


Lecture - 64
Discrete Logarithm and Cryptographic Applications

(Refer Slide Time: 00:22)

Lecture Overview

- ❑ Public-key cryptography
 - ❖ Definition
- ❑ Instantiating public-key cryptosystems
 - ❖ ElGamal encryption scheme
 - ❖ RSA encryption scheme

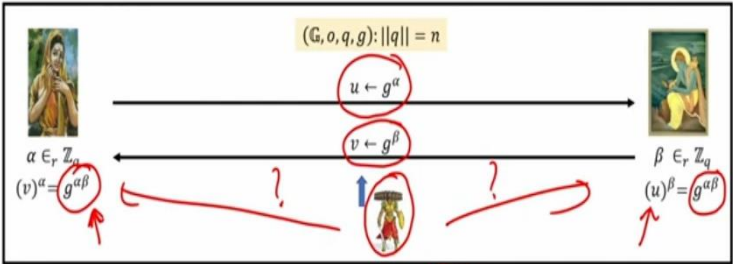


Hello, everyone, welcome to this lecture. So, in this lecture we will continue our discussion on cryptography. And we will see some more applications of the concept from number theory and abstract algebra in the context of cryptography. Namely, we will see the definition of public key cryptosystem. And we will see 2 very popular instantiations of public key cryptosystem, namely that of ElGamal encryption scheme and RSA encryption scheme.

(Refer Slide Time: 00:48)

The Birth of Public-key Cryptography

$(G, o, q, g): |q| = n$



- ❑ The DH key-exchange protocol requires both the parties to be **"online"**
 - ❖ Imagine the parties to be in different time zones
 - ❖ Hinders the spontaneity of applications like email !!
- ❑ To get around the problem, Diffie and Hellman proposed an **architecture** of a **new type of cryptosystem**
 - ❖ Today such cryptosystems are called **public-key cryptosystems**

So, let us start with the definition of public key cryptography. What exactly is public key cryptography? Why exactly we need that and so on? So, this is the Diffie Hellman key exchange protocol which allows 2 parties, Sita and Ram to talk over the internet publicly and agree upon a common key k . And if we perform all the operations over a sufficiently large group where a random instance of discrete log problem is very difficult and any third party in a reasonable amount of time will not be able to come up with the value of key that is Sita and Ram have agreed upon.

So, even though this is a very breakthrough result because before the invention of the Diffie Hellman key exchange protocol, people simply thought that it is not at all possible to solve the key agreement problem. But now Diffie and Hellman showed that, indeed it is possible to agree upon a key by talking publicly.

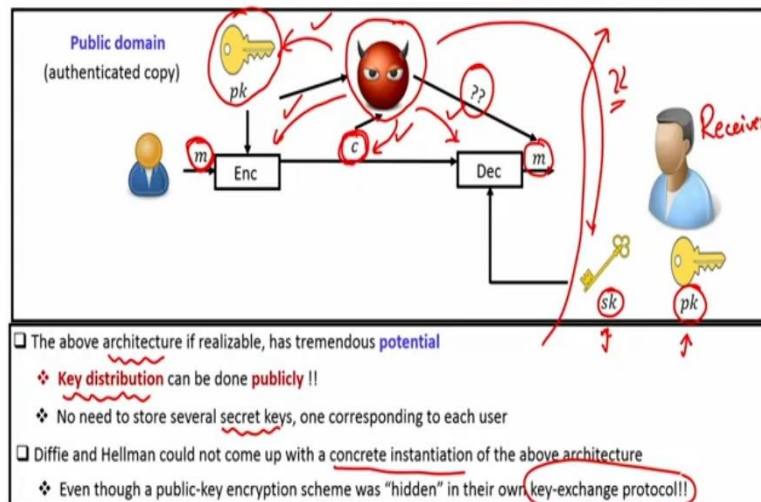
But the downside of the Diffie Hellman key exchange protocol is that it requires both the parties to be online. So, imagine the 2 parties in different time zones then it hinders the spontaneity of applications like email. So, for instance, if Sita and Rama are in 2 different time zones and Sita wants to send an encrypted mail to Ram then Sita will initiate a Diffie Hellman key exchange protocol instance with Ram, she will send her message namely g^α to Ram.

