






Syllabus Content:

3.5 Security




3.5.1 Asymmetric keys and encryption methods

-  show understanding of the terms: public key, private key, plain text, cipher text, encryption and asymmetric key cryptography
-  show understanding of how the keys can be used to send a private message from the public to an individual/organization
-  show understanding of how the keys can be used to send a verified message to the public




3.5.2 Digital signatures and digital certificates

-  show understanding of how a digital certificate is acquired
-  show understanding of how a digital certificate is used to produce digital signatures

3.5.3 Encryption protocols

-  show awareness of the purpose of Secure Socket Layer (SSL)/Transport Layer Security (TLS)
-  show awareness of the use of SSL/TLS in client-server communication
-  show awareness of situations where the use of SSL/TLS would be appropriate

3.5.4 Malware

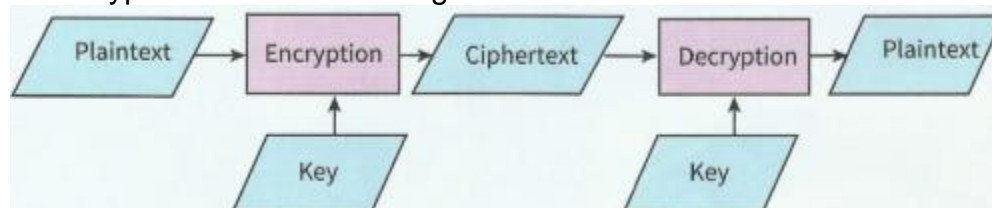
-  show understanding of malware: viruses, spyware, worms, phishing, pharming
-  describe vulnerabilities that the various types of malware can exploit
-  describe methods that can be used to restrict the effect of malware

Security:

Encryption fundamentals

Encryption can be used as a routine procedure when storing data within a computing system. However, the focus in this chapter is on the use of encryption when transmitting data over a network.

The use of encryption is illustrated in figure.



The process starts with original data referred to as plaintext, whatever form it takes. This is encrypted by an encryption algorithm which makes use of a key. The product of the encryption is **ciphertext**, which is transmitted to the recipient. When the transmission is received it is decrypted using a decryption algorithm and a key to produce the original **plaintext**.

Security concerns

There are a number of security concerns relating to a transmission:



Confidentiality: Only the intended recipient should be able to decrypt the ciphertext.



Authenticity: The receiver must be certain who sent the ciphertext.



Integrity: The ciphertext must not be modified during transmission.



Non-repudiation: Neither sender nor receiver should be able to deny involvement in the transmission.



Availability: Nothing should happen to prevent the receiver from receiving the transmission.

This chapter will consider only confidentiality, authenticity and integrity.

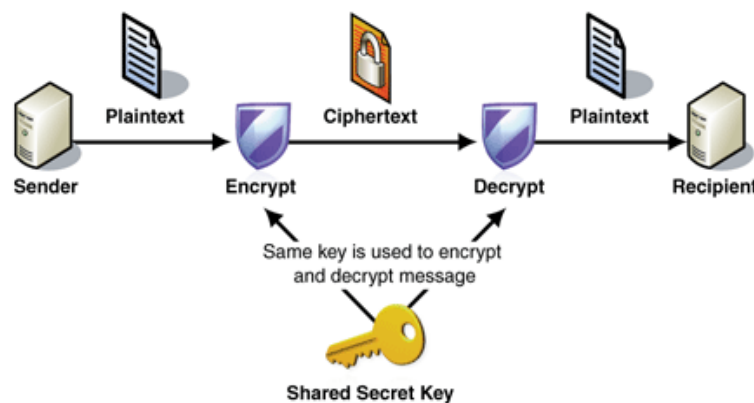
The confidentiality concern arises because a message could be intercepted during transmission and the contents read by an unauthorised person. The concern about integrity reflects the fact that the transmission might be interfered with deliberately but also that there might be accidental corruption of the data during transmission.

