this alternate message authentication code is as follows: to authenticate a message m, the tag basically is the value this truly random function on the message as input. And the tag verification algorithm of the alternate MAC namely is analogous to verify a (message, tag) pair. The receiver recomputes the value of the tag by evaluating the truly random function on the message part and compares it with the tag part. And if the verification is successful then the output is 1 otherwise output is 0.

Now, it is easy to see that the truly random function-based MAC is definitely strong CMA secure. The reason is that if an adversary has seen the value of the tag on several messages of its choice in the CMA game in the past. And now if it wants to forge a tag on new message basically it has to guess the value of the truly random function on the new message on which it wants to generate the forgery or create the forgery.

And the probability with which it can guess the value of the truly random function on a message m for which it has not seen the output of the truly random function is at most $1/2^L$, because f is a truly random function. So that is a simple argument based on which you can state that the truly random function based MAC is definitely strong CMA secure.

**(Refer Slide Time: 24:25)**



Secure MAC for Fixed-length Messages from PRF

Key-Gen
$k \in_R \mathcal{K}$
$m \in \mathcal{M}$
$k$
Tag-Gen
$t \in \mathcal{T}$
$t := F_k(m)$
$(m, t)$
$k$
Tag-Ver
1, iff $t = F_k(m)$

❑ Theorem: If $F$ is a secure PRF then the above construction is an SCMA-secure MAC

❑ Proof idea:

❖ Consider an alternate MAC, with $F_k \{0,1\}^\ell \Rightarrow \{0,1\}^L$ replaced by a TRF $f: \{0,1\}^\ell \Rightarrow \{0,1\}^L$

❖ The alternate MAC is SCMA-secure

➢ Forging a tag is equivalent to guessing the value of the TRF --- prob. at most $1/_{2^L}$

❖ Adversary should not be able to forge a tag with a significant probability, if the TRF $f$ is replaced by a PRF $F_k$

➢ Else, we can design a distinguisher to significantly distinguish apart a TRF and a PRF

Intuitively, the same should hold namely the strong CMA security should hold even if the TRF $f$ is replaced by a PRF $F_k$ where the key is not known to the attacker. Because that is what is the security property of a secure pseudo random function. Namely, if the adversary was not able to successfully forge a tag on a message on the truly random function based MAC because it was interacting with a truly random function, the same should hold even if the sender and receiver are using a message authentication code where in place of truly random function, a keyed pseudo random function is used.

Because, if at all it is possible now for an adversary to successfully do the forgery with a significant probability which is not negligible, that means we now know an adversary or we can design an adversary who can differentiate apart an interaction with a truly random function and an keyed pseudo random function. But that will be a contradiction to the assumption that we are assuming the F to be a secure pseudo random function. So that is a overall intuition of this construction, I am leaving the formal details of the reduction proof as an exercise for you.

So that brings me to the end of this lecture. In this lecture, we have introduced the problems of message integrity and authenticity. And we have seen 2 equivalent notions of message authentication code, security notions of message authentication code, namely, we have seen the definition of strong CMA security which holds against randomized message authentication codes.

And we have seen the security definition, CMA security definition for deterministic message authentication codes. And I stress that the goal of the message authentication code is not to solve the problem of privacy, but rather to solve the problem of integrity and authenticity where a receiver would like to verify whether it is receiving a message from a designated sender. And also it would like to verify whether the contents which it has received from the designated sender has changed en-route or not. I hope you enjoyed this lecture, thank you!