But now it might be the case that when g^α is delivered to Ram, Ram is sleeping because Ram is in a different time zone. Until and unless Ram also gets up and communicate back, g^β , Sita cannot use the key $g^{\alpha\beta}$ beta for encrypting her email. So, in that sense, the spontaneity of the application is lost here in. So that is why to get around this problem, Diffie and Hellman proposed an architecture for a new type of cryptosystem which is different from symmetric key cryptosystem.

So, remember, the symmetric key cryptosystem, the same key is used both for encryption and decryption. So, we were looking for a mechanism where the key, $g^{\alpha\beta}$, will be used for encrypting the email, as well as for decrypting the email. But Diffie and Hellman were thinking about a new form of cryptography. And today, we call such cryptosystems, such forms of algorithms, such form of cryptographic algorithms as public key cryptosystem.

(Refer Slide Time: 03:22)

The Architecture of Public-key Cryptosystem



So, let us see the architecture of public key cryptosystem. So, in this system, the receiver will have 2 keys, a key which we call us public key, pk available in the public domain. And there will be another key, sk which will be secret key and available only with the receiver. Now in this system, any person who wants to encrypt a message for this receiver will look for the copy of the public key in some public domain, say for example, a telephone directory or the homepage of the receiver.

Once the public key copy is available to the sender and sender has the plain text m , he will use the encryption algorithm and produce a cipher text or the scrambled text which is communicated to the receiver. Receiver upon receiving the scrambled text, will now use a different key, namely the secret key which is available only with the receiver and he will decrypt and recover back the message m .

And now the security property that we require here is that if there is a third party an attacker, who knows the description of the public key, who knows the description of the encryption algorithm, who knows the description of the decryption algorithm and who also knows the description of the cipher text, should not be able to figure out what exactly in the underline message. Because the secret key is not known to him. That is a loose Security property that we require here.

So, the analogy that I can give here is the following, you can imagine that receiver has created multiple copies of a padlock, all of which can be opened using a single key. And now, the public key is nothing but copies of that padlocks, but in an open state. If I am a sender and I

want to communicate some message secretly to the receiver, what I will do is, I will take, I will take one copy of that open padlock, I will take the message and keep it inside a box.

And now I will lock the box using that padlock by pressing the padlock. So that is equivalent to saying that I have encrypted my message. Now when that locked box reaches the receiver, receiver has to open the padlock and that, he can do by using the secret key which is available with the receiver. So that is the analogy and this is different from your symmetric key cryptosystem where both sender and receiver are using the same key for opening the locked box as well as for closing the locked box.

So, for the moment, just imagine that, we have a public key cryptosystem namely, we have an instantiation of public key cryptosystem. But even if I assume that we do not have an instantiation of public key cryptosystem, if at all this architecture is realizable, it has got tremendous potentials. It has got tremendous potential in the sense that, now the whole problem of key distribution is easily solved.

If I am a receiver, and if I am a amazon, for instance, I do not have to worry, who is the potential sender, he can be any entity from the world. Whoever wants to communicate with me, I just have to publish my public key for him which I can do once for all. And then anyone who wants to communicate to me has just have to use that public key, encrypt a message and communicate to me. So, in some sense, the problem of key agreement is solved.

And I do not need to have a dedicated secret key with each and every entity in this universe, I will just have a secret key and the corresponding public key can serve the role of the encryption key for every potential user with whom I want to do a secure communication. So, even though Diffie and Hellman thought about this architecture, this new system, they failed to give a concrete instantiation.


Namely, a concrete encryption algorithm, concrete decryption algorithm, a concrete mechanism of coming up with a public key and a concrete mechanism of coming up with a secret key. And the race for coming up with the first instantiation of public key cryptosystem was won by another Turing Award winner triplet namely, RSA which we will discuss very soon. But the interesting feature here, the interesting fact here is that, even though Diffie and Hellman failed to come up with a concrete instantiation of above architecture, it was hidden in

their key exchange protocol itself, it was. So, what we are going to do next is, we will again recall their key exchange protocol, the Diffie Hellman key exchange protocol and then we will see that how by doing a minor tweak, a minor modification to the key exchange protocol, we can get an instantiation of public key cryptosystem. But unfortunately, Diffie and Hellman failed to realize that.

