Ethical Hacking
Prof. Indranil Sengupta
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture - 33
Digital Signature and Certificate

In this lecture, we shall be exploding yet another application of hash functions which is Digital Signatures and Certificates. Now, digital signature is something which you must have heard of, which is being used nowadays in many online transactions and you will see that underlying any scheme for digital signature, you need to use some kind of a hash function. Let us try to understand the process here, ok.

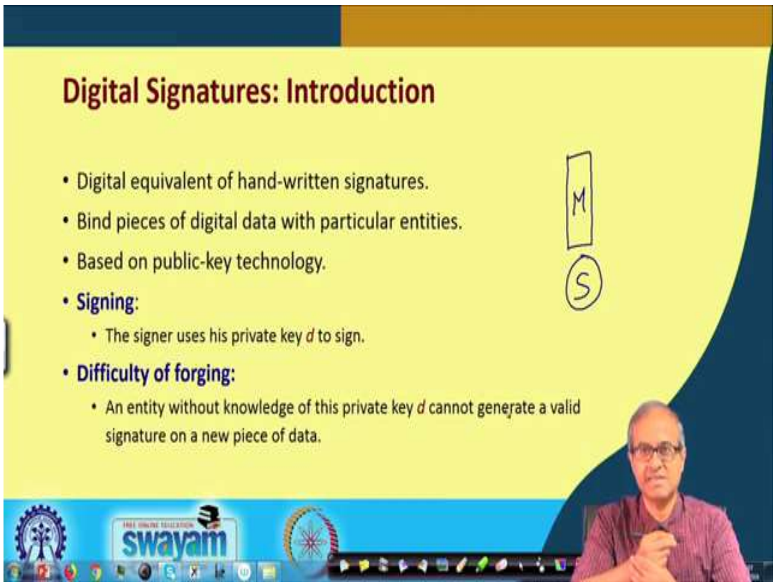
(Refer Slide Time: 00:44)

CONCEPTS COVERED

- ❑ Digital signatures
- ❑ Digital certificates

Now, in this lecture we shall be looking at two things, digital signatures and digital certificates. Well, digital certificate is nothing but some kind of a standardization. Standardization, so that people across the world can use digital signatures to, I mean authenticate documents and themselves with each other, ok.

(Refer Slide Time: 01:14)


Digital Signatures: Introduction

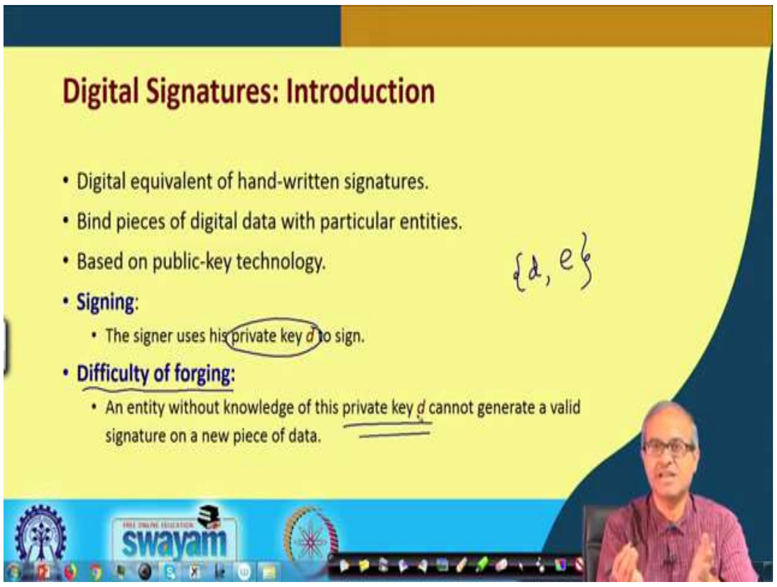
- Digital equivalent of hand-written signatures.
- Bind pieces of digital data with particular entities.
- Based on public-key technology.
- **Signing**:
  - The signer uses his private key *d* to sign.
- **Difficulty of forging**:
  - An entity without knowledge of this private key *d* cannot generate a valid signature on a new piece of data.

Talking about digital signatures, let us try to understand what it is. Well, you know what is mean by pen signatures. Say the letter we sign, because the signature of every person is unique that is the underlying assumption. We assume that it is unforgivable. It cannot be forged. That is so called handwritten signature.

Now, in digital signature you can say it is the digital equivalent of handwritten signature. I have a message, now it is not in the form of a hard copy or printed thing. It is in the form of a file. There should be an equivalent digital process when someone, some individual can put a digital signature. Digital signature will be something similar to hash digest. Like, I have a message M, I am saying that I will be appending a signature S to it, something similar to hash digest, so that if the receiver or any other person can verify that it is indeed my signature, if I have signed it, ok.

Now, digital signature technology is based on private key technology, because of the nice property of public key algorithms like RSA you have seen. So, if I sign with my private key, then anyone in the world can decode it using my public key that is the good thing. So, for this signing, for generating the signature what we need is the person who is signing. Suppose I am signing, I must be using my private key. So, you recall every individual will be having a private key and a public key, ok.

