NETWORK SECURITY ESSENTIALS

BCA – IV

Credits – 4

Evaluations – 5

The course is designed to build an understanding of various network security components, protocols and creating the awareness about the issues due to security.

Pre-requisites: An understanding of Basic Computer Networking and security

UNIT 5

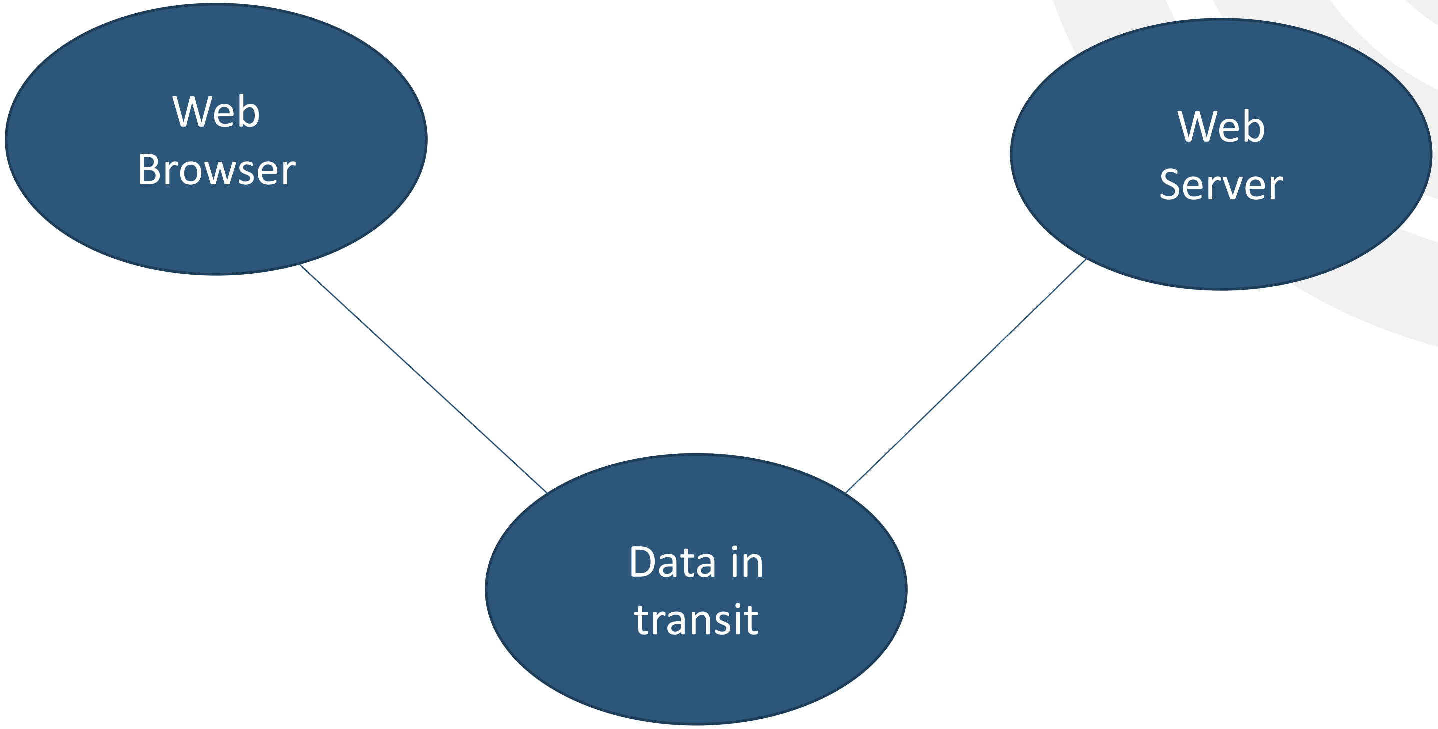
Security protocols: IPsec, SSL/TLS, PGP and S/MIME, Kerberos, SSH

WEB SECURITY REQUIREMENT
SSL (SECURE SOCKET LAYER)
TLS (TRANSPORT LAYER SECURITY)

- Definition

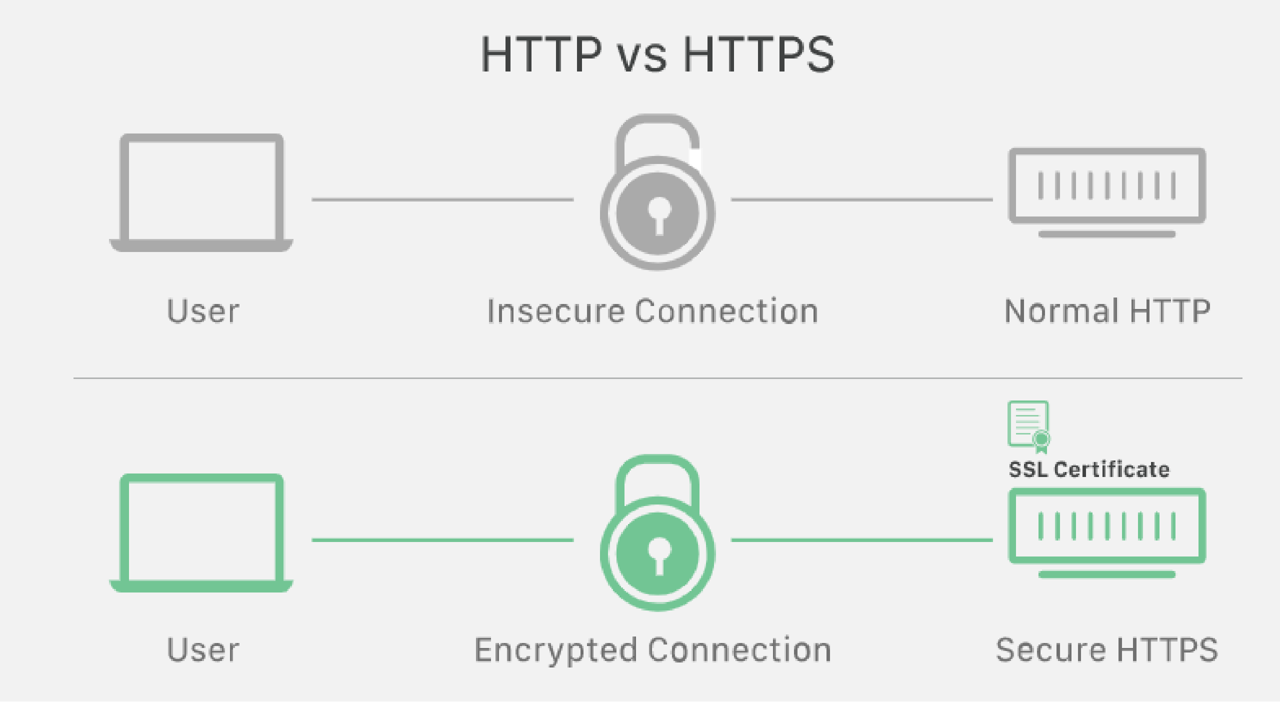
A set of procedures, practices, and technologies for assuring the reliable, predictable operation of web servers, web browsers, other programs that communicate with web servers, and the surrounding Internet infrastructure.