(Refer Slide Time: 08:40)

El Gamal Encryption Scheme : Intuition

□ Recall the DH key-exchange protocol --- for simplicity, consider a **multiplicative cyclic group**



Receiver


$\alpha \in_r \mathbb{Z}_q$
 $k = (v)^\alpha$
 $= g^{\alpha\beta}$
 $m = c \circ k^{-1}$


$(\mathbb{G}, o, q, g): ||q|| = n$

$u \leftarrow g^\alpha$

$v \leftarrow g^\beta$

$c \stackrel{\text{def}}{=} m \circ k$





Sender

$\beta \in_r \mathbb{Z}_q$
 $k = (u)^\beta$
 $= g^{\alpha\beta}$
 $m \in \mathbb{G}$

□ Consider the following **encryption process**:

- ❖ Sender and receiver runs an instance of DH key-exchange protocol to obtain a **shared key** $k \in \mathbb{G}$
 - If **computing DLog is difficult** then adversary cannot compute k
- ❖ Sender can use k to **"mask"** its plaintext m --- receiver can **"unmask"** k from the ciphertext

So, this encryption scheme is called as ElGamal encryption scheme attributed to Tahir Elgamal, who made this very crucial observation regarding the Diffie Hellman key exchange protocol and what exact modification needs to be done. So, this was the Diffie Hellman key exchange protocol, Sita sends her mixture namely g^α , Ram sends his mixture g^β , where alpha and beta are individual components picked by Sita and Ram and the final key $g^{\alpha\beta}$.

Now, the whole process can be visualized as an instance of public key encryption scheme as follows. The intuition is the following. If $g^{\alpha\beta}$ is a common key which is going to be agreed upon between Sita and Ram and we know that if the discrete log problem is difficult to solve in my group, then any third party who has monitored the communication will be unable to compute $g^{\alpha\beta}$ or the key k in a reasonable amount of time, then I can use the same key k for encrypting the message.

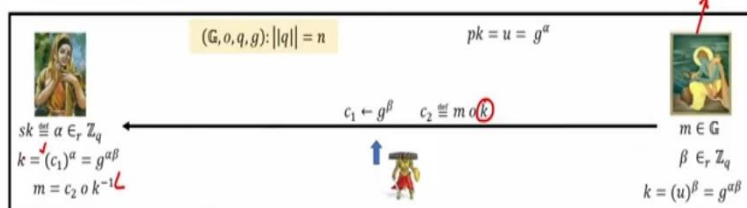
So, for instance, if Sita is the receiver and Ram is the sender and if sender is having a plain text m and again imagine that it is an element of the group g over which all the operations are performed then, what sender can do is the following. It can use the key, k namely $g^{\alpha\beta}$ for masking the message.

By masking the message I mean, here that perform the group operation between the plain text which is a member of the group and the key, k as well where key k is the element of the group as well. So that will give another group element, denote it as c. So, c will be the encryption of the message and now, how Sita can decrypt back the plain text. So, for recovering the plain text, Sita has to unmask the effect of key because the message is been masked with key.

So, if the key is unmasked, the effect of mask will go away and Sita will be able to recover back the plain text. And for doing that unmasking here, is nothing but taking the element c and performing the group operation with c and the inverse of k. Because if I perform the group operation on c and inverse of k then the effect of k and k cancels out. And what I will be left with is the plaintext m which sender wants to encrypt.


(Refer Slide Time: 11:23)

El Gamal Encryption Scheme : The Details



□ How to visualize the above encryption process as an instance of a public-key encryption ?

- ❖ Imagine **receiver's message as per public-key**
 - Receiver's contribution for the DH key-exchange protocol with every potential sender, once for all
- ❖ Imagine **sender's messages as the ciphertext**
 - **First message** : Sender's contribution for the DH key-exchange protocol
 - **Second message** : Masking of the plaintext, with the resultant DH key



So, now let us see the whole thing as an instance of a public key cryptosystem. So, this was the message which sender, so, I am treating Ram as the sender here and I am treating Sita as the receiver. So, the crucial observation of Tahir Elgamal was the following. I can imagine Sita or receiver sending our contribution for Diffie Hellman key exchange protocol once for all, for every potential sender.

So, right now what is happening is, depending upon Ram, Sita was picking alpha and contributing g power alpha. So, if there are multiple Rams, she will be picking multiple alphas, independent alphas and will be sending g^{α_1} to first Ram, g^{α_2} to the second Ram and so on. The key observation here is that, do not do that for every potential Ram, in fact, it is not even

required that who is going to be Ram, let receiver or Sita start executing her instance of Diffie Hellman protocol once for all.

