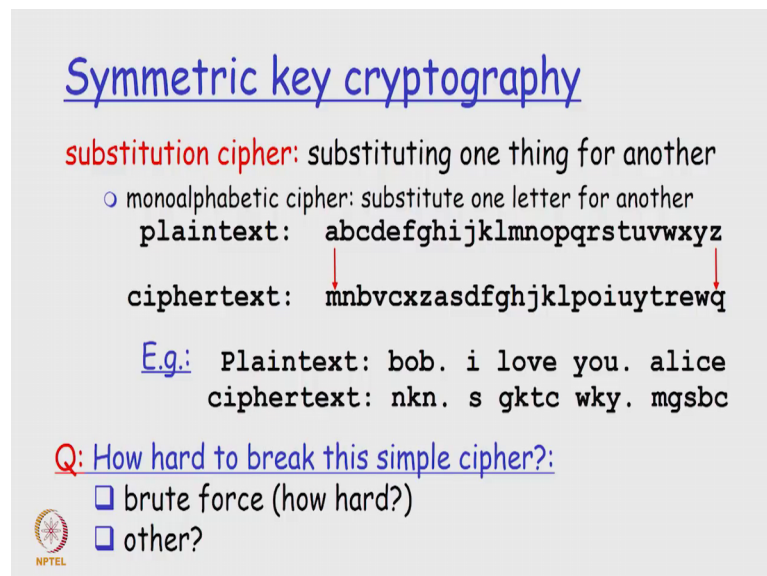


Information Security: Level #4
Prof. Kamakoti
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Lecture - 05
Symmetric Key Cryptography and Digital Signatures

Welcome to this session and we will start about symmetric key cryptography. What is symmetric key cryptography? Let me give a very simple example this is this is basically the script encryption algorithm is called substitution cipher.

(Refer Slide Time: 00:26)



Symmetric key cryptography

substitution cipher: substituting one thing for another

- monoalphabetic cipher: substitute one letter for another


plaintext: abcdefghijklmnopqrstuvwxyz

ciphertext: mnbvcxzasdfghjklpoiuytrewq

E.g.: Plaintext: bob. i love you. alice
ciphertext: nkn. s gktc wky. mgsbc

Q: How hard to break this simple cipher?:

- brute force (how hard?)
- other?



So, what is this substitution cipher? So, this the example that I am showing on the screen is called a mono alphabetic cipher; for example, every alphabet will be replaced by some other alphabet like a becomes m, B becomes n, c becomes B like that a to z you have a another map.

So, what will happen? If plaintext Bob says I love you Alice; cipher text will become n K n; s g K t c; w K y; m g s B c. Now when Bob sends to Alice this message; Alice should know the exact reverse like from the cipher text, it easy she should know how to get back to the plaintext; so, this is the important thing. So, this mapping between plaintext to the cipher text is what both Bob and Alice should know. When they know this mapping to the this thing from this text to the other text; then they basically they again you know

send any message. And then basically get back you can encrypt using this message and decrypt using the same mapping.

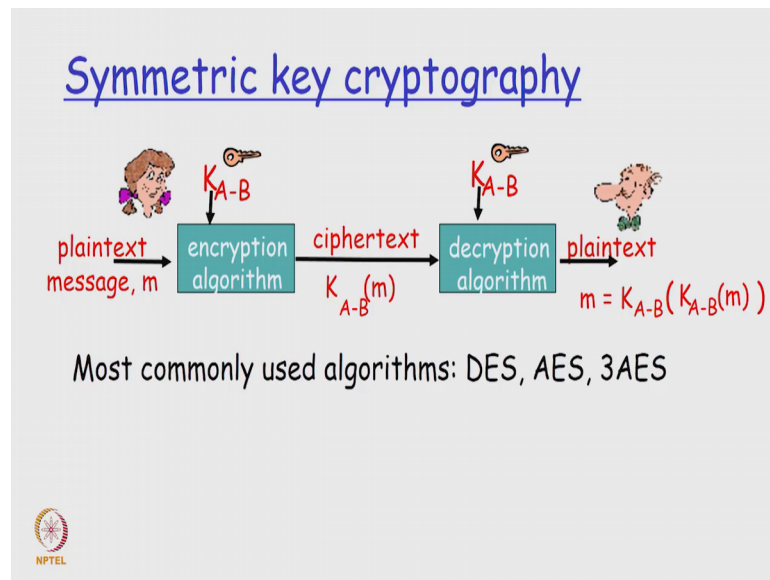
Now, this is this is a very easy way of implementing you know alphabet based communication, but the point is that when we are talking about confidentiality; when Alice sends to Bob; Bob alone should understand what the message is, but many people can receive the messages; it is a public network, it may it go into several lines and so, many any people can get it; many of the people can get it many of the components in the network can get it.

