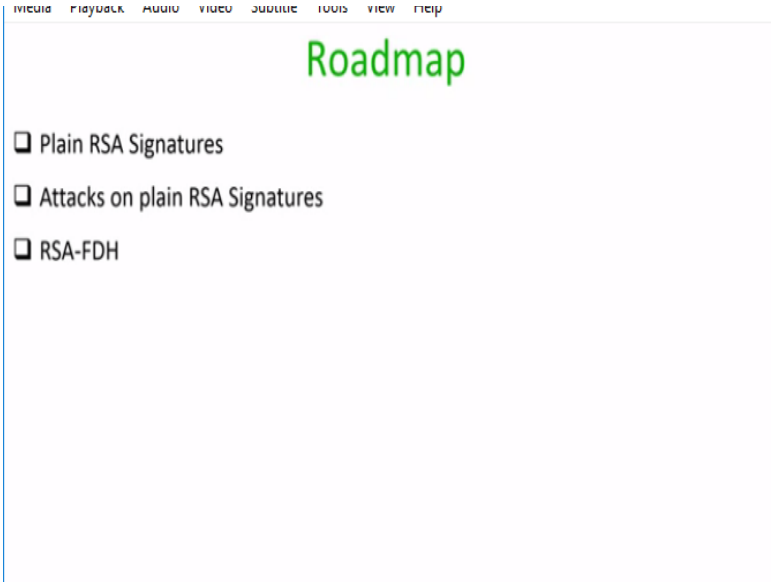
Foundations of Cryptography
Dr. Ashish Choudhury
Department of Computer Science
Indian Institute of Technology - Bangalore

Lecture – 52
RSA Signatures

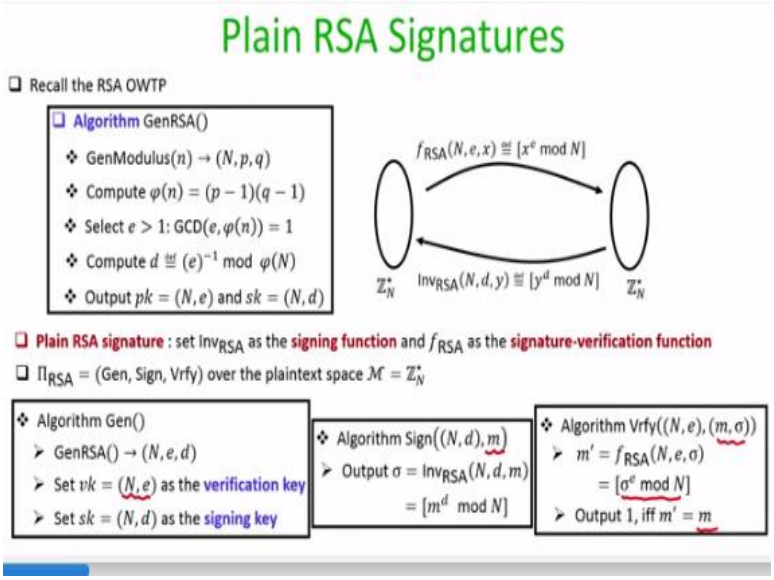
Hello everyone, welcome to this lecture. So just to recap, in the last lecture we have introduced digital signatures, we have seen their formal definition.

(Refer Slide Time: 00:40)



In this lecture we will see an instantiation of digital signatures based on RSA function. Namely, we will first see plain RSA signatures and attacks that can be launched on plain RSA signatures. And then we will see an instantiation of secure signatures based on RSA function which we call as RSA full domain hash.

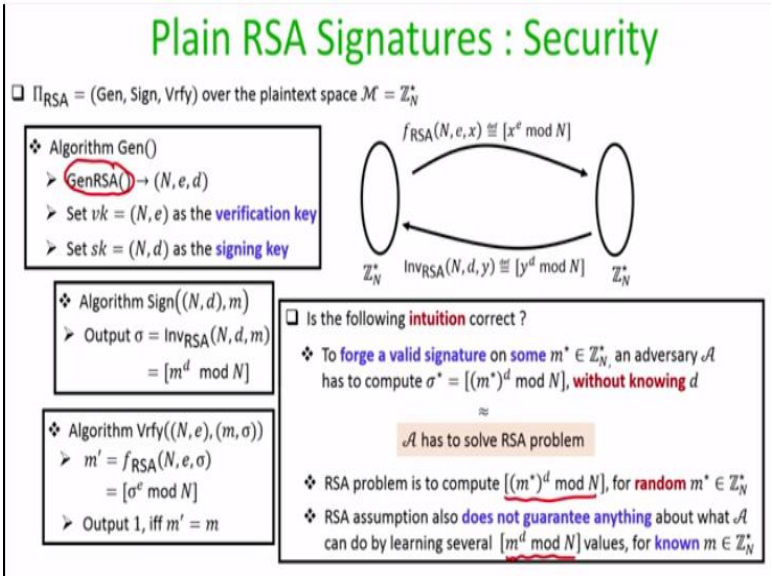
(Refer Slide Time: 00:55)