Namely, whatever is her contribution for the Diffie Hellman key and what exactly I mean by contribution? Our contribution was g^α . So, you can imagine that in the Diffie Hellman key exchange protocol, there are two contributions contribution g^α coming from Sita's site, contribution g power beta coming from Ram site and both these contributions are somehow combined to get the overall key $g^{\alpha\beta}$.

So, what Elgamal proposed is that, let receiver makes her contribution, public once for all. Namely, she picks some random alpha as her secret key and makes g^α available in the public domain as her public key. So, it is as good as saying that, she is declaring publicly that, I would not be again and again participating in different instances of Diffie Hellman key exchange protocol, whoever is going to be Ram, just to think as if I am going to send g^α to you, if I would have participated in the Diffie Hellman key exchange protocol.

That is the way public key and secret key will be picked by our receiver. Now, imagine there is a sender, Ram who has a plain text m , he wants to encrypt the plaintext m . How he can do that? Ram will now do his part of the Diffie Hellman key exchange protocol, namely, he will give his contribution which is g power beta.

And now Ram knows that once he sends g^β to Sita, using g^α which Sita anyhow has made public and treating g^β as a message coming from Ram, Sita will be able to compute the key, $g^{\alpha\beta}$. So, what Ram can do is, once he has sent g^β , he can use k namely, $g^{\alpha\beta}$ for encrypting the message.

And the overall encryption of the plain text will be now two messages. The first message will be Rams contribution for the Diffie Hellman key exchange protocol. And the second message is the actual encryption of the message. How Sita will be doing the decryption? So, this is the encryption process, sending his contribution of Diffie Hellman key exchange protocol and then masking of the message this whole thing can be visualized as encryption of the plain text.

The decryption happens as follows, Sita computes the key k , assuming that Ram has participated in an instance of Diffie Hellman key exchange protocol. So, she will be able to compute $g^{\alpha\beta}$. So, for that, she has to take her secret key and raised that secret key, she has to take the first component of the cipher text, namely c_1 and raise it to her secret key which will give her the common key $g^{\alpha\beta}$.

And now she can unmask it by taking the second component of the cipher text. And performing the group operation with k inverse where, k inverse she has computed in the previous step. So, that will be the decryption process for Sita. A very cool observation which unfortunately Diffie and Hellman missed. And that is why Taher Elgamal got whole credit of inventing this cryptosystem.

And now, why this whole process, whole mechanism will be a secure mechanism? So, imagine there is a third party or Ravana, will he be able to find anything about the message m in the reasonable amount of time? Well, the only way he can learn anything about the message m is by learning the key k . But for learning the key k , he has to actually attack the Diffie Hellman key exchange protocol or he has to solve instances of discrete log problem. So, assuming that solving random instances of discrete log problem is difficult, this whole process is indeed an instance of public key cryptosystem.

(Refer Slide Time: 16:38)

The Group \mathbb{Z}_N^* and Euler's Totient Function

<ul style="list-style-type: none"> □ \mathbb{Z}_N^*: set of integers modulo N, coprime to N ❖ Ex: $\mathbb{Z}_{10}^* = \{1, 3, 7, 9\}$ □ $\mathbb{Z}_N^* \cong \{b \in \{1, \dots, N-1\} : \gcd(b, N) = 1\}$ ❖ If the modulus N is a prime, then $\mathbb{Z}_N^* = \{1, \dots, N-1\}$ □ Theorem: \mathbb{Z}_N^*, \cdot_N constitutes a group
<ul style="list-style-type: none"> □ $\varphi(N) \cong \mathbb{Z}_N^*$ ❖ Number of elements in the set $\{1, \dots, N-1\}$, which are coprime to N --- order of the group \mathbb{Z}_N^* □ If N is a prime p, then $\varphi(p) = p-1$ $N = pq$ □ If N is the product of distinct primes p and q, then $\varphi(N) = (p-1)(q-1)$ ❖ Can be proved using principle of inclusion-exclusion □ Ex: $N = 10 = 2 \cdot 5$, so $p = 2$ and $q = 5$ ❖ $\mathbb{Z}_{10}^* = \{1, 3, 7, 9\}$, so $\varphi(10) = 4 = (2-1)(5-1)$ <div style="text-align: right; margin-top: 10px;"> $a^2 = \underbrace{a}_{\substack{\varphi(p) \\ \downarrow \\ 1}} \underbrace{a}_{\substack{\varphi(q) \\ \downarrow \\ 1}} \dots \underbrace{a}_{2 \bmod \varphi(N)}$ </div>
<ul style="list-style-type: none"> □ Theorem: for any $a \in \mathbb{Z}_N^*$, we have $(a^{\varphi(N)} \bmod N) = 1$ // The order of \mathbb{Z}_N^* is $\varphi(N)$ and 1 is the identity element of the group \mathbb{Z}_N^* ❖ for any $a \in \mathbb{Z}_N^*$, we have $(a^x \bmod N) = (a^{x \bmod \varphi(N)} \bmod N)$