Encryption methods

SYMMETRIC ENCRYPTION

SYMMETRIC ENCRYPTION is a secret key which can be a combination of characters. If this key is applied to a message, its content is changed which makes it unreadable unless the recipient also has the decryption key.

One key is needed to encrypt a message and same key is needed to decrypt a message. It is obviously important that the sender and receiver have the same encryption and decryption key. There is clearly a security risk here, since the sender has to supply the key to the recipient. This key could be intercepted by, for example, a hacker which puts the security of the encrypted message at risk. This situation is referred to as the **KEY DISTRIBUTION PROBLEM**

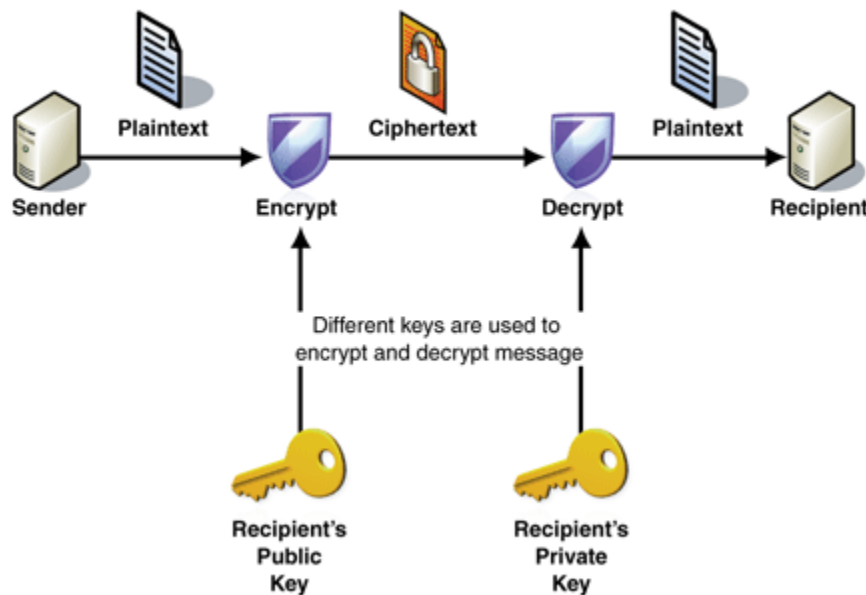


If symmetric key encryption is used, there needs to be a secure method for the sender and receiver to be provided with the secret key.

ASYMMETRIC ENCRYPTION

- Using **asymmetric** key encryption, the process actually starts with the **receiver**.
- The receiver must be in possession of two keys.** One is a public key which is not secret. The other is a **private key** which is secret and **known only to the receiver**.
- The **receiver** can send the **public key** to a **sender**, who uses the public key for encryption and sends the ciphertext to the receiver.
- The **receiver** is the only person who can decrypt the message because the private and public keys are a matched pair.
- The public key can be provided to any number of different people allowing the receiver to receive a private message from any of them.
- Note, however, that if two individuals require **two-way communication**, both communicators need a private key and must send the matching public key to the other person.

There are two requirements to ensure confidentiality should the transmission be intercepted and the message extracted: the encryption algorithm must be complex and the number of bits used to define the key must be large.



Asymmetric cryptography, also known as public key cryptography, uses public and private keys to encrypt and decrypt data.

Digital signatures and digital certificates

Using asymmetric encryption, the **decryption- encryption** works if the keys are used the other way round.



An individual can **encrypt a message** with a **private key** and send this to many recipients who have the corresponding **public key** and can therefore **decrypt** the message.



This approach would **not be used** if the content of a **message was confidential**.



