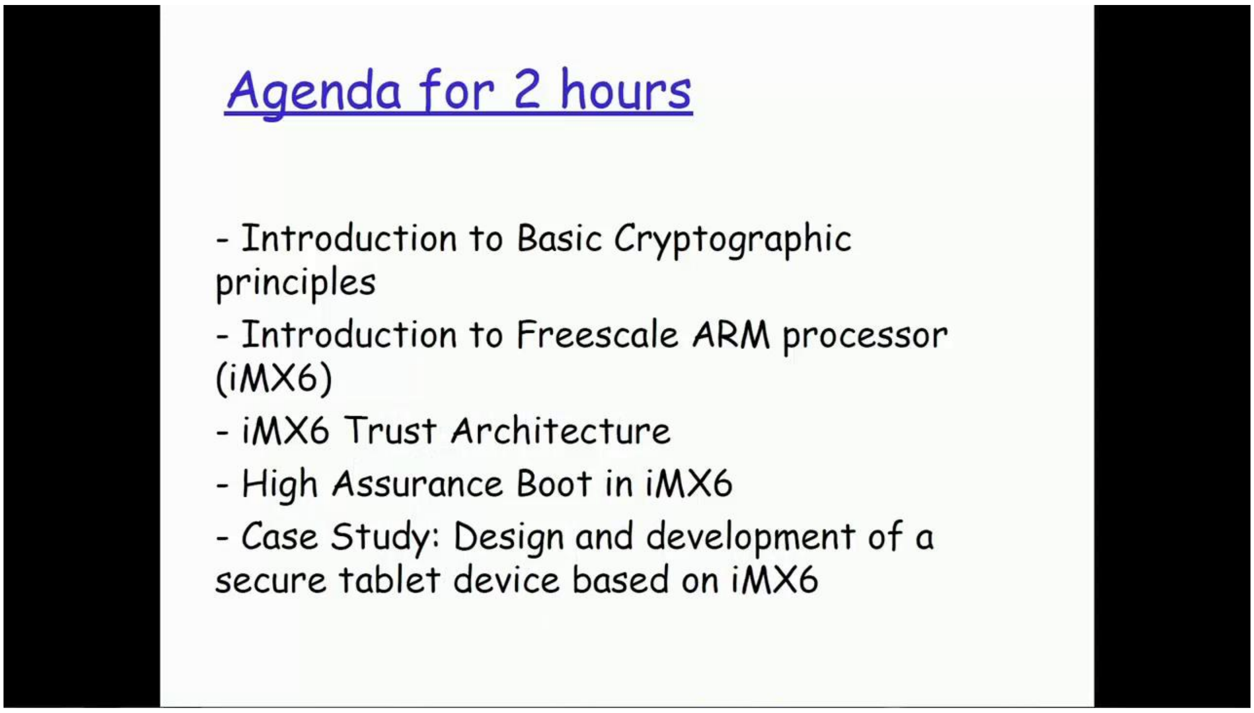
Information Security - II
Prof. V. Kamakoti
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Lecture – 35
ARM-based processor
Security features and implementation

We will actually, try to start off with very, very quick basic Introduction of the Basic Cryptographic principles.

(Refer Slide Time: 00:25)

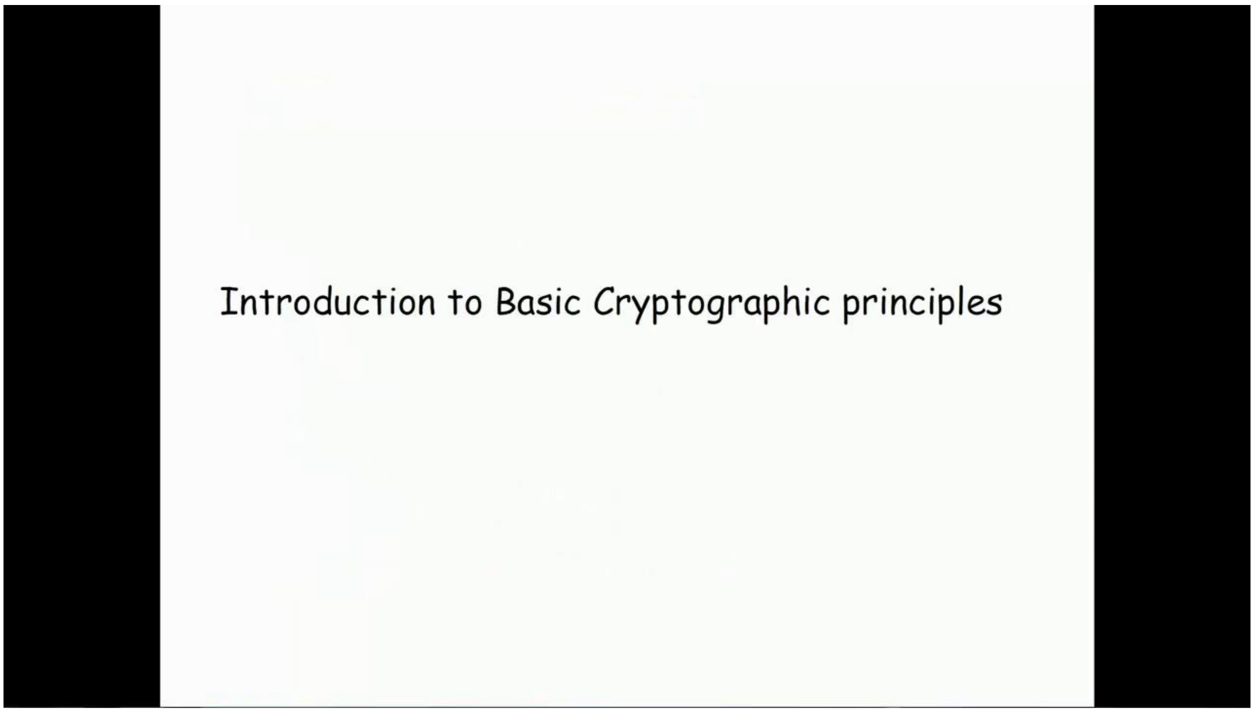


Agenda for 2 hours

- Introduction to Basic Cryptographic principles
- Introduction to Freescale ARM processor (iMX6)
- iMX6 Trust Architecture
- High Assurance Boot in iMX6
- Case Study: Design and development of a secure tablet device based on iMX6