Now, what is some other component is interested in understanding this? Suppose they try to understand this; this mono alphabetic substitution cipher becomes extremely easy. One can even do a brute force whatever be; however, hard it may be we can just do a brute force; and then you can find out. Somebody can actually you know cook up some small sentences and if you know one or two mapping then it is very easy for them to find out to other mapping.

But this is symmetric key because the encryption that is known to Bob is the same encryption the mapping that is known to Bob is the same mapping that Alice also need to understand. And if they understand that both then this entire substitution cipher will work, but this is not too secure because we could have many types of attacks including some dictionary attacks and other things by which you can basically decrypt the message ok.

So, this is one example of symmetric key cryptography, but not a very you know strong example, but this is a way this is this is an example that I can introduce to you to make you understand what is symmetric key cryptography; to repeat Alice and Bob has introspective encryption and decryption algorithms. And these encryption and decryption requires a key and the key is the same; in this case the key is actually the mapping from one to another and vice versa.

(Refer Slide Time: 03:48)



So, the symmetric key cryptography works like this there is a plaintext message m that is sent there is a key K ; A to B , but its Alice to Bob that will be used for encrypting this that will give a ciphertext which is K of m and this reaches the decryption side; we get where again the same key K A minus B because symmetric key same key is used and you get a plaintext which is again K A minus B of K A minus B of m should be back equal to m .


So, the encryption algorithm that I follow should ensure the property that; I take a message, I encrypt using a key, I again decrypt using the same key; I should get back the original message. The most commonly used symmetric key algorithms are the DES the AES and 3 AES.

(Refer Slide Time: 04:39)

Public Key Cryptography


symmetric key crypto

- ❑ requires sender, receiver know shared secret key
- ❑ Q: how to agree on key in first place (particularly if never "met")?



public key cryptography

- ❑ radically different approach [Diffie-Hellman76, RSA78]
- ❑ sender, receiver do *not* share secret key
- ❑ *public* encryption key known to *all*
- ❑ *private* decryption key known only to receiver



Now let us go into what are the issues in symmetric cryptography? The main issue here is that the requires; this requires the sender and the receiver should know what the shared secret key is right.

So, this is very important; so, if I have 100 people I am communicating with for each one I should have a key. So, I from 100 different keys and I have to use them judiciously whenever I am communicating right this is one part. The second thing is how to even agree on the key in the first place, particularly if you have never met each other.

How do we first agree? Like I should send a message to you saying this is the key and how will I encrypt that message? Because I need a key to encrypt it right and that key itself I am communicating, how will I encrypt a key right. So, that is the most important aspect right.

So, and we could have see today I am; so, let us say we have created a router today that router would not have talked to some country other country router or some other router in the first day. Second day it will discover a new router because as and when somebody tries to access say something outside in that region, then only this router will start route talking to the next router.

