

Information Security 3
Sri M J Shankar Raman,
Consultant Department of Computer Science and Engineering,
Indian Institute of Technology Madras
Module 54
Digital Signatures

So in this module we will actually be talking about a cryptographic technique called as a digital.

(Refer Slide Time: 00:22)

Digital Signatures

Cryptographic technique analogous to hand-written signatures.

- sender (Bob) digitally signs document, establishing he is document owner/creator.
- **verifiable, nonforgeable:** recipient (Alice) can prove to someone that Bob, and no one else (including Alice), must have signed document

Signature, so what exactly is a digital signature, so it's the cryptographic technique that is very similar to our normal handwritten signatures right, so when we actually try to attach a piece of a document, a hard copy document with our own signature what essentially we are trying to convey here is that we are basically accepting the contents of whatever has been written down or printed out.

In that piece of paper till the line that we in which we are actually putting our physical signature right, so essentially the whole reason for this as we are very well understand is tomorrow we should not be in a position to basically sort of deny that we have never accepted the contents of whatever we are signing right now with right, so similarly if I have a requirement that I should basically sign a document in a digital form.

In a soft copy form then the digital signature that we are going to be talking about here is the very common technique that is typically used to sort of establish that this particular person has actually accepted the contents of the document that he is right now digitally signing with, so he could actually be the document owner or he could be a just a a person who has modified the contents of the document or whatever it is.

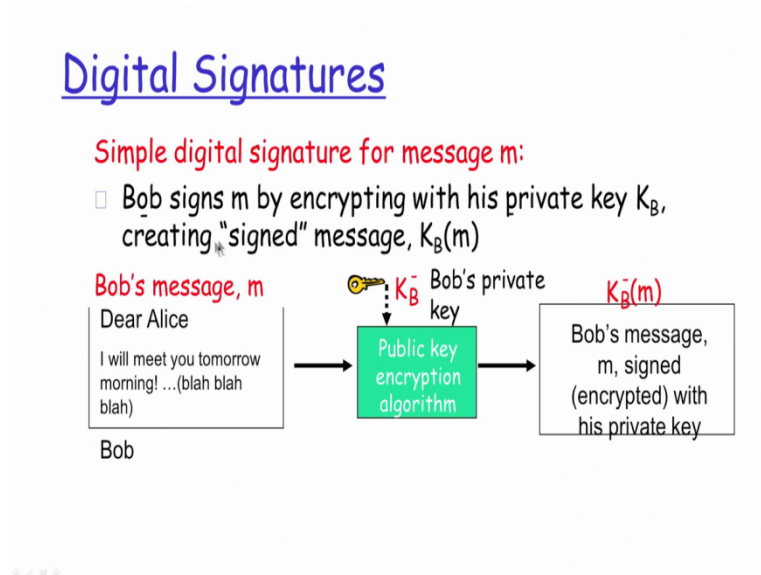
But essentially by affixing the digital signature, the person is sort of accepting that he is agreeing to whatever are the contents of that particular soft copy or the digital document right, so this digital signature is expected to be verifiable and also non forgeable, now what we mean by verifiable is that if it all there is a contention that it has to be taken to a third party for a sort of verifying that it was indeed this person who has actually affixed his digital signature.

Then there is something that should be possible, number one, number two when we talk about non forgeable as we would very clearly understand forge ability is basically a mechanism by which somebody tries to put our signature or sort of forges our signature in such a way that without our real acceptance we are born by whatever has been actually stated in that particular document right, now with this digital signature mechanism.

It also should be possible that nobody should be capable of forging our signature out and then sort of then making it as a part of our acceptance to the contents of whatever has been forged and signed with our digital signature, so these two are the primary requirements of the digital signature one it should be verifiable and then two it should be non forgeable by any kind of a unauthorized person right.

So in this case the recipient should be able to prove to someone that the the original sender as been the one who has actually been signed the documents digitally and on his behalf or without his consent or awareness nobody else had actually signed it right, so with these important characteristics we will see how the digital signature is actually created and how does it actually work.

(Refer Slide Time: 03:49)



So how does this digital signature really work? So when Bob wants to send a message m to a recipient, what Bob actually tries to do is he basically signs a message m by encrypting with his own private key, so as we had seen in the previous modules a private key is something that is actually owned by the person alone and it is expected that the private key will not be known even to the second person right, so Bob basically signs the message that he wants to send out by encrypting with his private key.

And creates what is called as a signed message, the signed message is nothing but the original message encrypted by the private key, so if this is a message that Bob wants to send out he basically takes his private key encrypts it and then sends it out to the recipient over the network channel in order to make it, make the message finally reach the the recipient.

(Refer Slide Time: 04:47)

Digital Signatures (more)

- Suppose Alice receives msg m , digital signature $K_B(m)$
- Alice verifies m signed by Bob by applying Bob's public key K_B to $K_B(m)$ then checks $K_B(K_B(m)) = m$.
- If $K_B(K_B(m)) = m$, whoever signed m must have used Bob's private key.