Our focus for today's session, my session is going to be on, what are the security features that are actually available inside ARM processor? So, we are going to take ARM processor from free scale. You have the iMX class of processors of free scale, which is actually very popular and which is actually used in lot of handled devices as of today. Then we are going to look at a couple of security features and the architecture that iMX6 provide us. One which is basically iMX6 Trust Architecture, where I will talk about in very brief detail about, what are the different components inside the processor that helps us to run different kinds of security application by building them? Then finally, we will have very quick walk through of case study, where we have actually used the free scale iMX6 processor to build a secure device and I will just very briefly talk about what were our learning? What kind of mistakes we did? Why did we do the mistakes in the first place? And how did you go about correcting it? But those most like the case study just give in a idea right. So, with that in mind, we will actually start off with very brief introduction to some of the cryptographic principles.

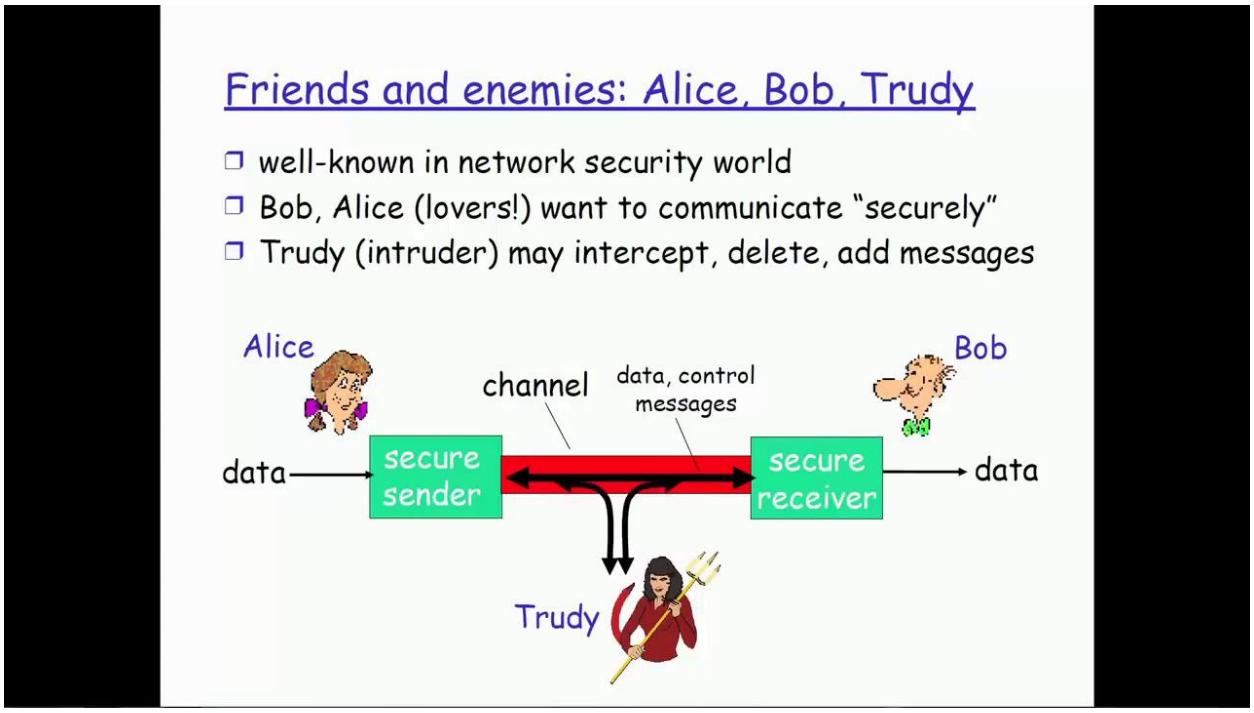
(Refer Slide Time: 02:07)



Introduction to Basic Cryptographic principles

Before, I actually go ahead with it, I just wanted know whether, are we aware of any kind of cryptographic algorithm till now that we have used or implemented or any kind of an application we can recall in which we have to use cryptography without which it will not work.

(Refer Slide Time: 02:28)



Friends and enemies: Alice, Bob, Trudy

- well-known in network security world
- Bob, Alice (lovers!) want to communicate "securely"
- Trudy (intruder) may intercept, delete, add messages