So, in a network new routers are added and every time you start talking to the new net network router. So, every time when I am talking to someone new; I need to start

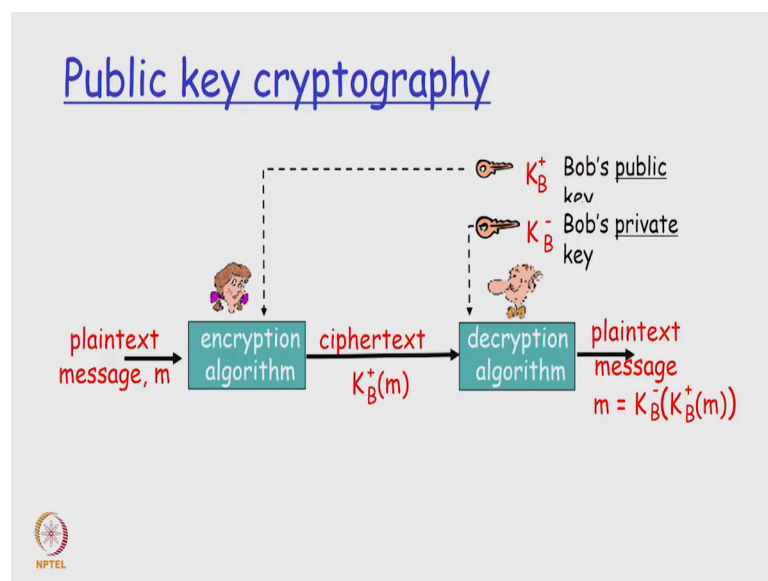
generating a key and understand what that keys and both of them should start using that key; this is going to be practically impossible.

So, the thing is I will have a public key; I encrypt using that public key whoever wants to take, you have a private key you decrypt with the private key; this is the basic infrastructure of what we call as public key cryptography. This public key cryptography is actually radically different approach than the symmetric key cryptography, wherein the sender and receiver has the same key.

So, in the public key cryptography the sender receiver do not share secret key that is the most important thing. If I start sharing secret key then the question come in when I have not even met you, a new router comes into the place and the old router wants to talk to the new router.

Now, how does it security send the key so, that nobody else actually finds out what the key is. So, that is the major challenge when we go for public key cryptography. So, in the if that is a major challenge when we go for symmetric key cryptography which now we want to get rid by having this public key cryptography. So, the Diffie Hellman and RSA are two interesting examples of public key cryptography. The public encryption key is known to all, while the private decryption key is known only to the receiver. So, this is where we stand with respect to the public key cryptography.

(Refer Slide Time: 07:30)



So, what happens here? So, I have a plaintext message m I do an encryption algorithm using a public key which I call it as K_B plus and this goes out and then there is K_B minus which Alice has; which is Bobs private key right. So, I; I used with the public key I; Bobs private key is there with K_B minus and so, Alice actually decrypt the cipher message which is K_B plus m ; using this K_B minus which is Bobs private key.

So, for all public encryption I can use the for anybody who wants to communicate with me; they can use my public key and then they need to have a private key which basically helps them get back the message. So, the public key and the private key are matched; they are actually you know one to one matched in such a way that; if I take the m and I encrypt using public key, we get K_B plus of m . I decrypt using the private key which is K_B minus of this. So, K_B minus of K_B plus of m should be back to m and this is what public key cryptography basically ensures you.

The another important property of RSA; which is a public as I told you RSA public key.

(Refer Slide Time: 08:50)

Public Key Cryptography

| <u>symmetric key crypto</u> | <u>public key cryptography</u> |
|---|--|
| <ul style="list-style-type: none">requires sender, receiver know shared secret keyQ: how to agree on key in first place (particularly if never "met")? | <ul style="list-style-type: none">radically different approach [Diffie-Hellman76, RSA78]sender, receiver do <i>not</i> share secret key<i>public</i> encryption key known to <i>all</i><i>private</i> decryption key known only to receiver |

NPTEL

The slide features a stick figure holding a lightbulb next to the public key cryptography section. The NPTEL logo is in the bottom left corner.