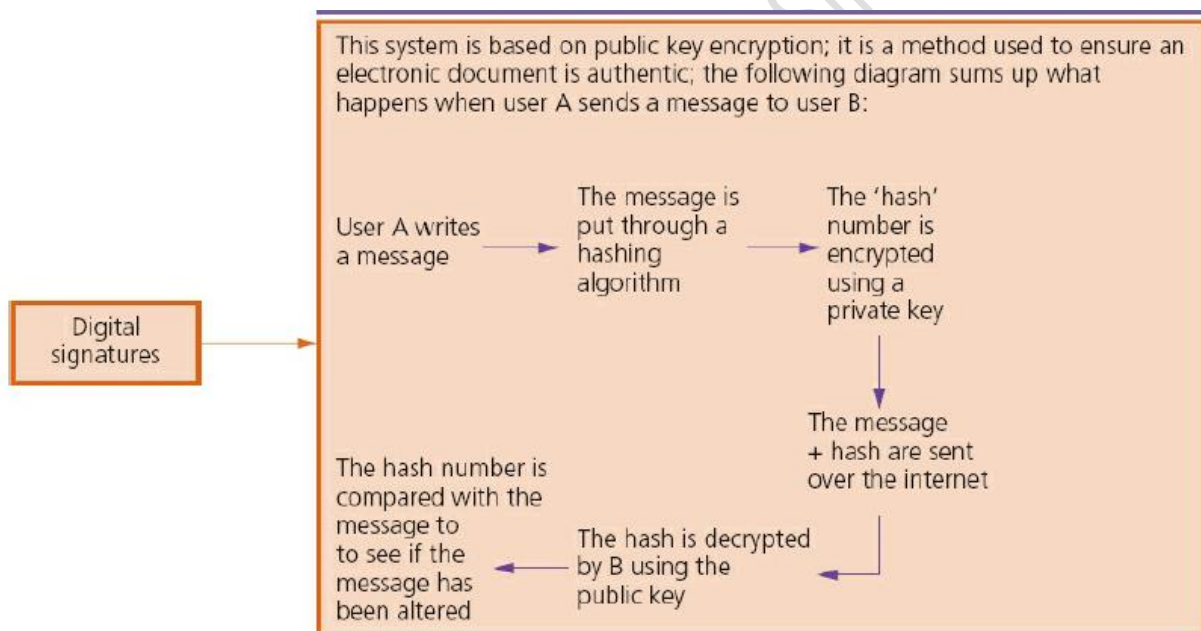
However, it could be used if it was important to verify who the sender was.



Only the **sender** has the **private key** and the **public keys** only work with that one specific private key.



Therefore, used this way, the message has a digital signature identifying the sender.



There is a disadvantage in using this method of applying a digital signature in that it is associated with an encryption of the whole of a message.



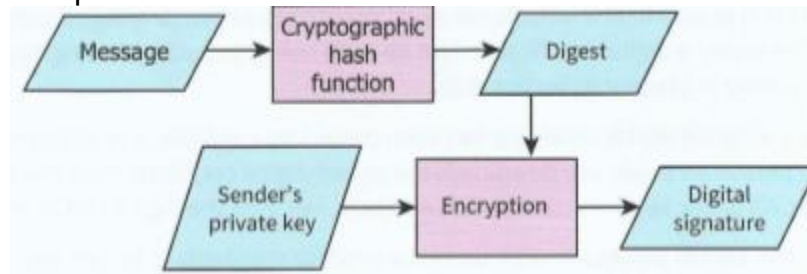
An alternative is to use a cryptographic one-way hash function which creates from the message a number, uniquely defined for the particular message, called a '**digest**'.



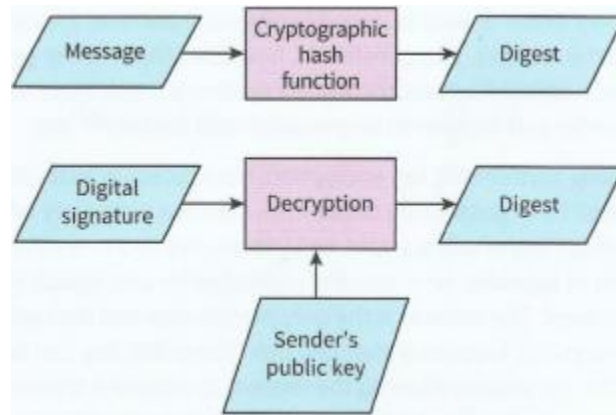
The private key is used as a **signature for this digest**. This speeds up the process of confirming the sender's identity.