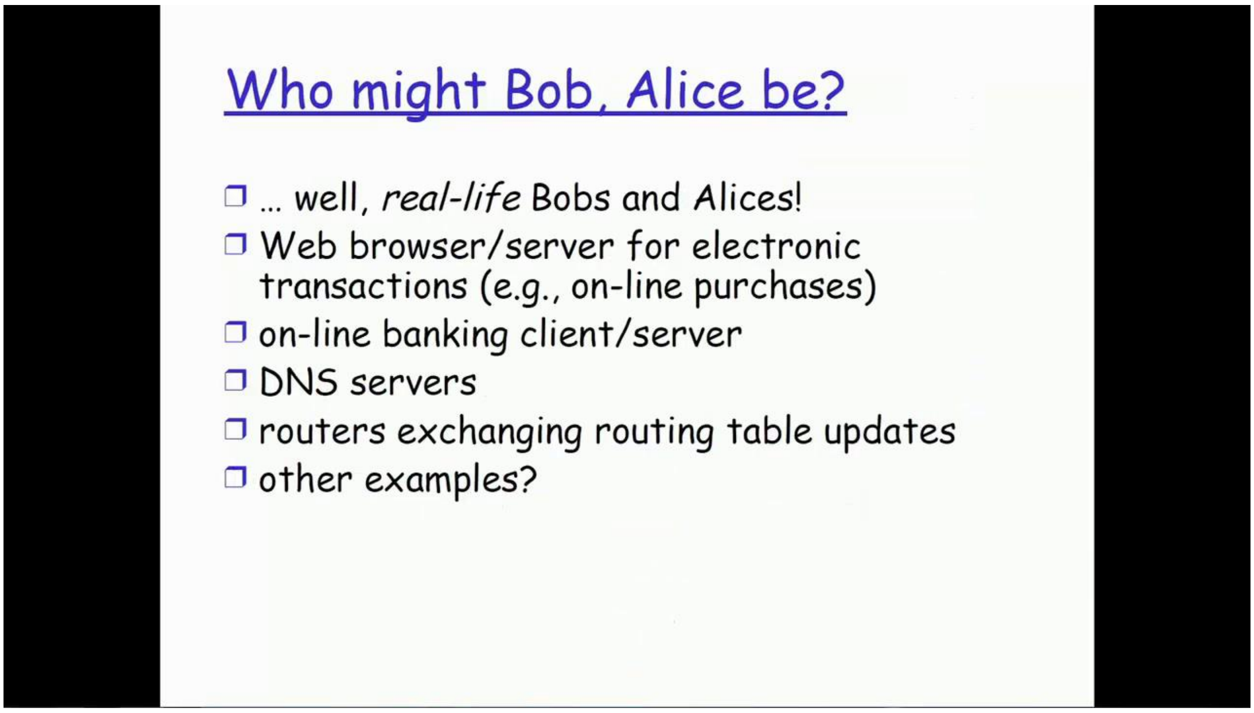
Alice
Bob
channel
data, control messages
data
secure sender
secure receiver
data
Trudy

So, any kind of thoughts or ideas on that? The moment we say security does any application come to your mind? Yeah. So, password is one, but anything beyond password, so basically accessing your bank account online, right? So, you really do not want to have your account numbers or your password or your pin numbers going in a free form, right? E-commerce sites, banking financial sites and those kind of things, right. So, essentially there are certain principals that are there, which would require to be available for different kinds of applications.

For example, look at three different people like Alice, Bob and Trudy. They could either be friends or they could be either enemy. So, we will actually look at few of the very commonly used security techniques that are implemented, so that the communication that actually goes between them is sort of protected. So, you are going to look at Alice and Bob being the relevant people trying to exchange some messages and then Trudy is basically a hacker or a impersonator, who wants to basically find out what are the messages that is actually going between these two people. So, we would really want a sender to be secure, we want the receiver to be secure and both of them are going to be actually communicating over a channel.

The idea here, why I am basically talking about this is without having a very brief overview, about the typical security techniques and the protocols that are there, we are not going to be in the position to appreciate, what is it that is actually available inside the processor? So, I just thought I will take this small example and then, try to walk through different kind of techniques that are required.

(Refer Slide Time: 05:06)



Who might Bob, Alice be?

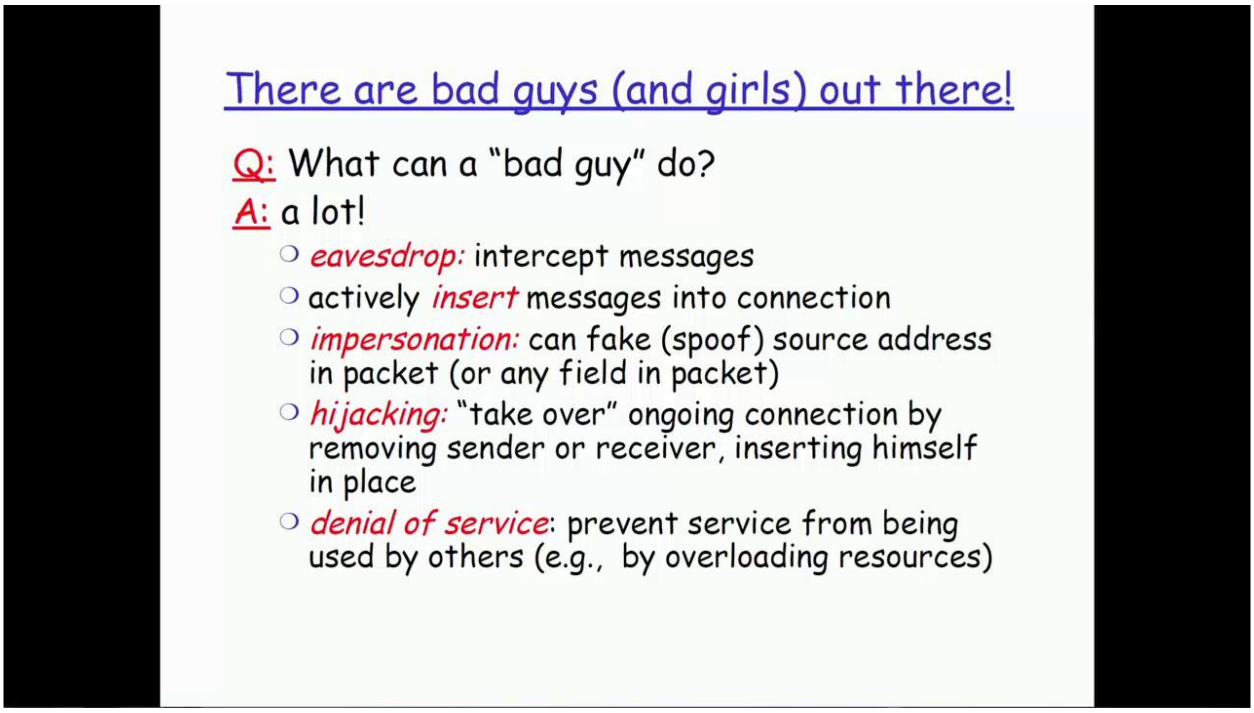
- ... well, *real-life* Bobs and Alices!
- Web browser/server for electronic transactions (e.g., on-line purchases)
- on-line banking client/server
- DNS servers
- routers exchanging routing table updates
- other examples?