Alice thus verifies that:

- Bob signed m .
- No one else signed m .
- Bob signed m and not m' .

Non-repudiation:

- ✓ Alice can take m , and signature $K_B(m)$ to court and prove that Bob signed m .

So let us say that Alice receives this particular message with the digital signature that is basically the signed message.

Now Alice basically verifies the m signed by Bob by trying to apply the Bob's public key, so as we had actually seen before the basic essence of the public key cryptography is that if I apply private key on one side while at the time of encryption and I apply the public key on the other side at the time of decryption, I should be able to generate the original plain text message back successfully right, and also vice versa so if I apply the the public key first and I can actually apply the private key.

At the time of decryption on the other side and then also I will be able to generate the original message back, so for the purpose of digital signature what Alice has a recipient will basically do here is that, since she is actually having access to Bob's public key, Bob's public key will be applied on the signed message and then the original message will be extracted, now how is this really turning out to be a digital signature for the properties of it being verifiable and non forgeable that we saw.

In the previous slide, this is how it is, so Alice basically verifies that Bob has signed m why has Bob signed m because of the fact that it was Bob's public key that has been used at the time of

decrypting it should have been actually Bob's private key that should have been applied on the message right, then that is how by the property of the public key cryptography I would have been able to get back the original message right.

Now because of that fact since the private key is going to be available only with Bob and I have been able to generate the original message back by applying the public key of Bob, it has basically been proved that it has been Bob who has actually signed the original message, because it is expected that the private key of Bob is with nobody else other than Bob right, so with that fact Alice basically verifies that Bob has actually signed the message.

Now no one else could have signed this particular message because again it is expected that the private key of Bob is with nobody else other than that of Bob and because of the fact that I have generated the message at the time of decryption by applying Bob's public key the original message, the signed message should have been encrypted only with that of the Bob's private key right, so this sort of indirectly proves that nobody else.

Other than Bob would have actually signed the original message right and then third fact is that Bob has actually signed the message and not some other message right, so not some other message because again if it has if he has basically signed something else and for some reason that that particular signed message that got modified, when I had actually applied Bob's public key at the time of decryption it I wouldn't, I wouldn't have actually got back the original message right.

So with these three facts it is basically proven that it was indeed Bob who had actually signed the message and then send it out to the Alice in this case who is acting as a recipient, now the next part of this is non repudiation right, now what we mean by non repudiation is that a person who has actually sent a message today cannot claim tomorrow that this particular message was not send by him at all right.

So now again how is this digital signature going to help us in establishing this non repudiation because Alice can basically take the message and the signature that Bob had actually sent Alice to court and then prove that Bob has actually signed m because since for the purpose of decryption here only the public key of Bob is required even the court of law would actually have

access to the public key of Bob and when the court actually applies the public key of Bob on this signature.

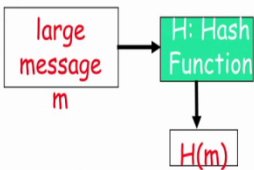
And on a signed message the original message which Bob would have actually send would be getting regenerated and thereby Bob cannot sort of deny tomorrow that this particular message was not send by him, right?

(Refer Slide Time: 09:06)

Message Digests

Goal: fixed-length, easy-to-compute digital "fingerprint"

- apply hash function H to m , get fixed size message digest, $H(m)$.



```
graph TD; A[large message m] --> B[H: Hash Function]; B --> C[H(m)];
```

Hash function properties:

- produces fixed-size msg digest (fingerprint)
- given message digest x , computationally infeasible to find m such that $x = H(m)$

So now coming into the message digest what exactly is a message digest, so if I really want to generate a digital finger print to such an extent that I want to sort of verify the data integrity part of it right.

Now what we mean by data integrity is that if the sender has actually sent a set of messages the receiver should basically receive the same message content and not a modified text right, so this basically what is referred to as a data integrity, now what exactly is the goal behind creating something called as a message digest, so the goal is basically to get a fixed length easy to compute digital fingerprint right, so fixed length message digest easy to compute digital fingerprint.

Which we will typically make use of by generating what is called as a hash function right, so when I apply hash function to my message I get as a output of that hash function a fixed size

message digest which we referred to as h of m right, so the original message is applied to the hash function and I get out something called as an hash value out of it right, now what should be the basic property of the hash function, the basic property of the hash function is that it should basically.

Produce me a a fixed size message digest that I am going to call it as a fingerprint and this particular message digest is going to be sent along with the message to the recipient right, so now what is expected is that the recipient will know the same hash function and how this hash function is operating also and because the hash function is going to be operating on the message after it has been received in the receiver side, if there has been no modification in the message content

Between the source and the destination, I should be able to get back the same fixed size message digest on the receiver side also right, now if I get back the same fixed size message digest on the receiver that we calling as fingerprint here, then that essentially means that as part of my transfer there has been no modification whether intentionally or unintentionally of the entire text message right, but in the other hand if there has been a modification right.