RSA78 is actually a public key cryptography system.

(Refer Slide Time: 08:55)


RSA: another important property

The following property will be *very* useful later:

$$K_B^-(K_B^+(m)) = m = K_B^+(K_B^-(m))$$

| | |
|---|---|
| use public key first, followed by private key | use private key first, followed by public key |
|---|---|

Result is the same!



And one of the very important property that RSA ensures is K_B^- of K_B^+ of m is equal to m is equal to K_B^+ of K_B^- of m .

So, whether I apply the private key first and then the public key; it is the same as applying the public key first and then the private key. So, whatever you see on the left hand side is used public key first followed by private key. So, K_B^+ of m followed by K_B^- , on the other hand use private key first K_B^- of m followed by public key. The result is actually the same in case of RSA; if your K is governed by RSA then the result of applying which one plus or minus first is the same. So, this is an interesting property that we will use later also.

Now, in addition; so, this we are talking about confidentiality the next thing that we want to talk about this integrity and authentication. There can be a message that I want to send from A to B anybody can read this message no problem, but when B receives it; he should have a definite decision that it has been received from a; nobody else has sent it on behalf of a and that the message is not tampered right.


These two are the next important integrity and authenticity authentication of the next two important things that we saw in the beginning of the previous lecture.

(Refer Slide Time: 10:21)

Digital Signatures

Cryptographic technique analogous to handwritten 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



So, importantly what we need to do here is that; I need to authenticate saying that this indeed it was a who has sent a message to B; who is interested in that? B is interested B is interested in find out that a as a sent this message; nobody else has sent that message and the second thing is that the message sent has not been tampered with.

So, for doing this the network community has lot of what we call as digital signatures. The digital signature is nothing, but you write a complete document; it will have several ascii characters the digital signature takes these ascii characters as input and create something called a signature through some algorithm. So, there is an algorithm called signature algorithm which will take all the alphabets, carries it and everything in your document; it will basically go through that algorithm and it will give you one signature.

Note that if you even change your document a little bit right if I say my name is Kamakoti; I just make it as my ame is Kamakoti; I just delete that n for example, my ame is Kamakoti then what happens is that the entire digital signature changes. So, when I had sent a message from here to B saying my name is Kamakoti and then I have put the digital signature corresponding to that and suppose somebody intermediately removes that n; the opponent party will get my ame is kamakoti without that n.

And then when he tries to sign it; he will get a signature he will signing is nothing, but giving the entire thing as input and getting generating an output which is called the

signature. So, when he does the same thing he will get a signature which is very different from whatever what I have put in the paper right; what I have put in the message.

So, when I say my name is Kamakoti; I put my name is Kamakoti along with the signature when the other party receives it; it takes the text and it will create the same signature and see if both signature matches. Even a small change in the data that is sent will create a large change in the signature; and by that what happens the opponent party will know oh there is some change that has happened. So, let us not believe in it.

One of; so, what we are talking now again please note that it is again from a forensic perspective right. The security perspective is nobody should be in a position to go and change it; what we are now talking of the use of distance again if somebody actually changes can I go and detect. So, please understand in this whole course I will always be talking about security and you know forensic as two different you know or on two different parts of your balance right.

So, one way is to see that nobody even changes your tempered say text that is called security, but if somebody changes that text I should find out and that forms subject matter of forensic. What we are talking in digital signature is basically from a forensic perspective all right. So, I will just go through this slide very quickly; the digital signature the; is a cryptographic technique analogous to handwritten signatures like I said the Kamakoti like that same thing.

Here the sender Bob basically digitally signs the document; establishing he is the document owner slash creator. And this the digital signature is both verifiable and also non forgeable meaning contemporary and the recipient Alice actually can prove to someone that Bob and no one else including Alice must have sign the document.