So, in the real world, who are we actually trying to map Bob and Alice to? There like tons of possibilities, they could be real life persons. So, two people trying to communicate and their intention is that, they want to have a secure communication being done. Alternatively, these two nodes could actually be a web browser and a server. The example, that we talked about right now, a banking application. So, you are actually having, let us say account in your ICICI bank, which you want to access online and you want to do it in a secure manner. So, the way you are going to do access is that, you are going to open up your browser like your IE or Firefox or whatever it is and then trying to connect to your banks web server and you want that entire communication to be completely secure. It could be online banking client server, very similar this thing it could be DNS servers.

Why would you want a DNS server to communicate over a secure connection? So, if you recall, you have been actually trying to be updated with the latest things that are happening in the security world. A few months back, access to YouTube.com landed into Pakistan that is the last place you want to have it. So, how did it happen? There were some nodes that were actually compromised because of which an incorrect DNS update could have been easily spread out. So, that is again another reason, why you would need to have security between DNS server.

Routers exchanging routing table updates, this again makes it very inductive. Another related example is there was one person hanged recently, a very wanted criminal, if you remember, there was calls that was actually being made from some place and the call was being monitored by the intelligence authority but the problem was, every few minutes the IP address of the call was actually getting changed. The call was not getting dropped but the IP address was getting changed, so with the net result that the actual physical place from where, the call was originated could never be tracked till the call got over. That is again because of the fact that, if I am able to have certain nodes which I know are sending routing updates without complete security inbuilt into it, those are those nodes that I can very easily compromise on to do all these kind of things and I could keep giving so many such example, but just quick overview, right.

(Refer Slide Time: 08:00)



There are bad guys (and girls) out there!

Q: What can a "bad guy" do?

A: a lot!

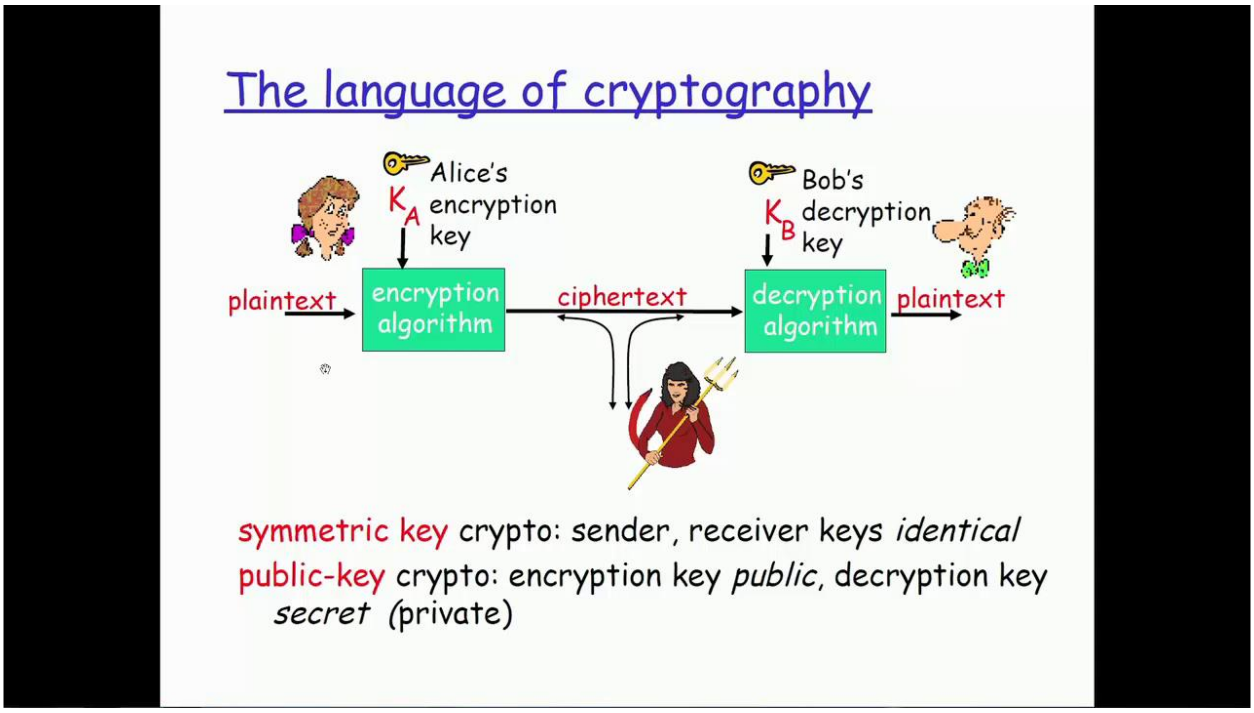
- *eavesdrop:* intercept messages
- actively *insert* messages into connection
- *impersonation:* can fake (spoof) source address in packet (or any field in packet)
- *hijacking:* "take over" ongoing connection by removing sender or receiver, inserting himself in place
- *denial of service*: prevent service from being used by others (e.g., by overloading resources)

So, now there are bad guys and bad girls out there. So, why do they want to really do that? We are going to see so many complicated, sophisticated features that we are going to talk subsequently inside the processor, but for what. So, why do not we actually allow unwarranted people trying to listen to the conversation trying to act as somebody else, trying to modify the data and then induce it? What is the problem? So, they can do actually a lot of damage, the basic thing is eavesdrop. So, eavesdropping essentially means it is read only. So, I am not going to be announcing myself that boss, I am here, but I am a very tacit person in the channel just getting the messages out whatever is actually coming on the channel between any parties. If we have actually heard of some tools like ethereal or wire shark. If I just install the tool and I have an administrator privilege on the machine in which I have installed the tool, I do not need really a doctorate degree. I do not need to be a research student to understand from the output to the tool which two guys are talking? What are they talking? What are they exchanging? Which could be actually very, very sensitive information, so I could be just listening then other next possibility is actively insert messages into connection.