(Refer Slide Time: 03:08)



Digital Signatures: Introduction

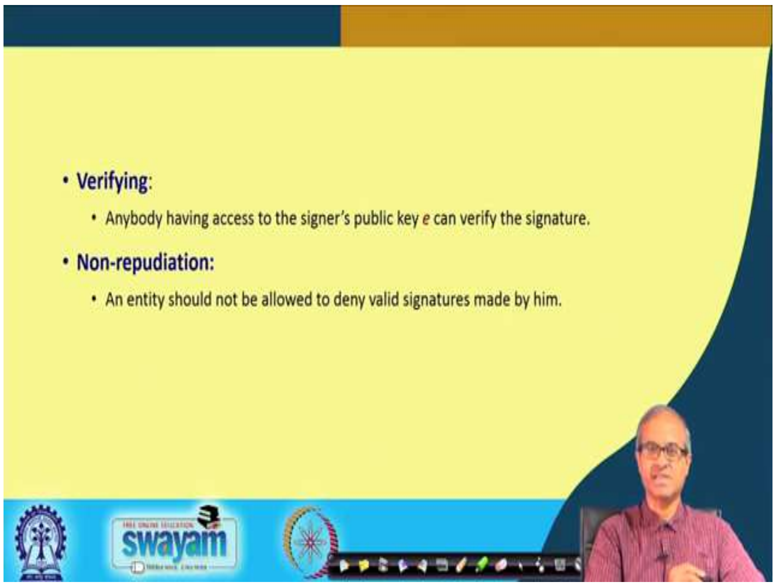
- Digital equivalent of hand-written signatures.
- Bind pieces of digital data with particular entities.
- Based on public-key technology.
- **Signing**:
  - The signer uses his private key d to sign.
- **Difficulty of forging:**
  - An entity without knowledge of this private key d cannot generate a valid signature on a new piece of data.

$\{d, e\}$

So, for signature the person will be using the private key which is not available with anyone else. And the point is that it is difficult to forge, because anyone else cannot generate the same thing, same signature, because the value of d is only lying with me, and it is not easy for anyone else to guess or break what the value of d was, ok. So, this private key value d cannot be replicated by anyone else and any entity without knowing d cannot generate the same valid signature, ok.

So, signature forging is not possible. You see even with the pen and paper signature someone can try to make a signature similar to my signature and forge, but in digital case it is simply not possible, ok.

(Refer Slide Time: 04:18)

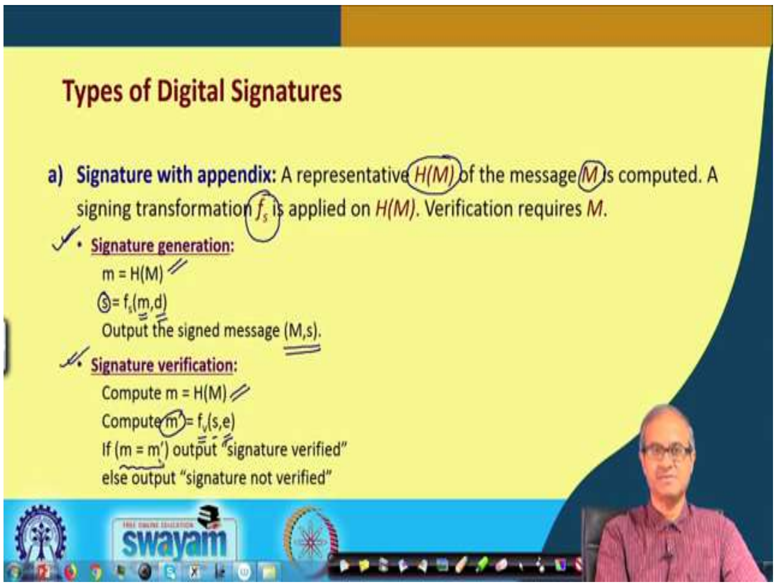


- **Verifying:**
  - Anybody having access to the signer's public key *e* can verify the signature.
- **Non-repudiation:**
  - An entity should not be allowed to deny valid signatures made by him.

For verifying the signature that as said, the signature was done using the private key of the person who is signing. Verification can be done by using the public key of the person, ok. This is the property of public key cryptography and this also allows non-repudiation. Because of the strength of the public key algorithm, well I am assuming quantum computers are not there in the picture. So, just without that public key algorithms are known to be very strong. They cannot be broken, ok.

So, this non-repudiation property also comes in as an added benefit. Someone who has signed cannot later deny that well that is not my signature, because it is not possible to make a duplicate signature or forge a signature using this scheme, using these methods, all right.

(Refer Slide Time: 05:21)



Types of Digital Signatures

a) **Signature with appendix:** A representative $H(M)$ of the message M is computed. A signing transformation f_s is applied on $H(M)$. Verification requires M.

- **Signature generation:**
  - $m = H(M)$
  - $s = f_s(m,d)$
  - Output the signed message (M,s).
- **Signature verification:**
  - Compute $m = H(M)$
  - Compute $m' = f_v(s,e)$
  - If $(m = m')$ output "signature verified"
  - else output "signature not verified"

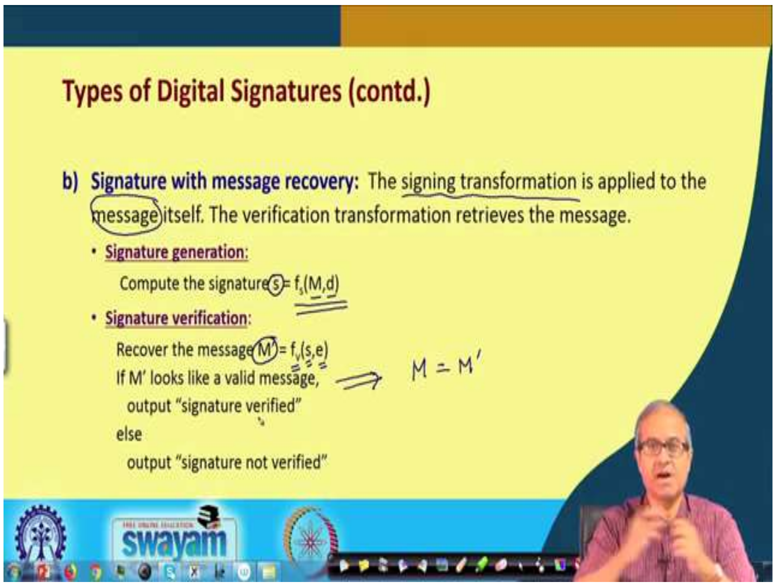
Talking about types of digital signatures, let us look at the first one which is, which was saying signature with appendix. Well, what is done here? Just you see here in the signature generation process. We will start with the message M, first we compute a hash value. We use a hash function.