So, let us start with plain RSA signatures. And for this let me recall the RSA one way trap door permutation. We have the parameter generation algorithm, which first generates the parameters N, p, q. Here, modulus N is the product of p and q, it picks e and d, where e and d are multiplicative inverse of each other. And then we set the public parameter to be N, e and the secret parameter to be N, d. And our forward direction RSA function is $x^e \bmod N$. And our reverse function is $y^d \bmod N$. And we have proved that both these functions are inverse of each other and we also consider it to be a candidate one way function, where the trap door is d.

Now, the plain RSA signature which we can obtain from this RSA one way trap door permutation is obtained by visualising or treating this inverse function as the signing function. That means, any entity who possess N, d can use the inverse function to compute the signature, and using the forward direction function to be the verification function. More specifically, the plain RSA signature over the message space $\mathbb{Z}_N^\star$ is obtained as follows. The key generation algorithm runs the Gen RSA algorithm to obtain the parameters N, e, d and now, the public parameter N, e is set as the verification key, whereas, the trap door information namely N, d is set as the signing key. To sign a message m belonging to $\mathbb{Z}_N^\star$, we basically compute $m^d \bmod N$, where d will be available with the signer. And to verify a message, signature pair (m, σ), what we compute first is the value of the RSA function on the signature component, namely we compute $\sigma^e \bmod N$ and compare it with the received message m.

(Refer Slide Time: 02:59)



If the comparison passes, then we accept the message, namely we output 1, otherwise the output is 0. Now we want to analyse the security of this plane RSA signature scheme. And one might say that the following intuitive argument should prove that the plain RSA signature scheme is indeed unforgeable or secure. The argument here is that if I am an adversary and if I do not know the value of the signing key, namely the value of the secret d. And if my goal is to forge a signature on a message $m^\star$, on which I have not seen the signature in the past, then to forge a signature basically I have to compute the value of $(m^\star)^d \bmod N$, where I do not know the value of$d$. And one may argue that this is nothing but an instance of the RSA problem. But if I assume that my RSA problem is difficult with respect to this Gen RSA function, then I can safely claim that this plain RSA signature scheme is secure.

But it turns out that this intuitive argument is not correct. One of the reasons is that the RSA assumption or the RSA problem is difficult to solve, only if $m^\star$ is randomly chosen from $\mathbb{Z}_N^\star$. RSA assumption does not say anything how hard or how easy it is to compute $(m^\star)^d \bmod N$, without knowing the d, for any m belonging to $\mathbb{Z}_N^\star$.

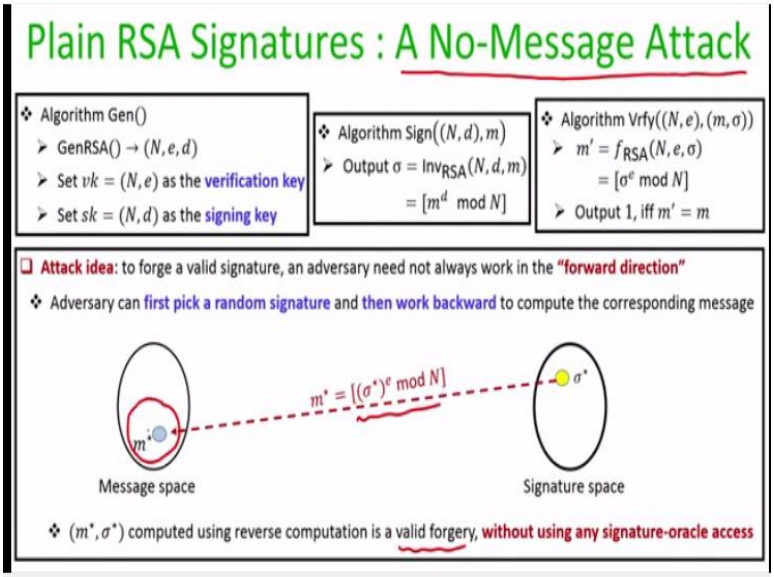
The second problem here in this intuitive argument is that the RSA assumption does not tell or does not guarantee anything about an adversary about the possibility of computing $(m^\star)^d \bmod N$, given that adversary might have seen the value of several $m^d \bmod N$ for any$m$ of its choice.