So, I am getting to be more and more nastier from being just a passive listener to get the data out I am going to start acting little bit more smarter and then say boss, I am now going to keep adding or inserting new messages in to it, most possibly not as myself but as somebody else. So, that is basically coming down to the impersonation. So, although I am Vasan, I will act as XYZ and also ensure the other party is getting conversant, but I am really XYZ and not Vasan. So, that is basically what does impersonation can fake source address in a packet or any field in packet to actually act as if I am somebody else. Typically, the very crude method of identifying who the person is on a network at least based on the IP address. So, if I basically spoof my IP address to be something else, it becomes very easy.

Hijacking that is the next worst case scenario, where I would be able to take over an ongoing connection by removing sender or receiver, inserting himself in place and also ensuring that the other party does not realize that the original guy has been removed out of the system right and then finally, what is call as a denial of service, you might have heard it, in short form as dos. Have you heard about something called d dos distributed denial of service? So, originally started as denial of service, where my availability was the under question and then when the firewalls starting getting more and more intelligent the hackers started getting more intelligent, parallel. So, they said, boss now you are tracking me coming from single source, I will attack you from multiple sources. Then the firewalls again, started getting intelligent and said boss, irrespective of the source, I am going to now track who is coming in at what point in time and we have a threshold limit over which, I will not allow anybody else to come in again for certain period of time. So, these are the different nasty things that somebody can do if they are really motivated.

(Refer Slide Time: 12:28)



The language of cryptography

symmetric key crypto: sender, receiver keys *identical*

public-key crypto: encryption key *public*, decryption key *secret* (private)

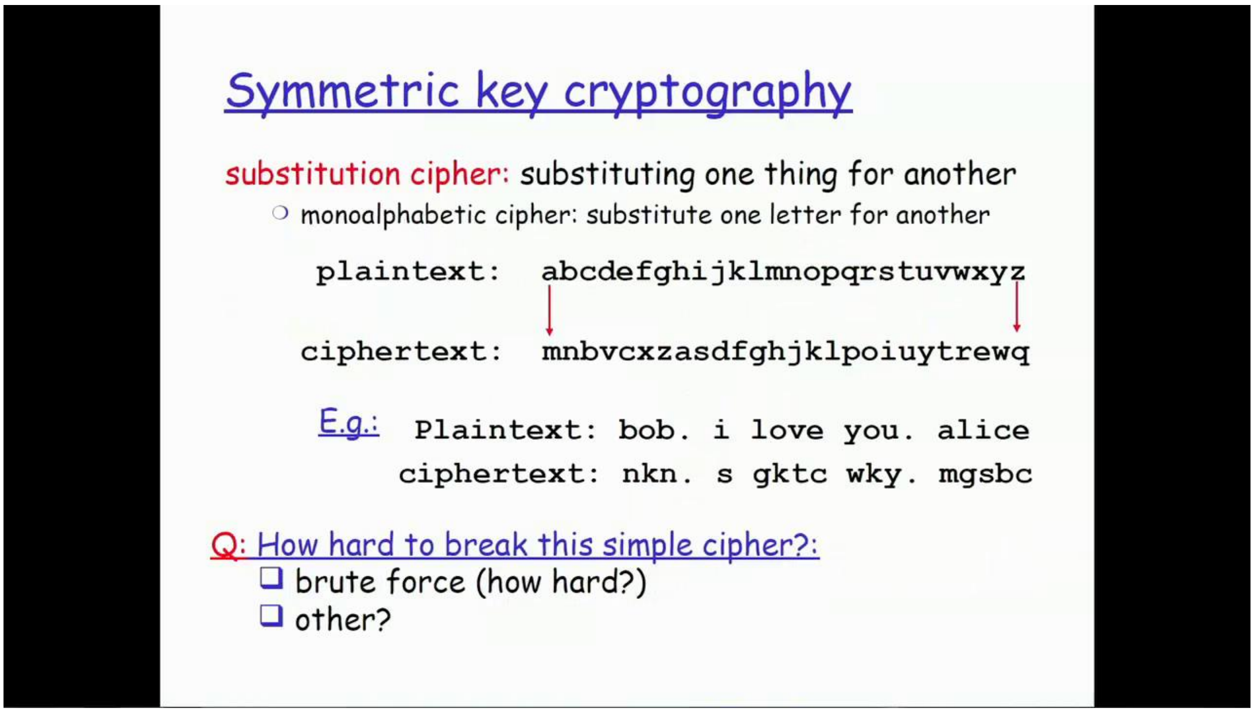
You have a symmetric key cryptography, you have a public cryptography. So, in a symmetric key cryptography, what is actually going to happen is, both sides Bob and Alice, who is basically are the valid parties. They are basically going to be actually using the same key right, the key that is used on the sending side for encryption. Encryption is a process where I basically convert plain text in to cipher text. So, when we say cipher text, what we essentially mean is, unless and until I am going to have the key that cipher text is going to be absolutely useless to me I cannot make head or tail, out of what is there actually is part of the cipher text.

When I say symmetric cryptography, symmetric meaning as we know is basically same. I am going to really have both parties, who want to establish the channel for communication to be actually making use of the same key right. So, that the person who is actually encrypting the data before it is sent on the channel is using one particular key and the person who is receiving the data in cipher text form will be using the same key, apply on the cipher text and then get back magically the original plain text. So, that is basically symmetric key whereas, when I say it is a public key, it is asymmetric key. So, when we say asymmetric key, I am going to use one key for encryption and another key for decryption right. So, when you are using it for this confidentiality kind of a purpose we use what is called as a public key for doing the encryption and we use a private key for what we call as decryption.