- Three components



What is SSL?

SSL, or Secure Sockets Layer, is an encryption-based Internet security protocol. It was first developed by Netscape in 1995 for the purpose of ensuring privacy, authentication, and data integrity in Internet communications. SSL is the predecessor to the modern TLS encryption used today.



SSL

- What does SSL provide us?
  - Data integrity, Confidentiality
  - Authentication(handshake)
- Limitation on SSL
  - doesn’t work with connection less protocol
  - doesn’t support non-repudiation
  - doesn’t protect the application itself
  - general-purpose data security

How does SSL/TLS work?

•In order to provide a high degree of **privacy**, SSL encrypts data that is transmitted across the web. This means that anyone who tries to intercept this data will only see a garbled mix of characters that is nearly impossible to decrypt.

•SSL initiates an **authentication** process called a handshake between two communicating devices to ensure that both devices are really who they claim to be.

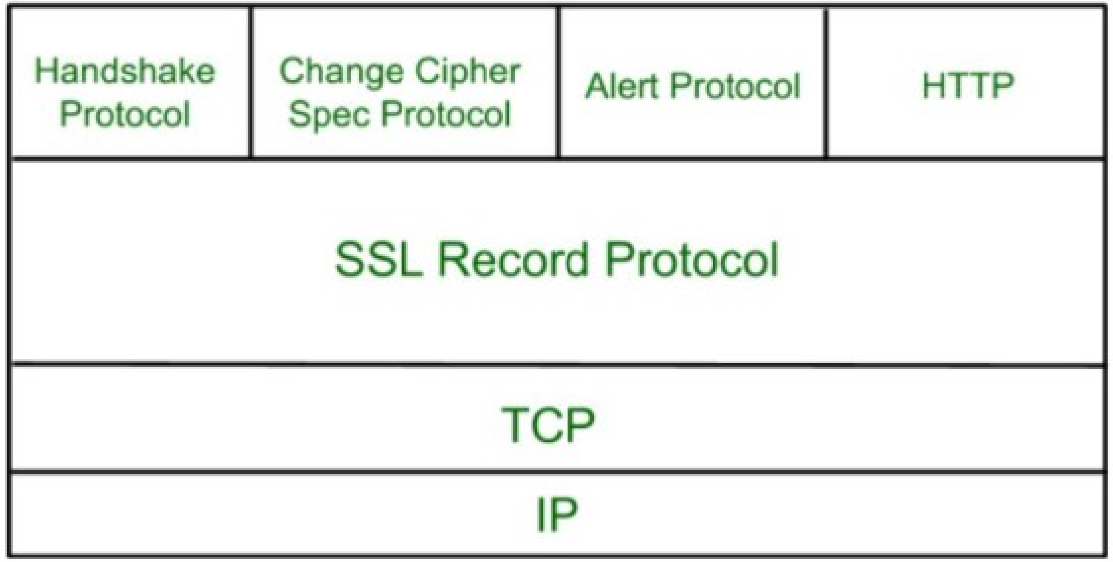
•SSL also digitally signs data in order to provide **data integrity**, verifying that the data is not tampered with before reaching its intended recipient. There have been several iterations of SSL, each more secure than the last. In 1999 SSL was updated to become TLS.

Secure Socket Layer (SSL) provide security to the data that is transferred between web browser and server. SSL encrypt the link between a web server and a browser which ensures that all data passed between them remain private and free from attack.

Secure Socket Layer Protocols:

- SSL record protocol
- Handshake protocol
- Change-cipher spec protocol
- Alert protocol

SSL Protocol Stack:



Handshake Protocol | Change Cipher Spec Protocol | Alert Protocol | HTTP

SSL Record Protocol

TCP

IP

Handshake Protocol:
Handshake Protocol is used to establish sessions. This protocol allow client and server to authenticate each other by sending a series of messages to each other. Handshake protocol uses four phases to complete its cycle.

- **Phase-1:** In Phase-1 both Client and Server send hello-packets to each other. In this IP session, cipher suite and protocol version are exchanged for security purpose.