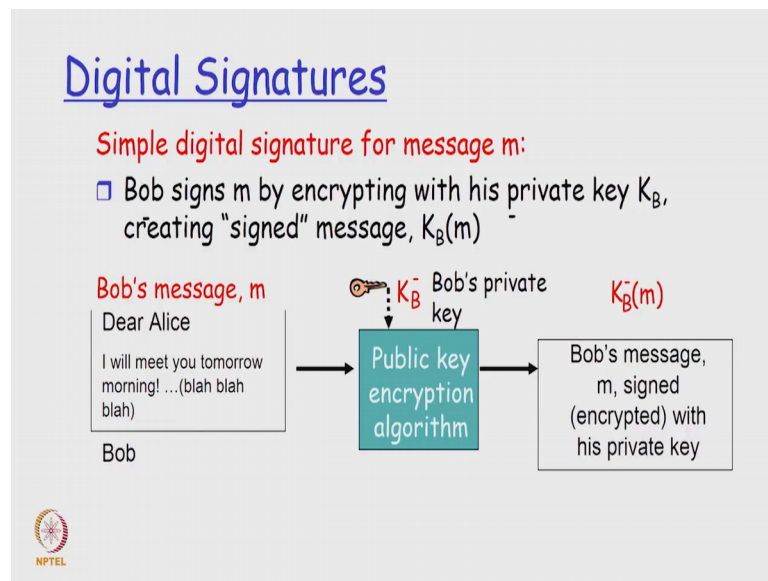
So, I received the document from somebody else and I have the digital signature, I can go and do a verification and say this has been indeed sent by this person and I have a legal through for it right. So, that is the most important thing; so, again I just quickly tell you a digital signature is nothing, but it is a cryptographic technique and analogous to hand written signature.

So, this is the sender actually takes the content of his whatever is sending and he digitally signs that document establishing that he is the document owner slash creator. When

somebody downloads this document, they again have the content they rerun the same thing and see if that word signatures match; if they mismatch then they know that somebody has tampered the document or whatever email or whatever the digital transmission happens.

Let us look at some simple forms of digital signatures.

(Refer Slide Time: 14:50)



The even the normal encryption itself is a digital signature for example, Bob signs m by encrypting with his private key; K_B creating a signed message $K_B m$ right. So, these are digital signature; so, what happens? Bob actually sends a message m say dear Alice I will meet you tomorrow morning etcetera Bob.

Now, this goes through the public key encryption algorithms by K_B minus and so, the Bobs message m which is signed which is private key; that is K_B minus m right. So, this encrypted itself is a part of a signature.

(Refer Slide Time: 15:27)

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 \bar{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 .



Suppose Alice receives a message m digitally signed with digital signature $K_B(m)$ as digital signature $K_B(m)$; Alice verifies m signed by Bob by applying Bob's public key to $K_B(m)$ and checks whether you know $K_B(m) + K_B(m) - m$ is equal to m .

So, there is a very simple thing; so, so, as per the public key cryptography $K_B(m) + K_B(m) - m$ should be equal to m . So, what Bob gets is; what Alice gets its $K_B(m) + m$ and he knows she knows $K_B(m) - m$. So, we will just do $K_B(m) - (K_B(m) + m)$ equal to m right. So, what we can do is that along with message m there is an encrypted value of message that can also be sent as a digital signature right and every fellow who receives it will receive the message m .

And if it wants to verify whether it has been sent can actually do this $K_B(m) - (K_B(m) + m)$ plus of a machine. So, this is a sort of a broadcast communication where I want to send my message to everyone; everyone can understand. So, I take the message m ; I send m itself (Refer Time: 16:44) along with this if somebody wants to find out whether I have send the message, then I also put I am Bob I also put $K_B(m) + m$.

So, what the other party receives? They receive m along with $K_B(m) + m$; if they just are interested in a message m they will read, but they want to find out whether Bob has sent; that means, I am Bob I have send it. So, they have my private key; so, they will do $K_B(m) - (K_B(m) + m)$ and then verify whether the message m they have got and

the output of K_B plus of K_B minus of m both coincides, if they coincide then they know that I have sent it.