So, what is meant by public key is the public key for a particular node or a user is something which is actually known publicly. So, everybody on the entire universe who wants to communicate with me will have access to my public key. But the private key is something which I alone will know. So, if I am going to be having a private key the private key is not disclosed to anybody else, it is a secret. Generally in normal parlance, we generally share secrets. So, whoever is our closest friend, we basically go and tell him or her that boss, I got access to some very secret information I am just sharing it with you do not tell it to somebody else, but in the computer security world secret means it is not known to the second person. The moment it is known to the second person, it is no longer to be considered as a secret. So, you have to go redo the entire generation of the keys completely from scratch.

The private key which is typically used for decryption will be known only to me and the public key which is actually use for encryption will be known to everybody and this also facilitates the process because somebody is sending me data, they need to know my public key for encrypting and that can be decrypted only by my private key and that private key by principle it supposed to be known only to me. So, there it facilitates a process of not really requiring having the same keys on both sides. So, that is basically what is called as a symmetric key or a public key cryptography.

(Refer Slide Time: 16:09)



Symmetric key cryptography

substitution cipher: substituting one thing for another

- monoalphabetic cipher: substitute one letter for another

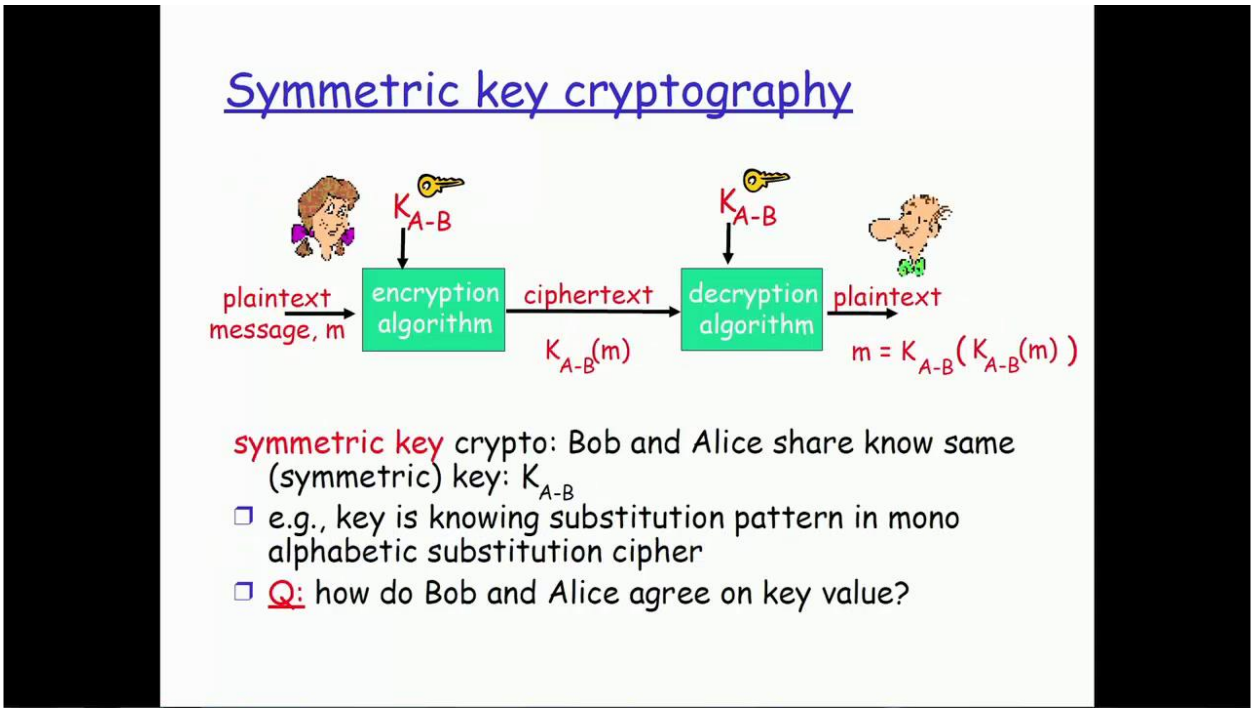
```
plaintext:   abcdefghijklmnopqrstuvwxyz

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, there are like different ways of symmetric key cryptography. I just have a substituting algorithm. Where I substitute one letter by another letter and then convert it as a plain as a cipher text in it send it out.

(Refer Slide Time: 16:24)



Symmetric key cryptography

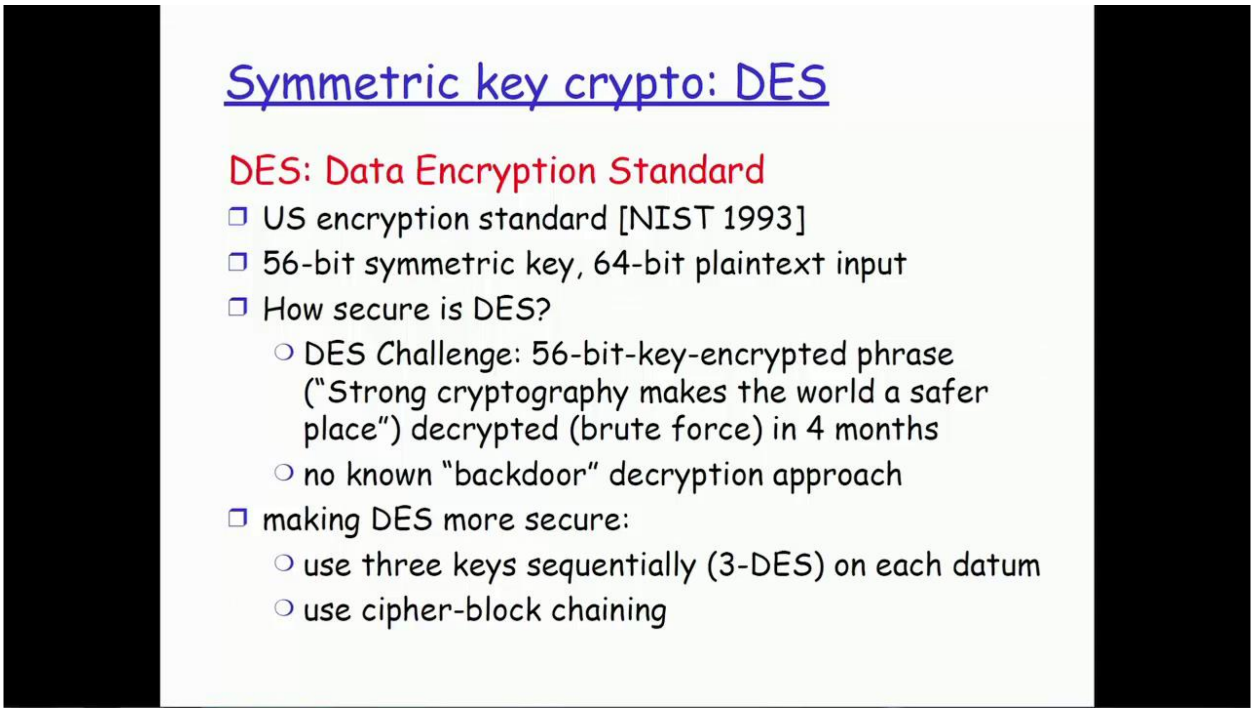
symmetric key crypto: Bob and Alice share know same (symmetric) key: K_{A-B}

- e.g., key is knowing substitution pattern in mono alphabetic substitution cipher
- Q: how do Bob and Alice agree on key value?