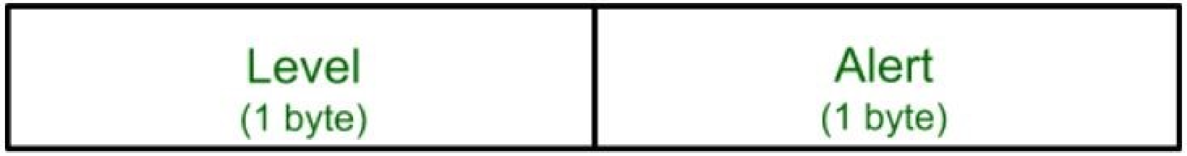
- **Phase-2:** Server send his certificate and Server-key-exchange. Server end the phase-2 by sending Server-hello-end packet.

- **Phase-3:** In this phase Client reply to the server by sending his certificate and Client-exchange-key.

- **Phase-4:** In Phase-4 Change-cipher suite occurred and after this Handshake Protocol ends.

Change-cipher Protocol:
This protocol uses SSL record protocol. Unless Handshake Protocol is completed, the SSL record Output will be in pending state. After handshake protocol the Pending state is converted into Current state. Change-cipher protocol consists of single message which is 1 byte in length and can have only one value. This protocol purpose is to cause the pending state to be copied into current state.

Alert Protocol:
This protocol is used to convey SSL-related alerts to the peer entity. Each message in this protocol contain 2 bytes.



Level (1 byte)	Alert (1 byte)

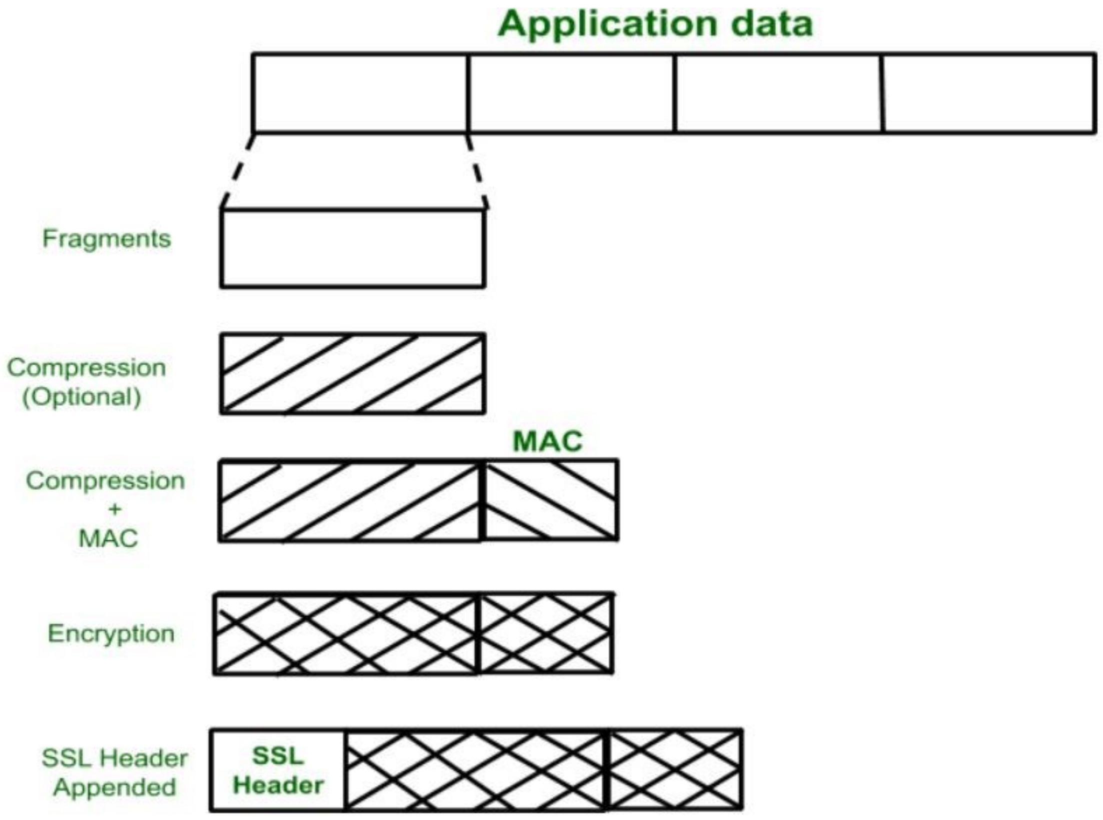
Level is further classified into two parts:
•**Warning:**
This Alert have no impact on the connection between sender and receiver.
•**Fatal Error:**
This Alert breaks the connection between sender and receiver.

SSL Record Protocol:

SSL Record provide two services to SSL connection.

- Confidentiality
- Message Integrity

In SSL Record Protocol application data is divided into fragments. The fragment is compressed and then encrypted MAC (Message Authentication Code) generated by algorithms like SHA (Secure Hash Protocol) and MD5 (Message Digest) is appended. After that encryption of the data is done and in last SSL header is appended to the data.



Silent Features of Secure Socket Layer:

- Advantage of this approach is that the service can be tailored to the specific needs of the given application.
- Secure Socket Layer was originated by Netscape.
- SSL is designed to make use of TCP to provide reliable end-to-end secure service.
- This is two-layered protocol.

SESSIONS AND CONNECTIONS

- an SSL session is an association between a client and a server
- sessions are stateful; the session state includes security algorithms and parameters
- a session may include multiple secure connections between the same client and server
- connections of the same session share the session state
- sessions are used to avoid expensive negotiation of new security parameters for each connection
- there may be multiple simultaneous sessions between the same two parties, but this feature is not used in practice

Are SSL and TLS the same thing?

SSL is the direct predecessor of another protocol called TLS (Transport Layer Security). In 1999 the Internet Engineering Task Force (IETF) proposed an update to SSL. Since this update was being developed by the IETF and Netscape was no longer involved, the name was changed to TLS.