So, the simple in public key encryption itself can be treated as a digital signature right. The difference between the public key encryption and this digital signature is; in the public key encryption I am interested in a point to point communication, but in this I am interested in a broadcast, using my public key I will sign it and send it also. So, the message is not encrypted; message goes it along with that the encrypted form of the message also goes and that is treated as a signature.

So, if somebody is interested; I repeat somebody is interested in the message they can just see the message and forget about it, but somebody is interested in finding out whether that message has been sent by somebody, but by say Bob then they can take the private key of Bob and put K_B minus of K_B plus of m and see that the message generated will be same as this message.

So, this is basically a very simple digital signature as a concept which you can get by public key cryptography. So, Alice can thus verify that Bob signed m ; no one else signed m and Bob signed m and not some other message m dash. This leads to a very important concept which is legally technically very important from a forensic point of view what we call as non repudiation.

What is mean by non repudiation? That is allies can take m and the signature K_B minus m to court and prove that Bob really signed that. So, if Bob has sent send its digitally signed a document to Alice and he says I am not done it then he can she can basically say that this is the message m as sent; this is this K_B plus m that is received that the signed and I have a secret key which we have agreed. Now I put that K_B minus of K_B plus of m and I get back the original m . So, it should be Bob that was signed it and nobody else.

So, on a court of law this becomes a valid proof and even from a cyber forensic point of view; somebody investigating it can use this to prove the whole thing. So, typically we need not encrypt the whole message that would be extremely complex and you need not send the two things are there; if I encrypt the entire message then that becomes extremely complex.

Because once if I am sending m ; I am sending m plus the encrypted value of m . So, signature need not be so, huge. So, people are interested in saying something called a message digest; message digest is nothing, but take the entire message; do some computation on it, get is a very smaller meaning and comparatively relatively smaller string which will be the digest of that message; which will reflect that message right.

(Refer Slide Time: 20:18)

Message Digests

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

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

Hash function properties:

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

The slide includes a diagram showing a box labeled 'large message m' with an arrow pointing to a box labeled 'H: Hash Function', which then has an arrow pointing down to a box labeled 'H(m)'. There is also an NPTEL logo in the bottom left corner.

So, this is basically what we may also call as a fingerprint. The fingerprint is the digest of the entire human body right. So, with the fingerprint I can uniquely identify whose fingerprint is; so, this is something like a unique identifier for the body. So, what we do? For this we use something called hash functions these hash functions need to satisfy certain properties.

We will look at some of the hash functions as we proceed; in the previous example what we did was we took the message and completely encrypted that the message using public key cryptography and then we said that that completely encrypted message was indeed the signature.

But in this case what we do is; we just take the message get it through an hash function whatever is the output of that hash function; now we say is the signature. Now this hash function basically gives the digest of that message that is what we say that it gives it is a good representative of the message. The hash function should also satisfy the property that if something small I changed there; something very minuet I changed there then this

the resulting hash. So, how did it what is that hash? Took an entire message and created a smaller string which we call it as the hash of that message or the digest of the message.

Now, if I change something small in that message very tiny very small thing then this digest should very back very big. So, if I have m I get H of m ; if I make a small change like m to m dash, your H of m dash should be much different than H m and this is something that we need to ensure right; so, this is basically the hash right. So, what we communicate is; we communicate the message along with the hash of that message. And the what is the hash algorithm? Everybody knows; on the other hand the person who receives it will get the message and if he wants to verify, he will apply the same hash function and see if this is indeed the message right.

So, that that is in this case what happens is I can prove that the message is not tamperable if something change in the message I know the hash will not agree, but I cannot prove who sent the message; which we could have proved in the earlier case. So, so this is the next stage many times I am interested in sending a communication to all right. And everybody is interested in the communication per se, they know who has sent it there is no many cases we do not have any doubt of who has sent it all right.