So, now, as I said earlier that, race for coming up with a first instantiation of public key cryptosystem was won by another Turing Award winner triplet, namely RSA, Rivest, Shamir and Adleman. Now, let me give you briefly a description of the RSA public key cryptosystem,

again which is based on several interesting results from number theory that we have discussed. So, let me recall the group Z_{N^*} and some concepts related to the group Z_{N^*} .

So, recall the definition of the Z_{N^*} is the collection of all the values in the range 1 to $N - 1$ which are co prime to your modulus N . So, for instance, the set Z_{10^*} will have the elements 1, 3, 7, 9, it would not have the element 2 because 2 is not co prime to 10. It would not have the element 4, it would not have the element 5, 6, 8 because all of them are not co prime to 10. It turns out that if your modulus N is a prime then, Z_{N^*} is nothing but a set 1 to $N - 1$.

And we already proved in one of our earlier lectures that, the group Z_{N^*} along with operation multiplication modulo N constitutes a group. So, the order of this group Z_{N^*} namely the number of elements in the set 1 to $N - 1$ which are co prime to N which is also the order of the group Z_{N^*} is called as the Euler totient function, denoted by this $\phi(N)$ function. And there are formulas for calculating the size or the order of the group Z_{N^*} depending upon the value of N .

So, the interesting cases are the following, if N is a prime number then the order of the corresponding group ϕp^* is $p - 1$ or equivalently they are $p - 1$ elements in the range 0 to $p - 1$ which are co prime to p . Whereas, the case which we will be using in the context of RSA cryptosystem is N is the product of distinct prime numbers p and q . So, if N is the product of distinct prime numbers p and q then the size of the group Z_{N^*} is being product of $p - 1$ and $q - 1$.

And this can be proved using the principle of mutual inclusion exclusion that we had discussed in one of our earlier modules. So, if you want to verify this, consider $N = 10$ which is the product of 2 and 5. And then, we know that there are 4 elements in Z_{10^*} namely, the order of Z_{10^*} is 4, so, I should get ϕ of 10 is 4 and indeed, ϕ of 10 is 4 because it is $2 - 1$ multiplied with $5 - 1$.

And we know that, if I take any element a in the group Z_{N^*} then, $a^{\phi(N)} \bmod N$ is 1. And this can be proved in multiple ways, I can use the following result from abstract algebra. I know that the order of the group Z_{N^*} is $\phi(N)$ because that is what is the definition of $\phi(N)$ and element 1, the numeric element, the numerical 1 is actually the identity element of this group Z_{N^*} .

And I know that you take any group element, irrespective of what is the order of that group element a to the power order of the group, will always give you the identity element. So, p of N is the order of the group, a is an element, so, it does not matter what is the order of a , irrespective of whatever is the order of a , I know a raised to the power order of the group is 1 and a raised to the power order is nothing but $a^{\phi(N)}$.

Based on this theorem, I can say the following. If I want to compute $a^x \text{ mod } N$ then, in the exponent I can perform modulo $\phi(N)$ because I can rewrite a power x as several blocks of a power $\phi(N)$ a power $\phi(N)$ and like that and the last block consisting of a power x modulo $\phi(N)$. Each of these blocks with a power $\phi(N)$ will give me the identity element 1, 1, 1 and I will be left only with the last block which has a power x modulo $\phi(N)$.