Since they are so closely related, the two terms are often used interchangeably and confused, because SSL still has so much name recognition.

Is SSL still up to date?

SSL has not been updated since SSL 3.0 in 1996 and is now considered to be deprecated. There are several known vulnerabilities in the SSL protocol, and security experts recommend discontinuing its use.

TLS is the up-to-date encryption protocol that is still being implemented online, even though many people still refer to it as "SSL encryption." The truth is that any vendor offering "SSL" these days is almost certainly providing TLS protection, which has been an industry standard for over 20 years.

Transport Layer Security (TLS)

Transport Layer Securities (TLS) are designed to provide security at the transport layer. TLS was derived from a security protocol called Secure Service Layer (SSL). TLS ensures that no third party may eavesdrops or tamper with any message.

There are several benefits of TLS:

•**Encryption:**

TLS/SSL can help to secure transmitted data using encryption.

•**Interoperability:**

TLS/SSL works with most web browsers, including Microsoft Internet Explorer and on most operating systems and web servers.

•**Algorithm flexibility:**

TLS/SSL provides operations for authentication mechanism, encryption algorithms and hashing algorithm that are used during the secure session.

•**Ease of Deployment:**

Many applications TLS/SSL temporarily on a windows server 2003 operating systems.

•**Ease of Use:**

Because we implement TLS/SSL beneath the application layer, most of its operations are completely invisible to client.

TLS Basics

Transport Layer Security (TLS) encrypts data sent over the Internet to ensure that eavesdroppers and hackers are unable to see what you transmit which is particularly useful for private and sensitive information such as passwords, credit card numbers, and personal correspondence.

What is TLS?

TLS is a cryptographic protocol that provides end-to-end security of data sent between applications over the Internet.

It should be noted that TLS does not secure data on end systems. It simply ensures the secure delivery of data over the Internet, avoiding possible eavesdropping and/or alteration of the content.

TLS operates on a reliable transport, such as TCP, and is itself layered into

- TLS Record Protocol
- TLS Handshake Protocol

How does TLS work?

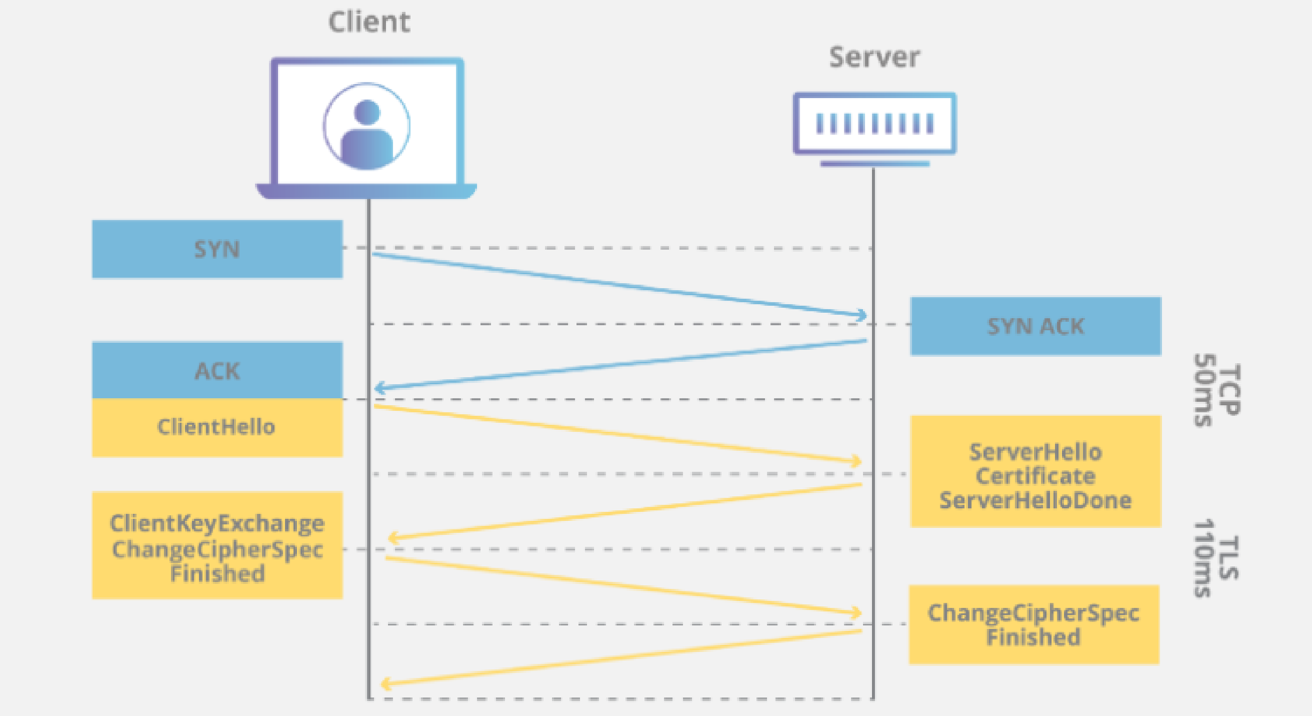
TLS uses a combination of symmetric and asymmetric cryptography, as this provides a good compromise between performance and security when transmitting data securely.

With symmetric cryptography, data is encrypted and decrypted with a secret key known to both sender and recipient; typically 128 but preferably 256 bits in length (anything less than 80 bits is now considered insecure). Symmetric cryptography is efficient in terms of computation, but having a common secret key means it needs to be shared in a secure manner.

Asymmetric cryptography uses key pairs – a public key, and a private key. The public key is mathematically related to the private key, but given sufficient key length, it is computationally impractical to derive the private key from the public key. This allows the public key of the recipient to be used by the sender to encrypt the data they wish to send to them, but that data can only be decrypted with the private key of the recipient.