The process at the sender's end of the transmission is outlined in Figure



We will assume that the message is transmitted as plaintext together with the digital signature as a separate file. The processes that take place at the receiver end are outlined in Figure



The same public hash key function is used that was used by the sender so the same digest is produced if the message has been transmitted without alteration.



The decryption of the digital signature produces an identical digest if the message was genuinely sent by the original owner of the public key that the receiver has used.



This approach has allowed the receiver to be confident that the message is both authentic and unaltered.



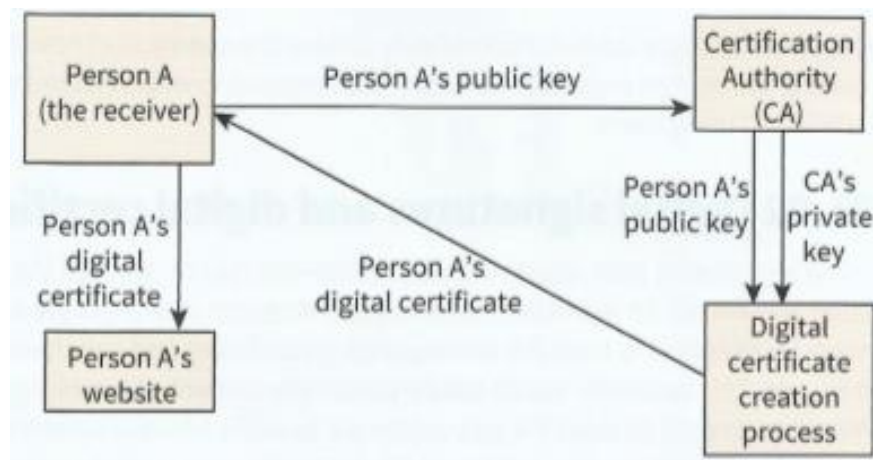
This sounds good but unfortunately it does not consider the fact that someone might forge a public key and pretend to be someone else.










Therefore, there is a need for a more rigorous means of ensuring authentication. This can be provided by a **Certification Authority (CA)** provided as part of a **Public Key Infrastructure (PKI)**.

Let's consider **A** would-be receiver who has a **public- private key pair**. This individual wishes to be able to receive secure messages from other individuals. The public key must be made available in a way that ensures authentication.



The steps taken by the **A** would-be receiver to obtain a digital certificate to allow safe public key delivery are illustrated in Figure



-  An individual (**person A**) who would-be receiver and has a **public-private key pair** contacts a local **CA**.
-  The **CA** confirms the identity of person A.
-  Person **A's public key** is given to the **CA**.
-  The **CA** creates a **public-key certificate (a digital certificate)** and writes person **A's public key** into this document.
-  The CA uses **encryption with the CA's private key** to add a **digital signature** to this document.
-  The **digital certificate** is given to person A.
-  Person A posts the digital certificate on a website.

Security protocols

We will now consider two forms of security protocols when using the internet:

-  Secure Sockets Layer (SSL)
-  Transport Layer Security (TLS).

SECURE SOCKETS LAYER (SSL)

SECURE SOCKETS LAYER (SSL) is a type of protocol (a set of rules used by computers to communicate with each other across a network).



This allows data to be sent and received securely over the internet.