Basically, the whole thing that is actually working is, there is a plain text message here. So, I have a key which is actually shared between Alice and Bob, which is K A of B. There is an encryption algorithm. We will see few sample algorithms right now, but the encryption algorithm is going to basically take two inputs. So, the first input is basically what is the text? And then the second input is basically what the key is? So, based on these two data inputs that the algorithm is given, he is going to magically generate a cipher text right and this cipher text is basically the message on which my key has been applied through this algorithm and then the cipher text is going to go over the channel to the recipient, where there is going to a corresponding decryption algorithm.

The difference between encryption and decryption algorithm is whatever actually happens on encryption side, the reverse actually happens on my decryption side till it finally generates back my original plain text. So, the same key because it is as in a symmetric key algorithm is actually applied on the cipher text and then it generates back the original plain text. So, this is the cipher text, I applied the same key on this cipher text, I get back magically the original message. So, here the same key is actually shared between the two parties that actually want to communicate.

(Refer Slide Time: 18:00)



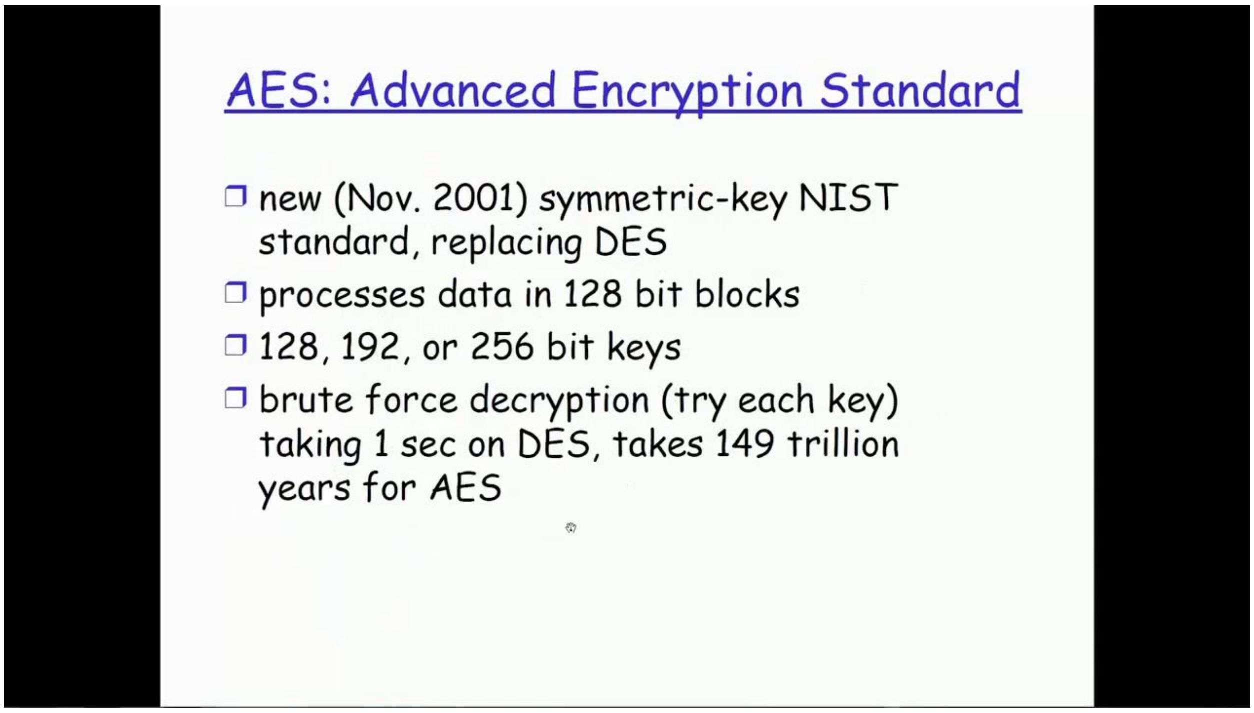
Symmetric key crypto: DES

DES: Data Encryption Standard

- US encryption standard [NIST 1993]
- 56-bit symmetric key, 64-bit plaintext input
- How secure is DES?
  - DES Challenge: 56-bit-key-encrypted phrase ("Strong cryptography makes the world a safer place") decrypted (brute force) in 4 months
  - no known "backdoor" decryption approach
- making DES more secure:
  - use three keys sequentially (3-DES) on each datum
  - use cipher-block chaining

So, some of the very commonly used symmetric cryptographic algorithms is DES, that is the first standard that actually came in for symmetric key. It actually uses a 64-bit key out of which only 56-bits are used for the calculation. So, it is actually very sequential algorithm that is actually done through. So, currently with the algorithm of DES, we will actually be able to apply a brute force method and then decrypt it back in around 4 months time. So, with the kind of processing power that we have within 4 months time, if I have not changed the key, I would be able to apply a brute force guess method that is why you call it actually is a brute force method. We will be able to generate back the original this thing by applying the brute force. So, in order to make it more secure you have further versions of DES called as 3-DES and AES that is actually come out.

(Refer Slide Time: 19:03)

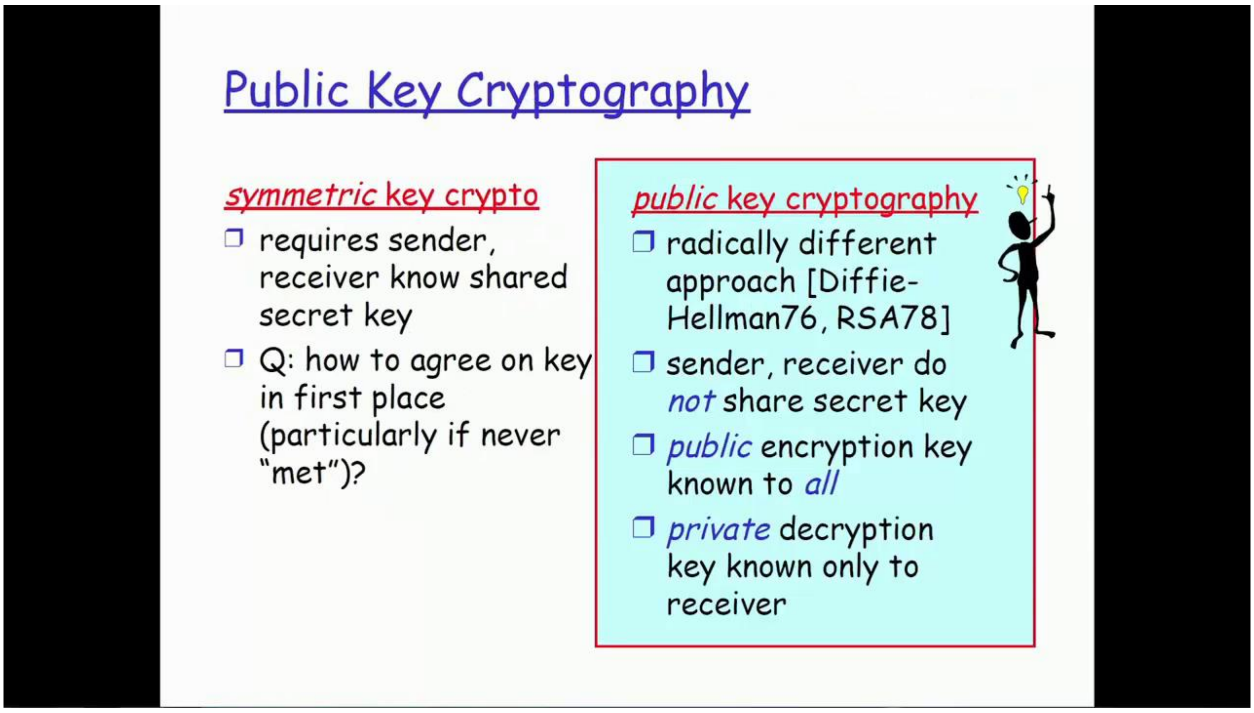


AES: Advanced Encryption Standard

- new (Nov. 2001) symmetric-key NIST standard, replacing DES
- processes data in 128 bit blocks
- 128, 192, or 256 bit keys
- brute force decryption (try each key) taking 1 sec on DES, takes 149 trillion years for AES