How does TLS work?

For a website or application to use TLS, it must have a TLS certificate installed on its origin server (the certificate is also known as an "SSL certificate" because of the naming confusion described above). A TLS certificate is issued by a certificate authority to the person or business that owns a domain. The certificate contains important information about who owns the domain, along with the server's public key, both of which are important for validating the server's identity.



A TLS connection is initiated using a sequence known as the TLS handshake. When a user navigates to a website that uses TLS, the TLS handshake begins between the user's device (also known as the *client* device) and the web server.

During the TLS handshake, the user's device and the web server:

- Specify which version of TLS (TLS 1.0, 1.2, 1.3, etc.) they will use
- Decide on which cipher suites they will use
- Authenticate the identity of the server using the server's TLS certificate
- Generate session keys for encrypting messages between them after the handshake is complete

The handshake also handles authentication, which usually consists of the server proving its identity to the client.

Once data is encrypted and authenticated, it is then signed with a message authentication code (MAC). The recipient can then verify the MAC to ensure the integrity of the data.

This is kind of like the tamper-proof seal found on a water-bottle; the consumer knows no one has tampered with the bottle the seal is intact.

TLS Record Protocol

The *Transport Layer Security* (TLS) Record protocol secures application data using the keys created during the Handshake. The Record Protocol is responsible for securing application data and verifying its *integrity* and origin.

It manages the following:

- Dividing outgoing messages into manageable blocks, and reassembling incoming messages.
- Compressing outgoing blocks and decompressing incoming blocks (optional).
- Applying a *Message Authentication Code* (MAC) to outgoing messages, and verifying incoming messages using the MAC.
- Encrypting outgoing messages and decrypting incoming messages.

When the Record Protocol is complete, the outgoing encrypted data is passed down to the Transmission Control Protocol (TCP) layer for transport.

- Advantage of TLS
  - applications can use it transparently to securely communicate with each other
  - TLS is visible to applications, making them aware of the cipher suites and authentication certificates negotiated during the set-up phases of a TLS session

IPSEC – INTERNET SECURITY

The **IP security (IPSec)** is an Internet Engineering Task Force (IETF) standard suite of protocols between 2 communication points across the IP network that provide data authentication, integrity, and confidentiality. It also defines the encrypted, decrypted and authenticated packets. The protocols needed for secure key exchange and key management are defined in it.

Internet Protocol Security (IPsec) is a secure network protocol suite that authenticates and encrypts the packets of data to provide secure encrypted communication between two computers over an Internet Protocol network. It is used in virtual private networks (VPNs)

IPsec includes protocols for establishing mutual authentication between agents at the beginning of a session and negotiation of cryptographic keys to use during the session. IPsec can protect data flows between a pair of hosts (host-to-host), between a pair of security gateways (network-to-network), or between a security gateway and a host (network-to-host).

IPsec uses cryptographic security services to protect communications over Internet Protocol (IP) networks. It supports network-level peer authentication, data origin authentication, data integrity, and data confidentiality (encryption).

IPsec is a layer 3 OSI model or internet layer end-to-end security scheme.

USES OF IP SECURITY

IPsec can be used to do the following things:

•To encrypt application layer data.

•To provide security for routers sending routing data across the public internet.

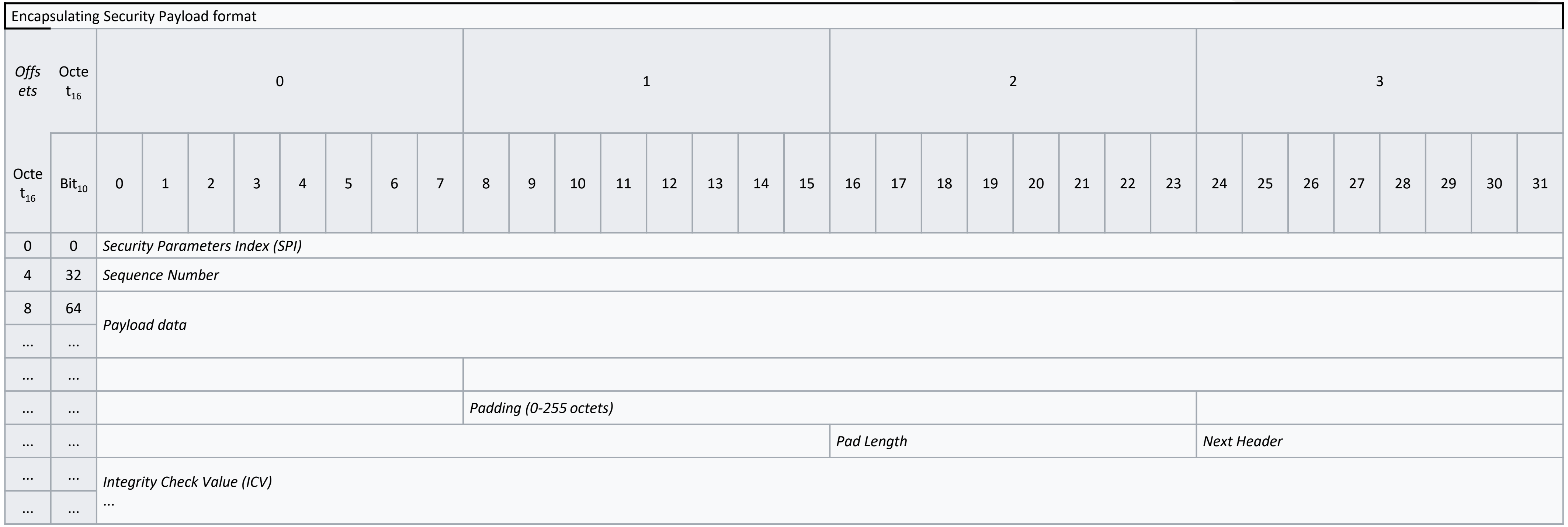
•To provide authentication without encryption, like to authenticate that the data originates from a known sender.