So, in the first step you see here which is a $m = H(M)$. We generate the hash value of the message and generate a hash digest, small m, then we sign on that hash value. A signing transformation f_s, f_s is referred to as the signing transformation, is applied on this $H(M)$, $H(M)$ is a small m. So, on this small m I do my signature using my private key. So, it is like an encryption process using RSA, let us say. Let us say I am using RSA. I have a data. I have my private key. I am doing an encryption that is the signature I am generating, that is my signature, S.

So, I can communicate the message M along with the signature S that is my signature. So, as I said the other side whoever is receiving the message can actually verify the signature and verify the authenticity. So, how the verification can be carried out? So, again from the message, the hash value, small m is come, computed and the signature S was received. Now, there is another function f_v which is applied, which in public key context will be the decryption operation. Now, S will be decrypted using the public key e. So, as you know encryption, decryption is exactly reversible with respect to the private key and the public key.

So, if you do the decryption, you will be getting some value m' and you expect m and m' to be the same. If they are the same, you say the signature has been made. This has been verified. If they do not match, you say that there is a mismatch in the signature, not verified, ok. This is how this key works, ok.

(Refer Slide Time: 08:04)



Types of Digital Signatures (contd.)

b) Signature with message recovery: The signing transformation is applied to the message itself. The verification transformation retrieves the message.

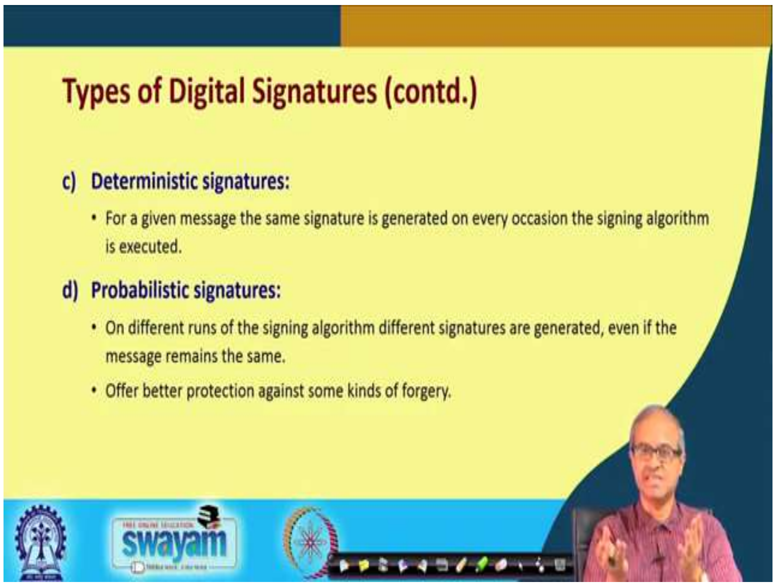
- Signature generation:
  Compute the signature $s = f_s(M,d)$
- Signature verification:
  Recover the message $M' = f_v(s,e)$
  If M' looks like a valid message, $\Rightarrow$ $M = M'$
  output "signature verified"
  else
  output "signature not verified"

There is another approach here, we are not generating a separate signature rather we are encrypting the whole message, but of course this is much more expensive. Just simply, for digital signature this second message is not visible, no one uses. Let us see what this method is? This is signature with message recovery. Here this signing transformation is applied on the entire message. Earlier we had used a hash function. Here we are not using the hash function, ok.

So, what you are doing? For signature generation, we are effectively encrypting the whole message, you see the signature, this signing transformation f_s is applied on the message and the private key d. So, whatever signature we are generating is actually the encrypted version of my whole message, but that was encrypted by the private key of the signature, signer. So, for verification the reverse process is done, f_v is applied on this signature using the public key e. So, you are expected to get back the message. So, whatever you get M', you verify it.

You see here we are not exactly sending a message, rather we are trying to authenticate, ok. So, after decryption if you see that it is like a normal text which means that you have been able to successfully decode it. You need not exactly verify whether M' is equal to M because if it looks like English text, that means, you have been successful in decoding. But if you want to be doubled sure you can make a check if $M = M'$, right. Then signature verified; otherwise not verified. But as I have said the second method is more expensive, because it involves encryption of a larger message and as I said public key encryption/decryption are expensive. They are much slower. So, if you are trying to encrypt a large chunk of data, it will be quite slow.

(Refer Slide Time: 10:38)



Types of Digital Signatures (contd.)

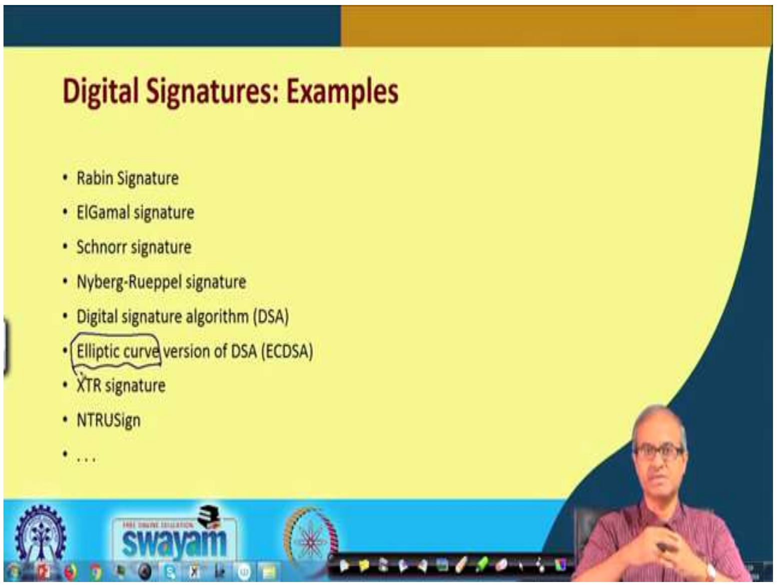
c) **Deterministic signatures:**
- For a given message the same signature is generated on every occasion the signing algorithm is executed.

d) **Probabilistic signatures:**
- On different runs of the signing algorithm different signatures are generated, even if the message remains the same.
- Offer better protection against some kinds of forgery.