Because remember, when we consider the forgeability game for the signature scheme, adversary is allowed to see a signatures of several messages of its choice.

So, if adversary has submitted a message m, it would have seen the signature $m^d \bmod N$. And it might have seen the value of $m^d \bmod N$ for polynomial number of messages m. And by seeing polynomial number of values of the form $m^d \bmod N$, where m is known to the adversary, the goal of the adversary is to come up with a forgery, namely, $(m^\star)^d \bmod N$.

So, RSA assumption does not guarantee anything how hard or how easy it is for an adversary to come up with the forgery, given that it has seen the signature, namely the output of the RSA function for several m of its choice, in the past.

(Refer Slide Time: 05:41)



Indeed it turns out that for this plain RSA signature, there are several ways with by which the adversary could come up with a forgery. So let us see a very simple attack which we call as no message attack. From the name, it might look like that there is no message on which the adversary is producing the forgery. But that is not the case, there is indeed a message on which the adversary is producing the forged signature.

The reason why it is called no message attack will be clear to you soon. So the idea behind this attack is that to forge a valid signature, adversary need not always work in the forward direction.

What I mean by that is if we consider the message space and the signature space of this plain RSA signature, both of them are the set $\mathbb{Z}_N^\star$. And signature for a message$m$ can be produced by computing $m^d \bmod N$.

So, if the goal of the adversary is to forge a signature on a message $m^\star$, it has to basically compute $(m^\star)^d \bmod N$, that is one of the ways by which adversary can produce a signature and this I call the as producing signature by walking in the forward direction. But it turns out that adversary could come up or forge a signature by walking in the backward direction as well.

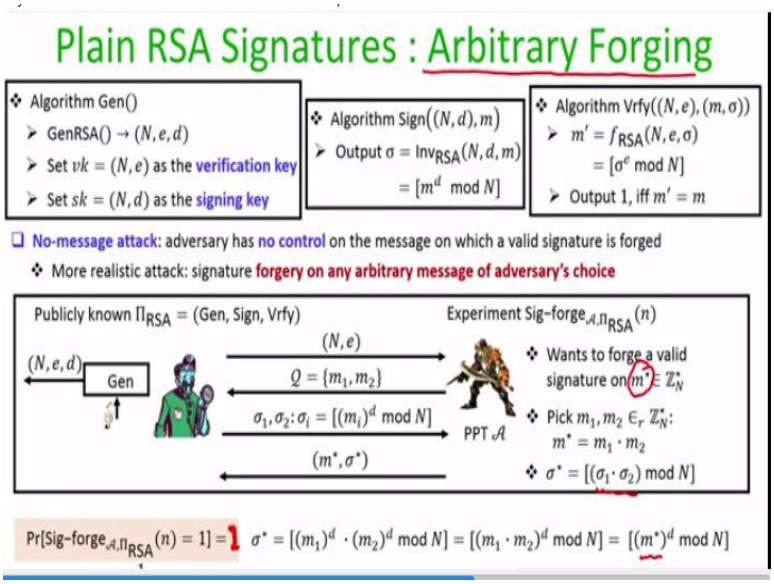
Namely, what it can do is, it can first pick up a random signature or a random group element $\sigma^\star$ and treat it as a signature. And then it can ask in its mind that what could be the valid or potential message from the message space, for which the RSA signature would have been the picked up $\sigma^\star$. And it is easy to see that by picking a random signature $\sigma^\star$, the corresponding message $m^\star$ which would have produced the signature $\sigma^\star$ is nothing, but $(\sigma^\star)^e \bmod N$.

Because if indeed my $m^\star$ is $(\sigma^\star)^e \bmod N$, then this $(m^\star\ \sigma^\star)$ constitutes a valid RSA signature as per the RSA verification algorithm. And to compute $m^\star$, adversary have everything at its disposal. Namely, it has $\sigma^\star$, it knows the value of e and it knows the value of N and hence it can easily compute $m^\star$. So, basically here it is easy to see that by walking in the reverse direction, adversary could come up with a valid forgery even without any access to the signature oracle.