•To protect network data by setting up circuits using IPsec tunneling in which all data is being sent between the two endpoints is encrypted, as with a Virtual Private Network(VPN) connection.

COMPONENTS OF IPSEC

1. Encapsulating Security Payload (ESP) –

It provides data integrity, encryption, authentication and anti replay. It also provides authentication for payload. It provides origin authenticity through source authentication, data integrity through hash functions and confidentiality through encryption protection for IP packets. ESP also supports encryption-only and authentication-only configurations, but using encryption without authentication is strongly discouraged because it is insecure.

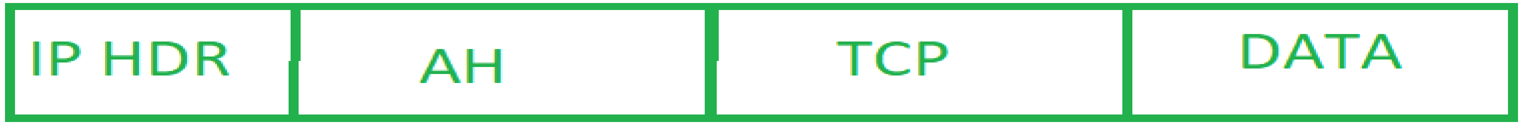


Encapsulating Security Payload format

Offsets	Octet$_{16}$	0								1								2								3							
Octet$_{16}$	Bit$_{10}$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	*Security Parameters Index (SPI)*																															
4	32	*Sequence Number*																															
8	64	*Payload data*																															
…	…																																
…	…																																
…	…									*Padding (0-255 octets)*																							
…	…																	*Pad Length*								*Next Header*							
…	…	*Integrity Check Value (ICV)*																															
…	…	…																															

2. Authentication Header (AH)

It also provides data integrity, authentication and anti replay and it does not provide encryption. The anti replay protection, protects against unauthorized transmission of packets. It does not protect data’s confidentiality.



IP HDR | AH | TCP | DATA

Authentication Header (AH) is a member of the IPsec protocol suite. AH ensures connectionless integrity by using a hash function and a secret shared key in the AH algorithm. AH also guarantees the data origin by authenticating IP packets. Optionally a sequence number can protect the IPsec packet's contents against replay attacks using the sliding window technique and discarding old packets.

In IPv4, AH prevents option-insertion attacks.
In IPv6, AH protects both against header insertion attacks and option insertion attacks.
In IPv4, the AH protects the IP payload and all header fields of an IP datagram except for mutable fields (i.e. those that might be altered in transit)

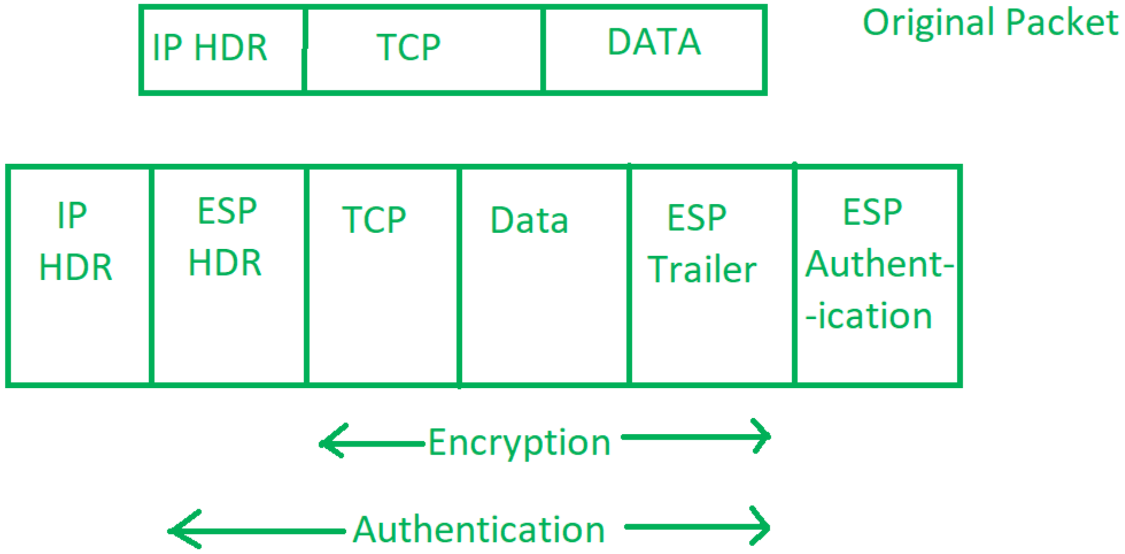
A replay attack (also known as playback attack) is a form of network attack in which valid data transmission is maliciously or fraudulently repeated or delayed. This is carried out either by the originator or by an adversary who intercepts the data and re-transmits it, possibly as part of a spoofing attack by IP packet substitution. This is one of the lower-tier versions of a man-in-the-middle attack. Replay attacks are usually passive in nature.

3. Internet Key Exchange (IKE) –

It is a network security protocol designed to dynamically exchange encryption keys and find a way over Security Association (SA) between 2 devices.

The Security Association (SA) establishes shared security attributes between 2 network entities to support secure communication. The Key Management Protocol (ISAKMP) and Internet Security Association which provides a framework for authentication and key exchange. ISAKMP tells how the set up of the Security Associations (SAs) and how direct connections between two hosts that are using IPsec.

Internet Key Exchange (IKE) provides message content protection and also an open frame for implementing standard algorithms such as SHA and MD5. The algorithm's IP sec users produces a unique identifier for each packet. This identifier then allows a device to determine whether a packet has been correct or not. Packets which are not authorized are discarded and not given to receiver.