My message digest is going to be getting modified because of which I will be able to verify that there has been some corruption in transit after it has actually been transmitted out from the sender right, so based on that in in a lot of situations I could basically take a call on what I want to do with that type of a data, once I detect that there has been a a corruption or an in in adverted modification of the data in transit right.

So the second property of the hash function should be that given a message digest text computationally it should be infeasible to find the m such that x is equal to h of m , so I don't want the the original message to be regenerated out by having a look at what is my message digest value right, so if that is a scenario it becomes difficult for me to use it from the network security point of view, so thereby this particular property is also mandated for any hash function that is used.

Wherein that if a particular person who is actually trying to look at the transit data gets access to the message digest looking at the message digest value it should be computationally infeasible to

generate back the message for which this particular message digest has actually been calculated right, so these are the typical properties of the hash function that would be acquire to have for implementing the purpose of the data integrity part of it, from the network security point of view.

(Refer Slide Time: 13:09)

Hash Function Algorithms

- **MD5 hash function widely used (RFC 1321)**
 - computes 128-bit message digest in 4-step process.
 - arbitrary 128-bit string x , appears difficult to construct msg m whose MD5 hash is equal to x .
- **SHA-1 is also used.**
 - US standard [NIST, FIPS PUB 180-1]
 - 160-bit message digest

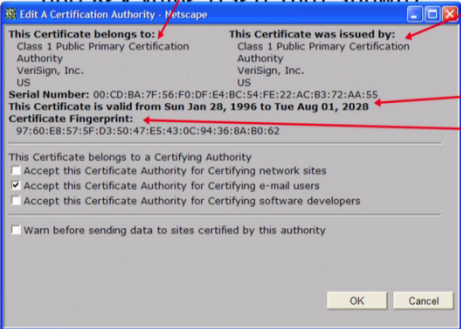
So some of the very common hash function algorithms you have an md5 hash hash function which is actually having a standard document RFC 1321 so it computes the 128 bit message digest in a typical four step process and md5 is an algorithm that is not that much strong today because of which you have another algorithm called as a sha1, so SHA basically stands for secure hash algorithm, so you have sha1, you have sha256, sha512 and so on.

Which are much more stronger than the first md5 hash function algorithm that came out for generating these message digests.

(Refer Slide Time: 13:53)

A certificate contains:

- Serial number (unique to issuer)
- info about certificate owner, including algorithm and key value itself (not shown)
- info about certificate issuer
- valid dates
- digital signature by issuer



The screenshot shows a Windows dialog box titled 'Edit A Certification Authority' with the following content:

This Certificate belongs to: Class 1 Public Primary Certification Authority VeriSign, Inc. US	This Certificate was issued by: Class 1 Public Primary Certification Authority VeriSign, Inc. US
Serial Number: 00:CD:BA:7F:56:F0:DF:E4:BC:54:FE:22:AC:B3:72:AA:55	
This Certificate is valid from Sun Jan 28, 1996 to Tue Aug 01, 2028	
Certificate Fingerprint: 97:60:EB:57:5F:D3:50:47:E5:43:0C:94:36:8A:B0:62	

Below the table, there are checkboxes for 'Accept this Certificate Authority for Certifying network sites', 'Accept this Certificate Authority for Certifying e-mail users', and 'Accept this Certificate Authority for Certifying software developers'. At the bottom, there is a checkbox for 'Warn before sending data to sites certified by this authority' and 'OK' and 'Cancel' buttons.

So coming down to the fact of a digital certificate, now why is a digital certificate required is that, we've always been telling that the public key is something that needs to be shared for example between the two parties, so how could the public key be effectively shared would be something like wherein I would typically make use of a digital certificate right.

So a typical digital certificate that will be exchanged between the client and the server right, before the the data transfer starts in an encrypted form that the digital certificate would be of this particular format type, so this will typically have a unique serial number, so you see a unique serial number here and it also will have details about who is a certificate owner, so including what algorithm is actually been used and what key value has being used.

But of course the key value will not be explicitly displayed out right, so there will also be a validity part of the digital certificate, which will be also available and whenever this the digital certificate is actually tried to be used after the validity has expired right, that is when we will typically find out so for example sometimes we actually get a warning message when we try to go to some website right, so it says that this particular digital certificate that the server.

Is actually giving to the browser right now is something that has got invalid maybe for any reason right, so it could be because the certificate itself has been revoked or the certificate could have become invalid because the validity period has expired or whatever it is right,

So at that time the the browser will basically show up a popup message to the user saying that there is a problem with this particular certificate that has been given by the server.

And whether the the user would still like to continue with it by adding it as an exception right, so if the user is very clear that he wants to take the address he can say that I want to edit as an exception and then continue with the session but on the other hand if the users thinks that it has a security problem, it's a security issue possibly for him, he can basically block it up as well and then say that he doesn't want to continue with the session right.

So because of that particular reason the the validity of the digital certificate that is used for exchanging the keys is also basically part of the digital certificate itself.

Thank you.