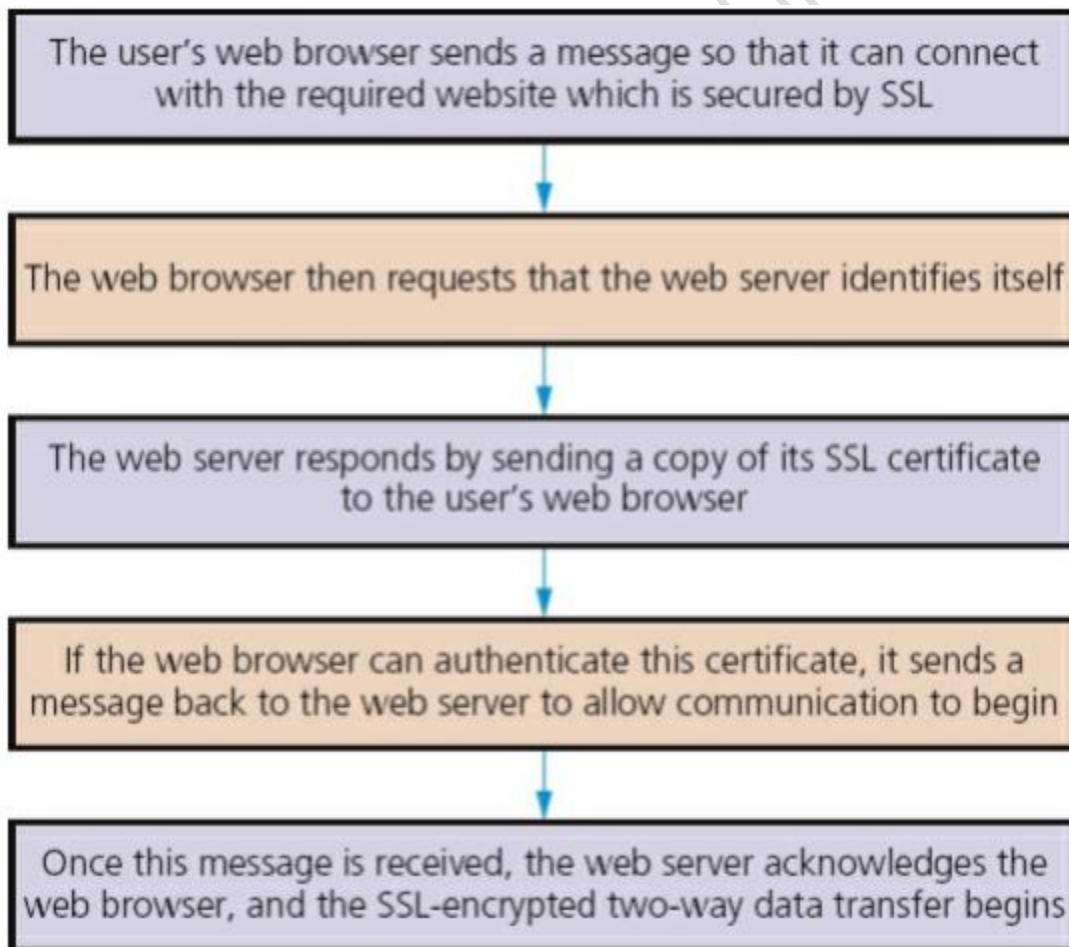
When a user logs onto a website, SSL encrypts the data – only the user's computer and the web server are able to make sense of what is being transmitted.



A user will know if SSL is being applied when they see https or the small padlock in the status bar at the top of the screen.



So what happens when a user wants to access a secure website and receive and send data to it?



TRANSPORT LAYER SECURITY (TLS) is similar to SSL but is a more recent security system. TLS is a slightly modified version of SSL

TRANSPORT LAYER SECURITY (TLS)

TLS is a form of protocol that ensures the security and privacy of data between devices and users when communicating over the internet. It is essentially designed to provide encryption, authentication and data integrity in a more effective way than its predecessor SSL.



When a website and client (user) communicate over the internet, TLS is designed to prevent a third party hacking into this communication causing problems with data security.