So, there are other ways of generating signature also, I am not going to detail of this. There is something called deterministic or probabilistic signatures. Deterministic signatures are what we have just now talked about. Given a message every time I sign, my signature will be the same, because I am using the same algorithm. I am using the same private key. I am using the hash function, so whatever signature S I generate, will be the same. That is what is meant by deterministic signature. The same signature will be generated on every occasion you run the signing algorithm.

But probabilistic signature is something which adds another level of security, that every time there may not be a match. With some probability you say, on different runs of the signing algorithm, different signatures can be generated even for the same message. So, there are many ways in which you can use this probabilistic signature. I am again not going into detail. But this adds the level of security to subsystem where really some very high level of security is required. There you can go for probabilistic signature, ok.

(Refer Slide Time: 12:05)



Digital Signatures: Examples

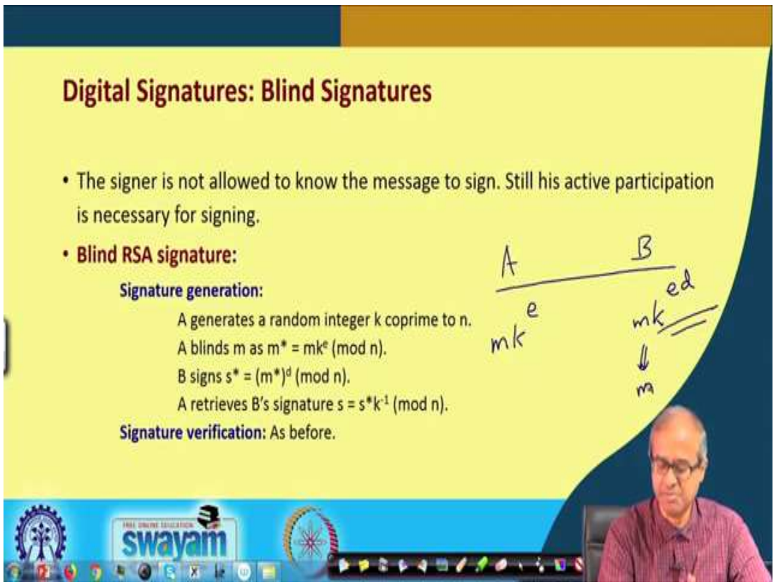
- Rabin Signature
- ElGamal signature
- Schnorr signature
- Nyberg-Rueppel signature
- Digital signature algorithm (DSA)
- Elliptic curve version of DSA (ECDSA)
- XTR signature
- NTRUSign
- . . .

Now, some examples of digital signatures, many examples are there Rabin, Elgamal, Schnorr, DSA, then elliptic curve, many methods are there. They use some transformation for generating the signature, some transformation for generating the or for actually creating the verification, verifying the signature. So, the reason that so many methods of come up is that the classic RSA algorithm is good, no doubt, but it is slow, because of the computation complexity and these methods are relatively easier to use, ok.

And just one thing I just think mention in this context, because this term has been mentioned here, but in this course, I shall not be discussing that. You see here, this elliptic curve is a phrase which have been used. Now, elliptic curve cryptography is something which is also a very emerging technique that is used for something called lightweight cryptography. I shall briefly talk about this later when we talk about lightweight cryptography. There are many devices or gadgets which run on battery. They are very limited computing resources.

Now, if you say that I have to run our RSA on top of that, it will be very much difficult. On that limited hardware, limited battery if you want to run a complicated algorithm, it can take longer time and longer battery drainage will happen. More battery drainage will happen. But elliptic curve cryptography is a method which is based on the mathematical theory of elliptic curves. Using this also you can implement public key cryptography. You can do encryption and decryption, but the advantage is that the overhead of implementations much less, both in terms of memory and also in terms of speed, right. This is one advantage.

(Refer Slide Time: 14:18)



Digital Signatures: Blind Signatures

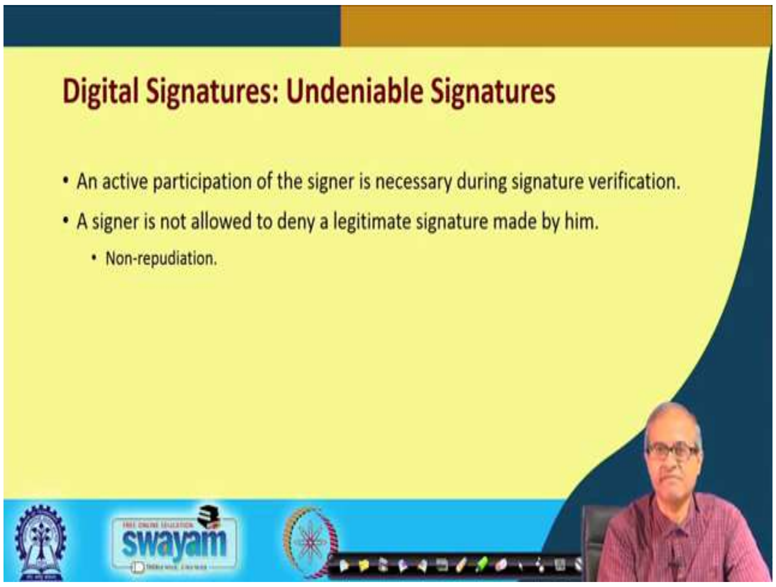
- The signer is not allowed to know the message to sign. Still his active participation is necessary for signing.
- **Blind RSA signature:**
  - **Signature generation:**
    - A generates a random integer k coprime to n.
    - A blinds m as $m^* = mk^e \pmod{n}$.
    - B signs $s^* = (m^*)^d \pmod{n}$.
    - A retrieves B's signature $s = s^* k^{-1} \pmod{n}$.
  - **Signature verification:** As before.