(Refer Slide Time: 21:24)

RSA Function

\mathbb{Z}_N^* $\xrightarrow{f_e(x)}$ \mathbb{Z}_N^*
 \mathbb{Z}_N^* $\xrightarrow{f_d(y)}$ \mathbb{Z}_N^*

- $e > 2$, such that $\text{GCD}(e, \phi(N)) = 1$
- ❖ $d \equiv (e)^{-1} \text{ mod } \phi(N)$
- ❖ $ed \text{ mod } \phi(N) = (de \text{ mod } \phi(N)) = 1$
- $f_e(x): \mathbb{Z}_N^* \rightarrow \mathbb{Z}_N^*$ $x \in \mathbb{Z}_N^*$
- ❖ $f_e(x) \equiv x^e \text{ mod } N$ $\phi(N) = 1$
- $f_d(y): \mathbb{Z}_N^* \rightarrow \mathbb{Z}_N^*$ $x \leq N$
- ❖ $f_d(y) \equiv y^d \text{ mod } N$

Claim: The function $f_d(y)$ is the **inverse** of the function $f_e(x)$

- ❖ Consider an **arbitrary** $x \in \mathbb{Z}_N^*$ and $y = f_e(x) = (x^e \text{ mod } N)$
- ❖ $f_d(y) = (y^d \text{ mod } N) = ((x^e \text{ mod } N)^d \text{ mod } N) = (x^{ed} \text{ mod } N) = (x \text{ mod } N) \equiv x$

So, now let me introduce RSA function which forms the basis of RSA public key cryptosystem. This is a function from the set $Z_N^* \rightarrow Z_N^*$. Then, how exactly is this function defined? So, imagine you have a public exponent e , this is not identity element, this is some notation, this is an exponent e which is we are going to use in the function. And this exponent e is relatively prime to $\phi(N)$, I stress it is not relatively prime to N , it is relatively prime to $\phi(N)$.

Now, since e is co prime to $\phi(N)$, it will have a multiplicative inverse. I call that multiplicative inverse as d , so, since e and d are multiplicative inverse of each other, this relationship hold. Now, the RSA function in the forward direction is the following. If I want to compute the output of the RSA function for x then that is same as computing $x^e \text{ mod } N$. So, remember x is a member of Z_N^* and hence $x^e \text{ mod } N$ will also be an element of Z_N^* .

Because my underlying operation is multiplication modulo N and x power e is like performing the group exponentiation. Whereas, my reverse function from $Z_{N^*} \rightarrow Z_{N^*}$ to will be the following. If I have a value y and if I want to invert it, I compute $y^d \pmod N$. So, I can prove that the function f_d in the reverse direction is actually the inverse of the function f_e in the forward direction.

And the way we can prove it is as follows. So, you take any arbitrary x and suppose, for that arbitrary x , the forward direction function gives you the output y . So, $y = x^e \pmod N$. What I have to show is that, now if I invert this y , namely $x^e \pmod N$ as per the inverse function, I should get back my x . So, let us do that. So, let us try to invert the value of y .

So, let me write down the value of y , y is nothing but $x^e \pmod N$ and then whole thing raised to the power d then, I can apply the rules of group exponentiation and say that this is nothing but $x^{ed} \pmod N$. And remember that $x^{\phi(N)}$ is 1 because x is an element of Z_{N^*} . That is what we discussed in the previous slide. So, $x^{ed} \pmod N$ will give the same answer as if x to the power in the exponent, I do $ed \pmod \phi(N)$ that means, I can reduce the exponent itself modulo $\phi(N)$.

But $ed \pmod \phi(N)$ is 1 that means, this $ed \pmod \phi(N)$ is 1, so, this is nothing but $x^1 \pmod N$ and which is strictly x because if x was a member of Z_{N^*} that means, x was strictly less than N . So, if x is strictly less than N then, the effect of mod would not take an x modulo N will be same as x . So that shows that these functions f_e and f_d they are inverse of each other. So, if you go to the forward direction through f_e , you can always come back in the reverse direction through f_d .