WORKING OF IPSEC

1.The host checks if the packet should be transmitted using IPsec or not. These packet traffic triggers the security policy for themselves. This is done when the system sending the packet apply an appropriate encryption. The incoming packets are also checked by the host that they are encrypted properly or not.

2.Then the **IKE Phase 1** starts in which the 2 hosts(using IPsec) authenticate themselves to each other to start a secure channel. It has 2 modes. The **Main mode** which provides the greater security and the **Aggressive mode** which enables the host to establish an IPsec circuit more quickly.

3.The channel created in the last step is then used to securely negotiate the way the IP circuit will encrypt data accross the IP circuit.

4.Now, the **IKE Phase 2** is conducted over the secure channel in which the two hosts negotiate the type of cryptographic algorithms to use on the session and agreeing on secret keying material to be used with those algorithms.

5.Then the data is exchanged across the newly created IPsec encrypted tunnel. These packets are encrypted and decrypted by the hosts using IPsec SAs.

6.When the communication between the hosts is completed or the session times out then the IPsec tunnel is terminated by discarding the keys by both the hosts.

MODES OF OPERATION

The IPsec protocols AH and ESP can be implemented in a host-to-host transport mode, as well as in a network tunneling mode.

Transport mode
In transport mode, only the payload of the IP packet is usually encrypted or authenticated. The routing is intact, since the IP header is neither modified nor encrypted; however, when the authentication header is used, the IP addresses cannot be modified by network address translation, as this always invalidates the hash value. The transport and application layers are always secured by a hash, so they cannot be modified in any way, for example by translating the port numbers.

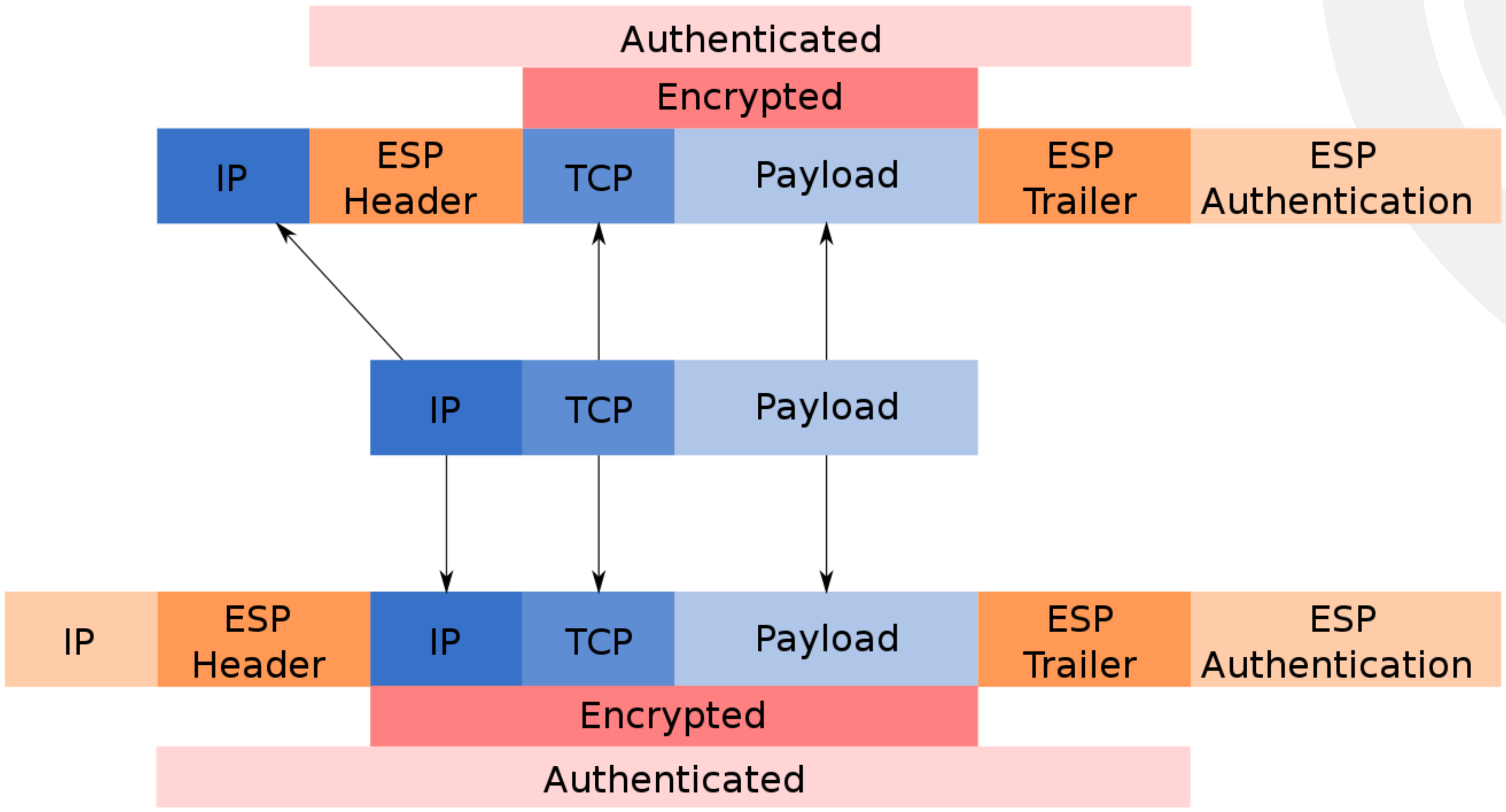
Tunnel mode
In tunnel mode, the entire IP packet is encrypted and authenticated. It is then encapsulated into a new IP packet with a new IP header. Tunnel mode is used to create virtual private networks for network-to-network communications (e.g. between routers to link sites), host-to-network communications (e.g. remote user access) and host-to-host communications (e.g. private chat).[33]

Tunnel mode supports NAT traversal.