With an AES, you really have very large sized keys, which actually goes around 256 bit keys as compared to the 64 bit keys of the DES algorithm. So, what will actually take one second on DES actually takes 149 trillion years for AES as a sort of a simple of comparison method. So, that is one of predominant reasons, why when you use the symmetric key algorithm, you always find today AES being the preferred method. So, you have a DES, you have tripled DES, you have AES and so on. But, AES is right now with the technology and with the processing power that is actually available is sort of safe in terms of the guessing of the key in a brute force method or generating the plain text back and brute force method.

(Refer Slide Time: 19:49)



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, with a symmetric cryptography as we have been saying, I need to have the same shared key. The problem comes in on how I share the key. So, can I write it in the plain text of paper and then give it to him or courier it to him or can I call him up and then tell him that this is the key that I am going to make use of. So, is that a hung parliament between yes and no, I see some people saying yes and see some people saying no. So many different Medias available, now I can send a Facebook message, I can send a Whatsapp message, I can send a normal sms message itself. For me to send a Facebook message or a Whatsapp message or a sms message, I might as well send a plain text message itself. So, because if I am basically going to be announcing the key in this manner, what prevents the interested person of, who is actually desperately trying to find out what is it the I am communicating with another party XYZ also interpret the same message and get to know the key.

There is a fundamental problem that I have there are solutions for it, but those solutions are actually very costly. So, when I say costly, what I mean is in terms of the amount of the time that it is going to take for me to transfer the key in a very secure manner to the other party right. That is basically, why I came out, why they have actually come out with the public cryptography, where the public key I openly communicate saying that boss over wants to take this can take it but for me to have the message decrypted successfully back to the original message, I can do it only when somebody as access to the private key. So, somebody has sent me a message, you are very desperate to get access to the message that somebody has sent me. You cannot do anything with that message because you would not be having access to my private key right. So, there by the problem that I actually have of sharing the key is sort of overcome.