It does not ask for the signature of any message of its choice; just it picks a signature and produces a corresponding message. Now, the reason we call this as a no message attack is here the forgery is produced by walking in the backward direction, the adversary does not have any message to begin with for which it wants to generate a forgery. But still as far the definition of forgery is concerned, the way adversary has computed $(m^\star\ \sigma^\star)$ it is a valid forgery.

It is a different discussion whether this $m^\star$ which adversary has produced by walking in the backward direction, indeed makes sense in the context of the application where the underlying signature scheme is used.

(Refer Slide Time: 08:58)



Now, what we are going to do next is, we are going to see a serious attack where the adversary now has a concrete message and we will see how by using the help of the signature oracle, adversary can come up with the forgery on any message of adversary's choice against this plain RSA signature.

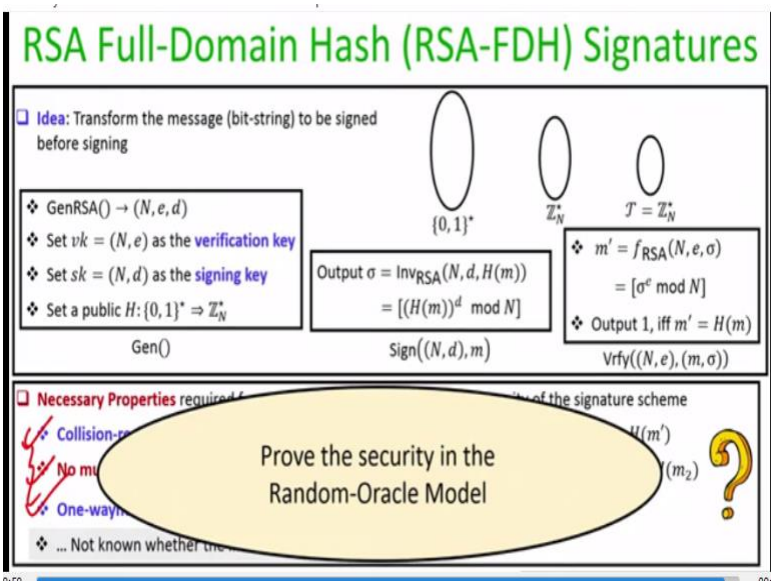
So, again to recall in the no message attack, adversary has no control on the message, which it obtained by walking in the backward direction. But a more realistic attack scenario will be where adversary has a concrete message of its choice on which it wants to forge the signer's signature. So, let me demonstrate an instantiation of the attack here. So as per the signature forgery experiment, the challenger would have thrown the challenge to the adversary, namely the verification key. And say the goal of the adversary is to forge the signature on the message$m^\star$. What it does is, it picks 2 message m_1, m_2 randomly from the set $\mathbb{Z}_N^\star$, such that $m_1 \cdot m_2 = m^\star$. And how it can pick such m_1 and m_2? Well, it can first randomly pick m_1 from the group$\mathbb{Z}_N^\star$ which requires polynomial amount of time. And then it can set $m_2 = m \cdot {m_1}^{-1}$, that will give it the required m_2, which again can be computed in polynomial amount of time.

Now, what the adversary can ask is, it can ask for the signature oracle access for the message m_1 and m_2. Namely it asks the challenger to sign the messages m_1 and m_2. So basically this models of fact that in the real world adversary's goal is to forge signer's signature on $m^\star$, basically it is now influencing the signer to sign the messages m_1 and m_2. So, as per the rules of the game, the

challenger signs the messages m_1 and m_2 and gives the signatures σ_1 and σ_2 respectively. And now that the adversary combine these 2 signatures. Namely, it sets $\sigma^\star = \sigma_1 \cdot \sigma_2$ and it submits the forgery $(m^\star, \sigma^\star)$. And my claim here is that the probability that the adversary here wins the game is 1. This is because $\sigma_1 = (m_1)^d \bmod N$ and $\sigma_2 = (m_2)^d \bmod N$ and hence $\sigma^\star = (m^\star)^d \bmod N$, which is nothing but the RSA signature on the message $m^\star$.

So, now you have a concrete attack, using the help of the signature oracle, adversary could come up with a signature on any message of its choice, which proves that this plain RSA signature is not secure.