There is another kind of digital signature, called blind signature. Here the idea is that the signature is not allowed to see the message. The person who is signing, is not allowed to see the message. Normally, in the earlier, other method the message is given to the person signing this. The person will be just applying some transformation directly on the message. But in blind signature, the idea is that send the, suppose A is the sender, ok, A is the person who is trying to send the message.

So, what A will do? A will generate a random number key which is co-prime to that n of, let us RSA. It, the example I have taken is for RSA. The product of the two prime number that n. So, what A does before signature process is done, is that it multiplies the message m with k^e so that the original message automatically becomes garbled. Then A will give this m^* to another person B, maybe secretary to do the signing on his behalf, but this B cannot see the message. B will simply apply a signature which is m^{*d}, right. So, this is just a process.

Or you can see in another way that, A is the sender, B is the receiver, you can also think in this way that A does mk^e. This is how signature is generated and B does $(mk^e)^d$ that means, mk^{ed}, right. Now, $ed = 1 \; mod \; n$ for RSA, we mentioned. So, this will vanish and we can get back m, right. So, this method is not that widely used, but still I have just mentioned, because this is an alternate approach.

(Refer Slide Time: 16:37)



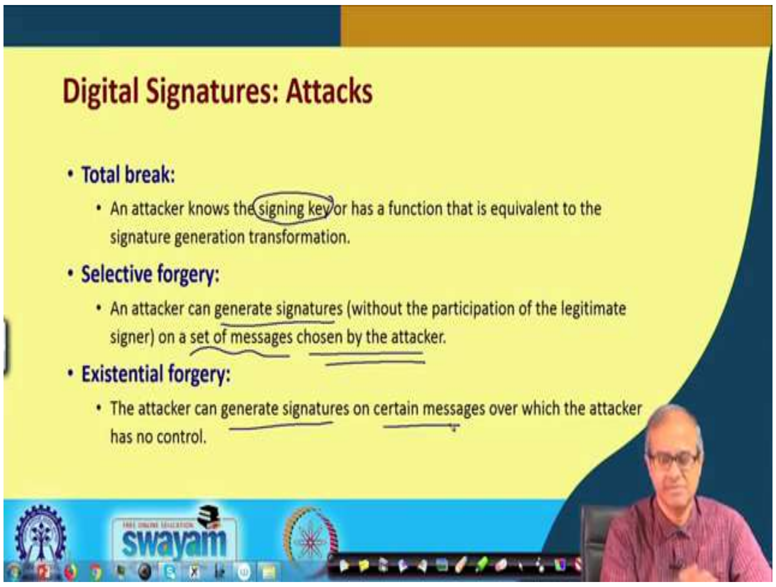
Digital Signatures: Undeniable Signatures

- An active participation of the signer is necessary during signature verification.
- A signer is not allowed to deny a legitimate signature made by him.
    - Non-repudiation.

Well, undeniable signature, this I have mentioned earlier also. Let me repeat. Here an active participation of the signer is necessary during signature verification. This is, this another scheme called undeniable signature. So, during signature verification process, the person who has signed that person's participation is also required.

So, there is another, I mean another class of algorithms where the person who is signing also participates with the recipient during the verification process, but again you can understand this scheme has limited applicability, because I am signing and sending. I am not worried about anything after that. If someone calls me and ask me every time that whether this is my signature or not, it would be quite annoying for me, right. So, this method also has a rather limited use.

(Refer Slide Time: 17:37)



Digital Signatures: Attacks

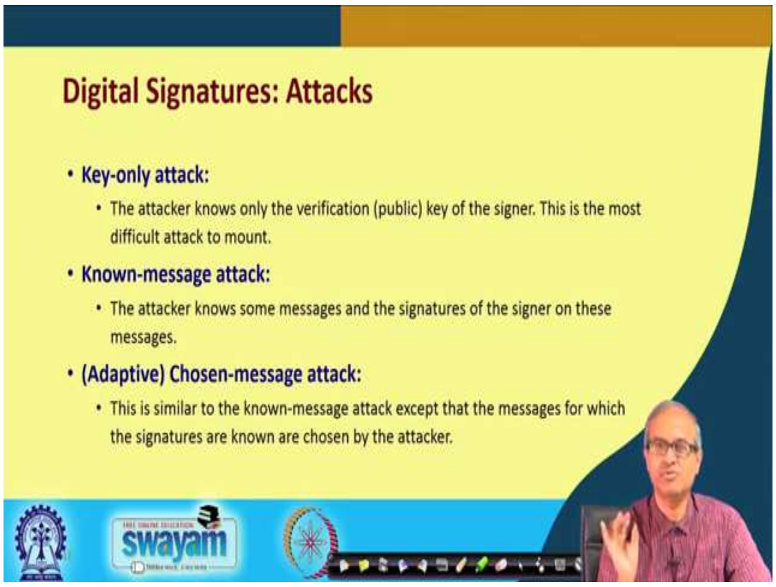
- **Total break:**
  - An attacker knows the signing key or has a function that is equivalent to the signature generation transformation.
- **Selective forgery:**
  - An attacker can generate signatures (without the participation of the legitimate signer) on a set of messages chosen by the attacker.
- **Existential forgery:**
  - The attacker can generate signatures on certain messages over which the attacker has no control.