(Refer Slide Time: 24:57)

RSA Problem

Algorithm GenRSA(n) → parameter generation

- Generate **random** n -bit prime numbers p and q
- Compute **modulus** $N = pq$
- Compute $\phi(n) = (p-1)(q-1)$
- Select $e > 2$, such that $\text{GCD}(e, \phi(n)) = 1$
- Compute $d \equiv (e)^{-1} \pmod{\phi(N)}$
- Output (N, p, q, e, d)

RSA Problem

Given: only (N, e) generated by GenRSA(n) and a random $y \in \mathbb{Z}_N^*$

Goal: compute $(y^e \pmod N)^{\frac{1}{e}}$ in $O(\text{poly}(n))$ time --- e^{th} -root of y modulo N

Note that $(y^e \pmod N)^{\frac{1}{e}} = (y^d \pmod N)$

$(y^d \pmod N)$ can be computed by **factorizing** N and computing $\phi(n), d$

Factorizing **believed** to be a **hard problem**

No efficient algorithm known, if p and q are sufficiently large (say 512 bits)

Handwritten notes: $de = 1$, $d = \frac{1}{e}$, $y^d \pmod N \approx y^{1/e}$

Now, based on all these things, let us introduce a computational problem which we believe is really difficult to solve. It is like your discrete log problem. So, we know that there are certain groups where solving discrete log instance is really difficult. In the same way RSA introduced a computational problem which we believe to be difficult to solve. Difficult to solve in the sense, in the reasonable amount of time, a practical amount of time, we may not be able to solve it.

I am not saying it is impossible to solve it, you can always solve it by doing a brute force, but the brute force algorithm will take enormously large amount of time. If we operate on very large numbers. So, the problem instance is as follows. So, we first define what we call RSA parameter generation algorithm. So, this is parameter generation. So, to generate the parameters, we randomly pick some n -bit prime numbers p and q .

And then we compute the modulus which is the product of p and q , we compute a value of $\phi(N)$. Since, N is the product of prime numbers p and q which are distinct, by the way, we will ensure that p and q are distinct. So, p and q are distinct, the value of $\phi(N) = (p-1) \cdot (q-1)$. We will pick an exponent e which is co prime to $\phi(N)$. And since e is co prime to $\phi(N)$, we will be able to compute its multiplicative inverse modulo $\phi(N)$ by running extended Euclid's algorithm.

And finally, the output of this parameter generation algorithm is the modulus, the prime factors of the modulus, the public exponent e and the secret exponent d . What do I mean by public and secret it will be clear soon. So, the RSA problem is the following. If I give you the modulus,

but not its prime factors and if I give you the public exponent in that sense, it is public, it will be known to you and it will be known that how exactly this parameters N and e are generated. What would not be known to you are the prime factors of N .

And if the prime factors of N are not known, you would not be knowing the value of $\phi(N)$ and the value of $\phi(N)$ is not known to you, you would not be knowing the value of d . So, d is not known, $\phi(N)$ is not known, p and q are not known. Now, the problem instance is the following. I will be giving you a random element from my group Z_N^* and your goal will be to compute the inverse function, output of the inverse function that we had just seen, for the randomly chosen y .

Namely, your goal will be to compute $y^d \bmod N$ where d is not given to you and $y^d \bmod N$ is nothing but computing y raised to power 1 over e where 1 over e is not numeric 1 over e but it is actually the multiplicative inverse of d , so, multiplicative inverse of d is nothing but, so, d and e , they are multiplicative inverse of each other. So, computing d is nothing but computing $y^{1/e}$.

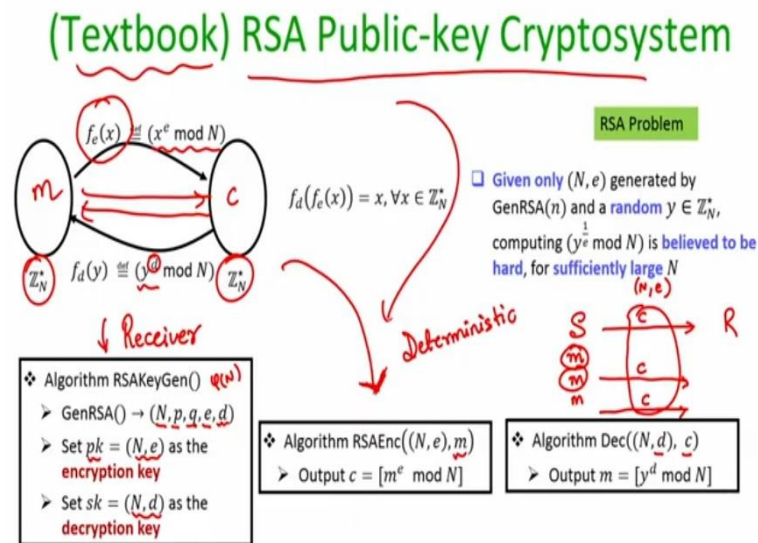
So, basically I am asking you to compute the e th root of $y \bmod N$ and I want you to solve this problem instance in polynomial of n number of time where n is the number of bits that I used to represent p and q that means, my p and q are n -bit numbers. So, one way of solving this problem instance is that you are able to factorize N . So, suppose you are able to factorize N in polynomial amount of time namely, you are able to compute p and q in polynomial amount of time.