(Refer Slide Time: 12:00)



So, now the question is can we design secure signature scheme based on the RSA function? And answer is yes, we can construct what we call as RSA full domain hash signature. And on a very high level, this might look like an instantiation of the hash and sign paradigm. But actually, it is not a full-fledged implementation or instantiation of hash and sign paradigm. The idea here is we again want to transform the message bit string which we want to sign, before signing using the RSA function.

So, what we do is the actual message space which we can sign here or the other messages which we can sign could be arbitrary bit string, we map them to the elements of $\mathbb{Z}_N^\star$ by applying some transformation function, which I call as H function, and then the actual transformed message is

signed using the RSA signature scheme that we had discussed now, the insecure RSA signature scheme.

So, the key generation algorithm for this full domain hash signature is as follows; we run the Gen RSA algorithm, set (N, d) to be the signing key and (N, e) to be the verification key. And we make the transformation function H to be publicly known, which maps arbitrary length bit strings to elements of $\mathbb{Z}_N^\star$. And now to compute the signature on the message m, which is an arbitrary length bit string, what we do is, we first compute $H(m)$. That is, we map the bit string m to an element of $\mathbb{Z}_N^\star$. And then we compute the value of inverse RSA function on the$H(m)$, namely we compute $\sigma = (m)^d \bmod N$, that is our signature. The verification happens in a canonical way. Namely if you are receiving (m, σ) and if you want to verify it, what we first do is we compute $m' = \sigma^e \bmod N$. And we accept the message m, if and only if $m' = H(m)$, which would be the case if everything has happened in a proper way.

So that is the overall idea of this RSA full domain hash. Now, one might wonder that what are the necessary properties that we require from the underlying transformation H to ensure that the overall scheme is secure. I stress here that this full domain hash signature should not be considered as an instantiation of the hash and sign paradigm.

Because for the security of the hash and sign paradigm, we need the fixed length signature scheme to be secure. But the fixed length signature scheme, which in this case is the plain RSA signature, we have already proved it is not secure. However it turns out that if we make some suitable assumptions for this underlying transformation function, then this overall way of hashing the message first, and then signing as per the insecure or the plain RSA signature gives us an overall secure signature scheme.

So, let us discuss what exactly are the security properties that we require from the transformation function? So, the list of the necessary properties is as follows. Definitely, we require that the transformation function should be collision resistant, namely, it should be computationally difficult for poly time adversary or an attacker to find out a pair of messages (m, m'), such that $H(m) = H(m')$. Because if that is the case, that means, if the adversary could come up with such a pair of

messages, then it can first ask for the signature on the message m and a signature on the message m' will be exactly the same as the signature on the message m, so it could easily come up with a forgery.

The second requirement from the transformation function is that it should avoid any kind of multiplicative or nice algebraic properties which we had exploited in the attack on the plain RSA signature. That means, it should not happen that we should have a triplet of the for (m, m_1, m_2) such that $H(m) = H(m_1) \cdot H(m_2)$. Because if it is possible for an adversary to find out such triplets in polynomial amount of time, then what adversary basically can do is, it can ask for the signature on the messages m_1 and m_2 as per this full domain hash. And given these, it can easily obtain a signature on the message m, that would be its forgery. So we would like that this transformation should be such that it should have no such nice algebraic properties.

We also need the one-wayness property from this transformation function. Namely, we require that given an arbitrary x from $\mathbb{Z}_N^\star$, it should be difficult to find a message m, such that $H(m) = x$. This is to prevent the no message kind of attack.

It turns out the definitely these 3 properties are required but we do not know whether the list is exhaustive or not, whether we need any 4th property, 5th property and so on. So, we are kind of stuck now. What we can do is, we do not care about whether the list is exhaustive or not, if we assume that the transformation function is modelled as a random oracle and if you are in the random oracle model, then we can give a complete security proof for this full domain hash signature. So, you can see that if we assume that if you are in the random oracle model, then definitely the 3 requirements that we have stated will be satisfied. And top of that the proof that the signature scheme is secure in the random oracle model takes care of the fact that no other attack can be launched. So, I am not going into the full formal details of the proof that this indeed the signature scheme is secure in the random oracle model. If you are interested to see the full formal proof, you can refer to the book by Katz-Lindell or by the manuscript by Boneh-Shoup.

So that brings me to the end of this lecture. Just to summarise, in this lecture, we have seen an instantiation of a secure signature schemes based on RSA function, we had first seen the plain

RSA signatures and we have proved that it is not secure by demonstrating 2 possible attacks. And we had seen an instantiation of secure signature schemes based on RSA function in the random oracle model. Thank you.