Now, talking about attacks on this kind of signatures. There can be a total break. Total break means somehow the attacker gets hold of my private key and that is the ultimate, ok. The private key is not supposed to be shared with anybody. If the attacker gets the signing key, then you can do anything, modify the message, again generate the signature appended to it. So, any recipient will feel that this is the correct message with the correct signature, right.

Selective an existing forgery are a subset of that. For selective forgery, says an attacker can generate signature without the participation of the signer. Signer is not the only picture can generate signature on a set of messages, only not on all messages, but the messages are chosen by the attacker. The attacker can choose some messages not all, on that valid signatures can be generated. Well, these are often made possible because of some weaknesses in the signature algorithm, weaknesses in the hash algorithm. Like earlier I said md5 was used, they had some weaknesses. So, this kind of forgery was possible.

And existing, for existential forgery means the attacker can generate signature on certain messages, but those messages cannot be generated by the attacker. Somehow, for some messages arbitrary messages the attacker can generate signature, but attacker cannot by their own choice generate a message and a generate signature for that. So, these two similar, but this existence forgery is more difficult; more difficult than selective forgery, ok.

(Refer Slide Time: 19:38)



Digital Signatures: Attacks

- **Key-only attack:**
  - The attacker knows only the verification (public) key of the signer. This is the most difficult attack to mount.
- **Known-message attack:**
  - The attacker knows some messages and the signatures of the signer on these messages.
- **(Adaptive) Chosen-message attack:**
  - This is similar to the known-message attack except that the messages for which the signatures are known are chosen by the attacker.

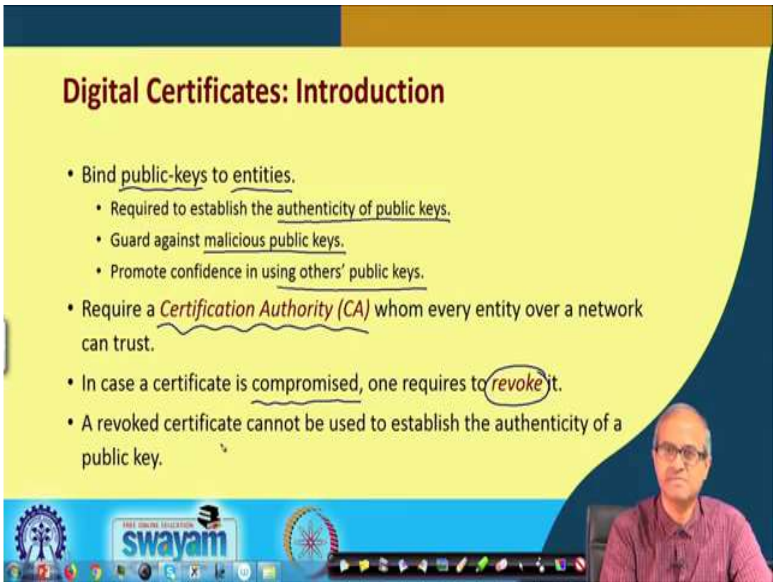
Then you can have key only attack. The attacker knows only the verification of the public key which is true in general, but this is not possible, if you assume that your let us say, RSA algorithm is good, it cannot be broken. So, this is the most difficult attack. If you say that you are able to break it, which means you are able to break RSA, right. Known message attack means the attackers knows some message and the corresponding signature pairs with that it can analyze and try to find out that whether you can make some modifications and generate a valid signature or not.

And chosen message attack or adaptive chosen message attack is similar to the last one. In the previous slide that the attacker can choose some messages for which the signatures are known. So, these are all different categories and classes of attacks, but just one thing you should remember, these are not easy to mount. These are all very difficult, ok.

Now, talking about digital certificates which we use regularly in our day-to-day online transactions, like when I said that when we do Internet banking, we normally visit a website that starts with the https. That is supposed to be a secure connection. After getting logged in, we type in our Internet password, username, password, everything which is supposed to be very secure, because there is money involved, right.

Now, what really happens underlying is that my browser verifies the digital certificate of the other side. Suppose, I am visiting State Bank of India, so I need to get the public key of State Bank of India. So, how do I get public key? I need to download the digital certificate of State Bank of India. Public key is part of that certificate. So, certificate is nothing but the public key of a person of an entity along with lot of other information which you can verify that it is the right certificate, you are looking at.

(Refer Slide Time: 22:09)



Digital Certificates: Introduction

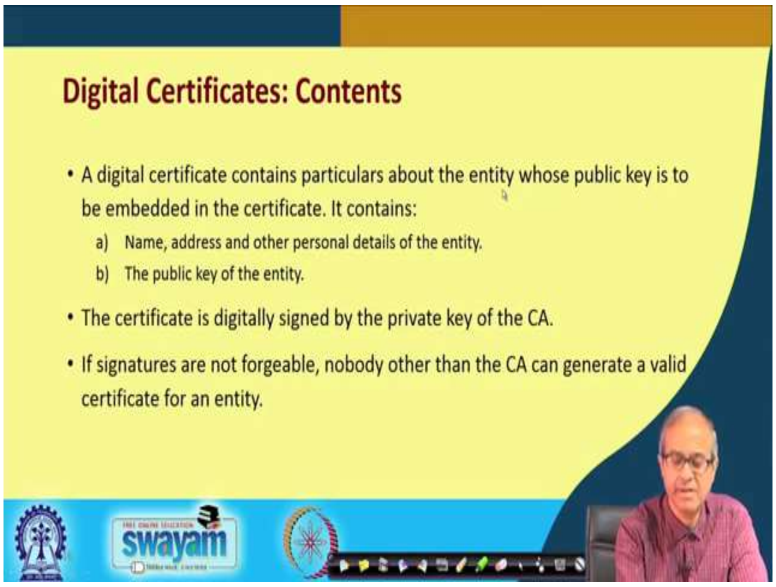
- Bind public-keys to entities.
  - Required to establish the authenticity of public keys.
  - Guard against malicious public keys.
  - Promote confidence in using others' public keys.
- Require a *Certification Authority (CA)* whom every entity over a network can trust.
- In case a certificate is compromised, one requires to *revoke* it.
- A revoked certificate cannot be used to establish the authenticity of a public key.