Once you are able to compute p and q you will be able to compute $\phi(N)$ easily. And since you know e and if you know $\phi(N)$, you yourself can run the extended Euclid's algorithm and compute d in polynomial amount of time and then you yourself can compute y^d . But it turns out that factoring very large numbers is believed to be an extremely hard problem, specifically, with the current computing power.

If I select my p and q to be as large as say, 512 bit prime numbers then my N will be an extremely large modulus then factorizing that extremely large modulus will be very time

consuming process and hence you would not be able to solve and random instance of the RSA problem.

(Refer Slide Time: 29:48)



So, now based on whatever theory we have discussed, let us see the concrete steps of the RSA public key cryptosystem. So, remember, we have a function in the forward direction from $\mathbb{Z}_N^* \rightarrow \mathbb{Z}_N^*$ namely, x power e modulo N and a reverse function is y power d modulo N . And we also have discussed RSA problem where if I do not give you the value of the secret exponent d then, computing $y^d \pmod N$ is very time consuming that is what is the general belief.

Now, the way RSA cryptosystem works is as follows. So, remember, there is a sender and a receiver. So, what receiver will do is the following. Receiver will run the parameter generation algorithm. Namely, it will pick a random prime number p , a random prime number q , will compute its product, will pick an index e which is relatively prime to $\phi(N)$. So, he can compute $\phi(N)$ because he himself has picked p and q .

And once he has picked e , he knows $\phi(N)$, he can compute d as well. And then he will set pk or the encryption key to be (N, e) and he will set as the decryption key to (N, d) . So, d is kept with himself, e is made public and N is also made public. If there is a sender, who has a plain text m and which it wants to encrypt then, encryption of m is nothing but computing the forward direction function as per the RSA function, namely, just output $m^e \pmod N$.

And if there is a receiver who obtains the cipher text c and who has the secret decryption key d then to get the cipher text, tend to recover back the message encrypted in c , he has to basically

compute the inverse function for the c , inverse function is computable, if d is available and if d is available with the receiver, he can easily compute $y^d \bmod N$ and get back the message. Now, why this is called as a textbook cryptosystem? Because this is not precisely the way we use RSA public key cryptosystem in practice.

There are lots of shortcomings. One very important one, a very bad feature of the way RSA cryptosystem is proposed is given here and if I use here, the major shortcoming here is that, it is deterministic. It is deterministic in the sense that, if there is a sender S and suppose, he wants to send the same message m after every one hour. Then, if you encrypt the same message m using the public key (N, e) , every time he will be producing the same c . So, the first time he wants to encrypt a message m , he will send c .

Next time he wants to send a message m using the same key, he will be again sending the c , again next time he wants to send the same message m , he will be sending the c , this itself is a lot of information for the third party or the attacker. He may not be able to learn the exact value of the message m . But he will be coming to know that actually it is the same message m which has been encrypted and communicated to the receiver.

And depending upon my underlying application, this itself can be a breach of security. Ideally, I am looking for a process where, even if the same message m is encrypted using the same public key multiple times, it should produce different cipher texts with high probability. But that is not the feature available with the way RSA public key cryptosystem was invented. But we can of course get rid of this shortcoming and actual way in which we used RSA public key cryptosystem is different from the way it is proposed here.

But this forms the basis of the RSA public key cryptosystem, a very interesting public key cryptosystem. So, with that, I conclude today's lecture. Just to summarize, in this lecture, we discussed about public key cryptosystem and we discussed two popular instantiations of public key cryptosystem namely, we have discussed ElGamal encryption scheme and we have discussed RSA public key cryptosystem. Thank you!