TLS is formed of two layers:

- **Record protocol:** this part of the communication can be used with or without encryption (it contains the data being transferred over the internet).
- **Handshake protocol:** this permits the website and the client (user) to authenticate each other and to make use of encryption algorithms (a secure session between client and website is established).



Only the most recent web browsers support both SSL and TLS which is why the older SSL is still used in many cases.



But what are the main differences between SSL and TLS since they both effectively do the same thing?



It is possible to extend TLS by adding new authentication methods.



TLS can make use of **SESSION CACHING** which improves the overall performance.



Once the session has been established, the client and server can agree which encryption algorithms are to be used and can define the values for the session keys that are to be used.



This interchange may involve **checking digital certificates**. For the transmission, SSL provides encryption, compression of the data and integrity checking.



When the transmission is complete the **session is closed** and **all records of the encryption disappear**









An application running **HTTPS** can guarantee secure communication allowing users to send confidential information such as **credit card details** in an ecommerce transaction. The user is completely unaware of the processes involved in ensuring confidential transmission with data integrity assured.

Malware




Types of malware

Malware is the colloquial name for malicious software. Malicious software is software that is introduced into a system for a harmful purpose. One category of malware is where program code is introduced to a system. The various types of malware-containing program code are:

-  **Virus:** tries to replicate itself inside other executable code
-  **Worm:** runs independently and propagates to other network hosts
-  **Logic bomb:** lies dormant until some condition is met
-  **Trojan horse:** replaces all or part of a previously useful program
-  **Spyware:** collects information and transmits it to another system
-  **Bot:** takes control of another computer and uses it to launch attacks.





The differences between the different types are not large and what is always called an '**antivirus**' package will detect all of the different types. The virus category is often subdivided according to the software that the virus attaches itself to. Examples are boot sector viruses and macro viruses.

Malware can also be classified in terms of the activity involved:

-  **Phishing:** sending an email or electronic message from an apparently legitimate source requesting confidential information
-  **pharming:** setting up a bogus website which appears to be a legitimate site
-  **Keylogger:** recording keyboard usage by the legitimate user of the system.

System vulnerabilities

Many system vulnerabilities are associated directly with the activities of legitimate users of a system. Malware can be introduced inadvertently by the user in a number of ways:

-  attaching a portable storage device
-  opening an email attachment
-  accessing a website
-  Downloading a file from the Internet.

Alternatively, a legitimate user with a grievance might introduce malware deliberately.

Other vulnerabilities are indirectly associated with the activities of legitimate users.



By far the most significant is the use of weak passwords and particularly those which have a direct connection to the user.



A poor choice of password gives the would-be hacker a strong chance of being able to gain unauthorised access. Other examples include a legitimate user not recognising a phishing or pharming attack and, as a result, disclosing sensitive information.



Systems inherently lack optimum security.



Operating systems are notorious for lacking good security. There is a tendency for operating systems to increase in complexity which tends to offer the potential for more insecurity.



The regular updates are often required because of a newly discovered security vulnerability. In the past, commonly used application packages have allowed macro viruses to flourish but this particular problem is largely under control.

A very specific vulnerability is buffer overflow. Programs written in the C programming language, of which there are very many, do not automatically carry out array bound checks.



A program can be written to deliberately write code to the part of memory that is outside the address range defined for the array implemented as a buffer.



This will overwrite what is stored there so when a subsequent program reads this overwritten section it will not execute as it should.



This might just cause minor disruption but if cleverly engineered it could lead to an attacker gaining **unauthorised** access to the system and causing serious problems.

References:



Computer Science Course book by Sylvia Langfield & Dave Duddell



IGCE Computer Science by David Watson & Helen Williams



<https://crypto.stackexchange.com/questions/42363/how-does-the-receiver-compute-the-private-key-when-using-rsa>



<https://support.microsoft.com/en-us/kb/246071>



https://en.wikipedia.org/wiki/Public-key_cryptography