So, basically digital certificate helps in binding public keys to entities. Just as I have said certificate contains the public keys along with the name of that entity, address and some other information. So, the certificate is some kind of a standardization, part of a standardization, so that you can actually use it in applications. So, you can use the certificates and here there comes the concept of a trusted third party. you will get the certificates from a trusted third party which is a certification authority.

So, with that you can establish the authenticity of the public keys you are downloading. Through the certificates, you can guard yourselves against malicious public, because you are trusting, this third party certifying authority is supposed to be trustable, ok. So, in this way you can promote your confidence in using others public key for actually message transmission, encryption, decryption, everything, ok.

Now, of course, there are instances where some certificate can be compromised and if the certifying authority identifies that some certificate has been compromised, then there is a process of revoking the certificate. Well, a revoked certificate cannot be any further used to establish the authenticity. Revoke certificate means it goes to means, another list which means, those are the certificates which have been compromised.

(Refer Slide Time: 24:05)



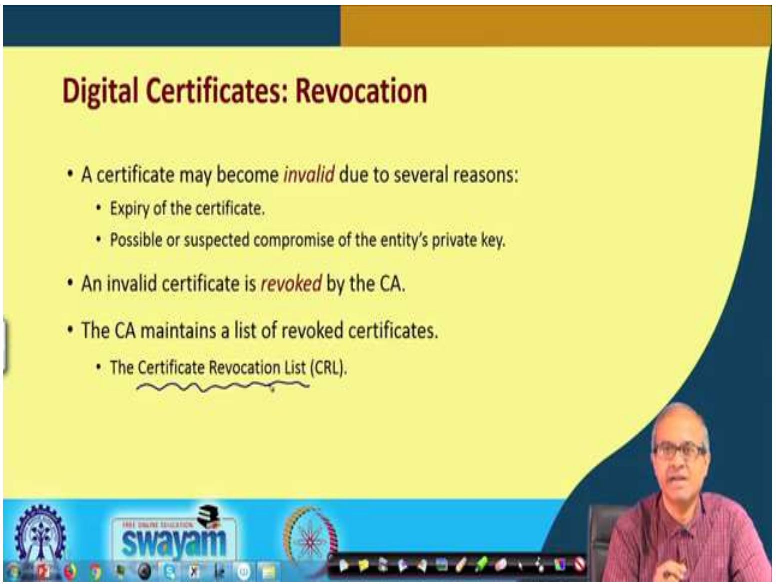
Digital Certificates: Contents

- A digital certificate contains particulars about the entity whose public key is to be embedded in the certificate. It contains:
  a) Name, address and other personal details of the entity.
  b) The public key of the entity.
- The certificate is digitally signed by the private key of the CA.
- If signatures are not forgeable, nobody other than the CA can generate a valid certificate for an entity.

Well, what are the contents of a digital certificate? It contains particulars about entity as it said. The person whose certificate you are downloading name, address and other personal details may be and of course, the public key; public key is the most important. And in order to verify that whatever you are downloading is correct and no one is modifying that in transit because of that whatever you are downloading that certificates will be digitally signed by the certifying authority.

With its own private key that CA will be signing that and you will first be verifying that signature, then you will be using the private key, public key, that you are downloading, right, fine. This is how it works.

(Refer Slide Time: 24:56)



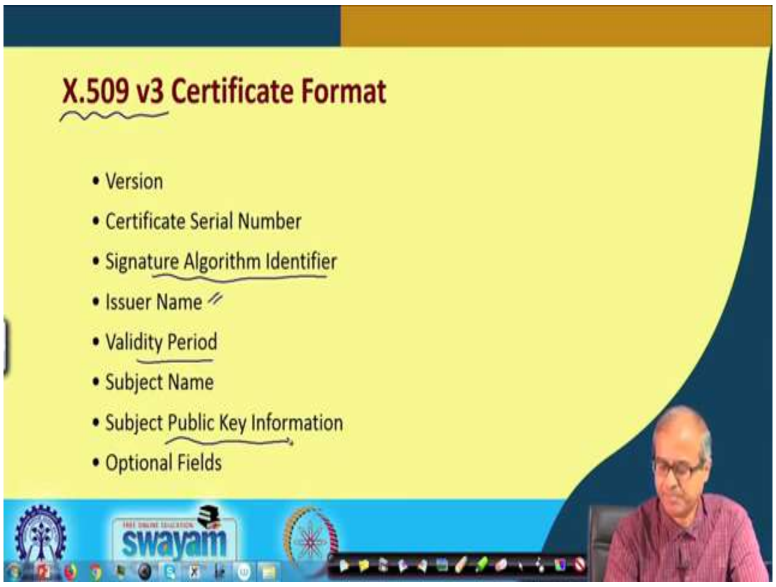
Digital Certificates: Revocation

- A certificate may become *invalid* due to several reasons:
  - Expiry of the certificate.
  - Possible or suspected compromise of the entity's private key.
- An invalid certificate is *revoked* by the CA.
- The CA maintains a list of revoked certificates.
  - The Certificate Revocation List (CRL).