But is the content of what is sent is correct or not that is the more important question in going back to our original term terminology authentication is not very important, but the integrity of the communication is important. And if that comes this hash or this message digest becomes extremely useful.


Then given a message digest x for me to find a message which will a tampered message it will actually map onto the same digest is computationally infeasible. Meaning I have a message m ; I do H of m let me call it y , having another m dash such that the same H of m dash is equal to y is very difficult. So, compute it has been proved that it is computationally infeasible with even today's modern computers ok.

So; that means, if I receive a message m with this hash; there is no chance that somebody could have tampered. So, suppose I am receiving some message m with the hash there is no way by which temporarily could have tampered this message and get the same hash this is that that we rely on mathematics and we show that this is computationally infeasible. So, this is about message digest.

(Refer Slide Time: 24:33)

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 functions that are used for creating this message digest are the MD5 hash function which is widely used in the RFC1321 of the it is widely used in the networking community right. So, what does MD5 do? It computes a 128 bit message; they just in a simple four step process. And this arbitrary 128 bit string x appears difficult to construct message m whose MD5 act is also equal.

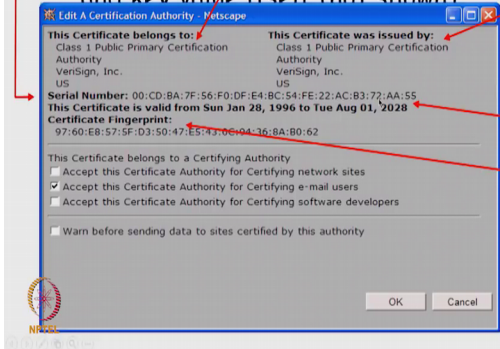
So, the MD5 basically gets back that same property right that if I have a message m which gives you a MD5 128 bit say x ; it is very difficult for me to find another m dash which will give you the same x . And by this I ensure that the m with x is a pair which cannot be which cannot be substitute; no one can be substituted I cannot make m with x dash or m dash with x both is not computationally feasible.

Similarly, SHA a; SHA 1 we call is also used this is a US standard NIST, FIPS, PUB all these things basically use this SHA 1 and this is a 160 bit message digest; well that is MD5 is a 128 bit message digest.

(Refer Slide Time: 25:51)

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 dialog box titled 'Edit A Certification Authority' with the following fields and options:

- 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:8A:7F:56:F0:DF:E4:BC:54:FE:22:AC:B3:77:AA:55
- This Certificate is valid from:** Sun Jan 28, 1996 to Tue Aug 01, 2028
- Certificate Fingerprint:** 97:60:E8:57:5F:D3:50:47:ES:42:0E:04:36:8A:80:62
- This Certificate belongs to a Certifying Authority:**
 - Accept this Certificate Authority for Certifying network sites
 - Accept this Certificate Authority for Certifying e-mail users
 - Accept this Certificate Authority for Certifying software developers
- Warn before sending data to sites certified by this authority

Buttons: OK, Cancel

Typically this is what you see; there is a certification authority that you see here. So, what you see here is first a thing that is a serial number which is unique to the issuer. So, somewhere you see this certificate for signing; these are the things that you should keep in mind. First thing is you see a serial number, which is unique to the issuer then there is something about some information about the certificate owner including the algorithm and key value itself right. So, the certificate belongs to so and. So, the substrate was right; so, so that is also here.

The next thing that you see is the info about the certificate yourself; this the who certificate issuer who has issued this certificate, there also talks about certain valid dates basically it says that it is valid from January 28, 1996 to August 01, 2020 and then it also gives you the digital signature by the issuer what is the digital signature. So, all these things form part of your digital signature right.

So to just quickly tell you; the serial number, which is unique to the issuer who has issued the information about this the information about certificate owner who belongs to this and then the valid dates and then the digital signature, actually the digital signature by the issuer. So, all these things can be taken care of.

Thank you.