Revocation as I have said that sometimes a certificate may become invalid. See whenever you create a certificate, there is an, there is a duration; there is a duration for which the certificate will be valid. So, if the time expires, that is one reason why it may become invalid or there can be some suspected compromise, some activities are going on which as, which are susceptible, because of that you can temporarily put it into the suspect list and it becomes invalid.

And invalid certificates are revoked by the see, how it is done? The certifying authority maintains a separate list called certificate revocation list. So, these certificates go to that CRL and after a time they get deleted or removed from the CRL also, ok.

(Refer Slide Time: 26:02)

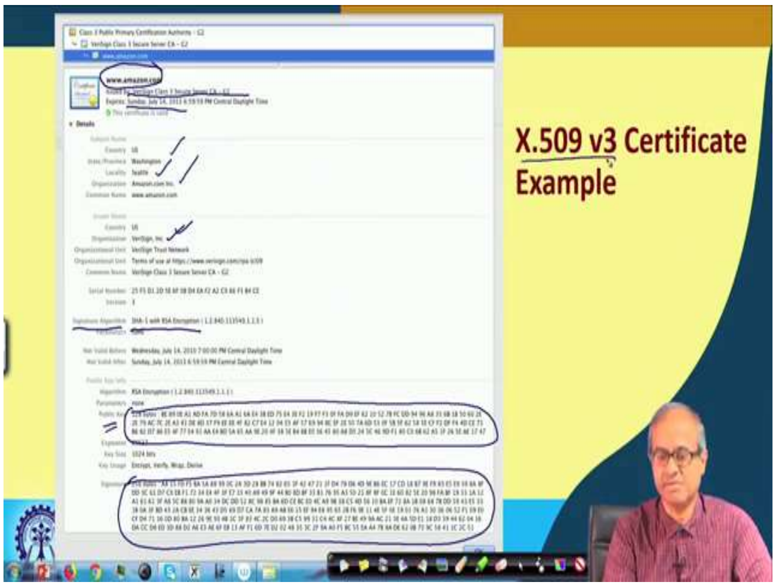


X.509 v3 Certificate Format

- Version
- Certificate Serial Number
- Signature Algorithm Identifier
- Issuer Name
- Validity Period
- Subject Name
- Subject Public Key Information
- Optional Fields

So, I have talked about the standardization. So, one of the commonly used certificate standard is $X.509\ v3$, that is most widely used. And among other things the certificate contains version, some serial number, which signature algorithm is used. Because you need to verify the signature, right. The certifying authority is digitally signing it. So, which algorithm is being used? Some information about that. Who is the issuer of the certificate, of the public key and private key? What is the validity period and of course, public key information of the sender?

(Refer Slide Time: 26:51)



X.509 v3 Certificate Example

Class 3 Public Primary Certification Authority - G2
VeriSign Class 3 Secure Server CA - G2
www.amazon.com

www.amazon.com
Issued by: VeriSign Class 3 Secure Server CA - G2
Expires: Sunday, July 14, 2013 6:59:59 PM Central Daylight Time
This certificate is valid

Details

Subject Name
Country: US
State/Province: Washington
Locality: Seattle
Organization: Amazon.com Inc.
Common Name: www.amazon.com

Issuer Name
Country: US
Organization: VeriSign, Inc.
Organizational Unit: VeriSign Trust Network
Organizational Unit: Terms of use at https://www.verisign.com/rpa (c)09
Common Name: VeriSign Class 3 Secure Server CA - G2

Serial Number: 25 F5 D1 2D 5E 6F 38 D4 EA F2 A2 C8 66 F1 84 CE
Version: 3

Signature Algorithm: SHA-1 with RSA Encryption (1.2.840.113549.1.1.5)
Parameters: none

Not Valid Before: Wednesday, July 14, 2010 7:00:00 PM Central Daylight Time
Not Valid After: Sunday, July 14, 2013 6:59:59 PM Central Daylight Time

Public Key Info
Algorithm: RSA Encryption (1.2.840.113549.1.1.1)
Parameters: none
Public Key: 128 bytes : [illegible]
Exponent: 65537
Key Size: 1024 bits
Key Usage: Encrypt, Verify, Wrap, Derive

Signature: 128 bytes : [illegible]

Let us take a sample certificate example. Of course, the font size is too small. You possibly not be able to read it, but you can download such things from the Internet and see for yourself. If you search with digital certificate example, you will get many such documents. Now, let me tell you here what are the things I can see.

Here I can see first is that this is a certificate of $amazon.com$, issued by VeriSign. VeriSign is a trusted party who have generated the public key and private key. Expires, some date is given here, expiry date, ok. So, the name of the organization, Seattle, address, $amazon.com$, Seattle Washington, ok. So, some information about the person whose certificate I am downloading. The entity is given here so that I can know that well I am downloading the correct certificate. VeriSign is the person who had signing on my own, signing on their behalf. And what signature algorithm is used? Signature algorithm is SHA-1 with the RSA encryption, it mentions clearly what algorithm is used and, ok.

At the end this is the signature. You can verify the signature from here, right. So, it generates a 256-byte signature. So, you can verify the signature from here. And you have the public key here, this the public key. Public key is what you are interested in ultimately, right, but for verification whether you are getting the correct public key, we have some other information along with the certificates coming to you. So, this is how a certificate looks likes and this is part of the X 509 standard, ok.

So, with this we come to the end of this lecture. Over the last 3 lectures, we have looked at this cryptographic hash functions and various ways we can use it for authentication and message integrity, prevention applications.

Thank you.