In computer networks, a **tunneling protocol** is a communications protocol that allows for the movement of data from one network to another. It involves allowing private network communications to be sent across a public network (such as the Internet) through a process called encapsulation.

Because tunneling involves repackaging the traffic data into a different form, perhaps with encryption as standard, it can hide the nature of the traffic that is run through a tunnel.

The tunneling protocol works by using the data portion of a packet (the payload) to carry the packets that actually provide the service. Tunneling uses a layered protocol model such as those of the OSI or TCP/IP protocol suite, but usually violates the layering when using the payload to carry a service not normally provided by the network. Typically, the delivery protocol operates at an equal or higher level in the layered model than the payload protocol.



Authenticated

Encrypted

IP | ESP Header | TCP | Payload | ESP Trailer | ESP Authentication

ESP Transport Mode

IP | TCP | Payload

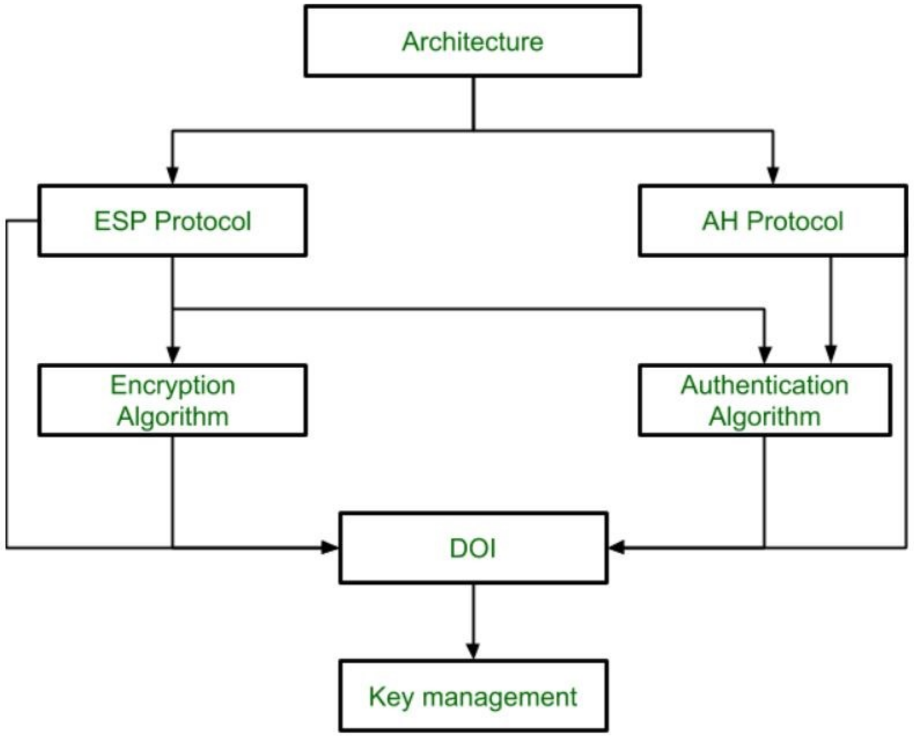
IP | ESP Header | IP | TCP | Payload | ESP Trailer | ESP Authentication

ESP Tunnel Mode

Encrypted

Authenticated

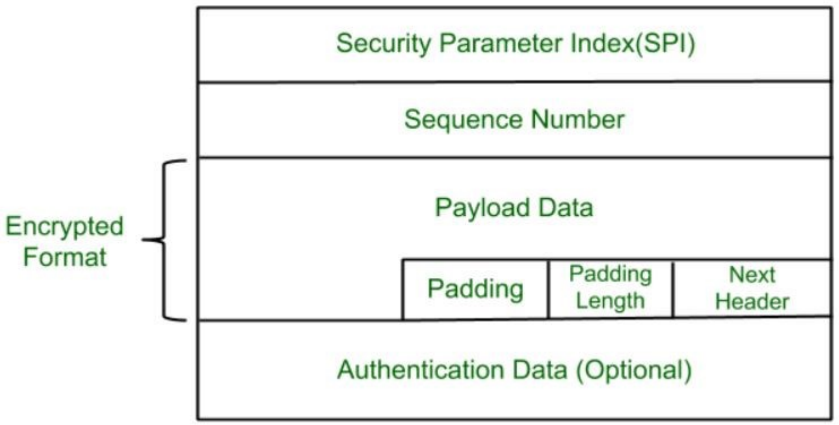
IPSEC ARCHITECTURE



IPSec (IP Security) architecture uses two protocols to secure the traffic or data flow. These protocols are ESP (Encapsulation Security Payload) and AH (Authentication Header). IPSec Architecture include protocols, algorithms, DOI, and Key Management. All these components are very important in order to provide the three main services:

- Confidentiality
- Authentication
- Integrity

Packet Format:



Security Parameter Index(SPI)

Sequence Number

Encrypted Format

Payload Data

Padding

Padding Length

Next Header

Authentication Data (Optional)

•Security Parameter Index(SPI):

This parameter is used in Security Association. It is used to give a unique number to the connection build between Client and Server.

•Sequence Number:

Unique Sequence number are allotted to every packet so that at the receiver side packets can be arranged properly.

•Payload Data:

Payload data means the actual data or the actual message. The Payload data is in encrypted format to achieve confidentiality.

•Padding:

Extra bits or space added to the original message in order to ensure confidentiality. Padding length is the size of the added bits or space in the original message.

•Next Header:

Next header means the next payload or next actual data.

•Authentication Data

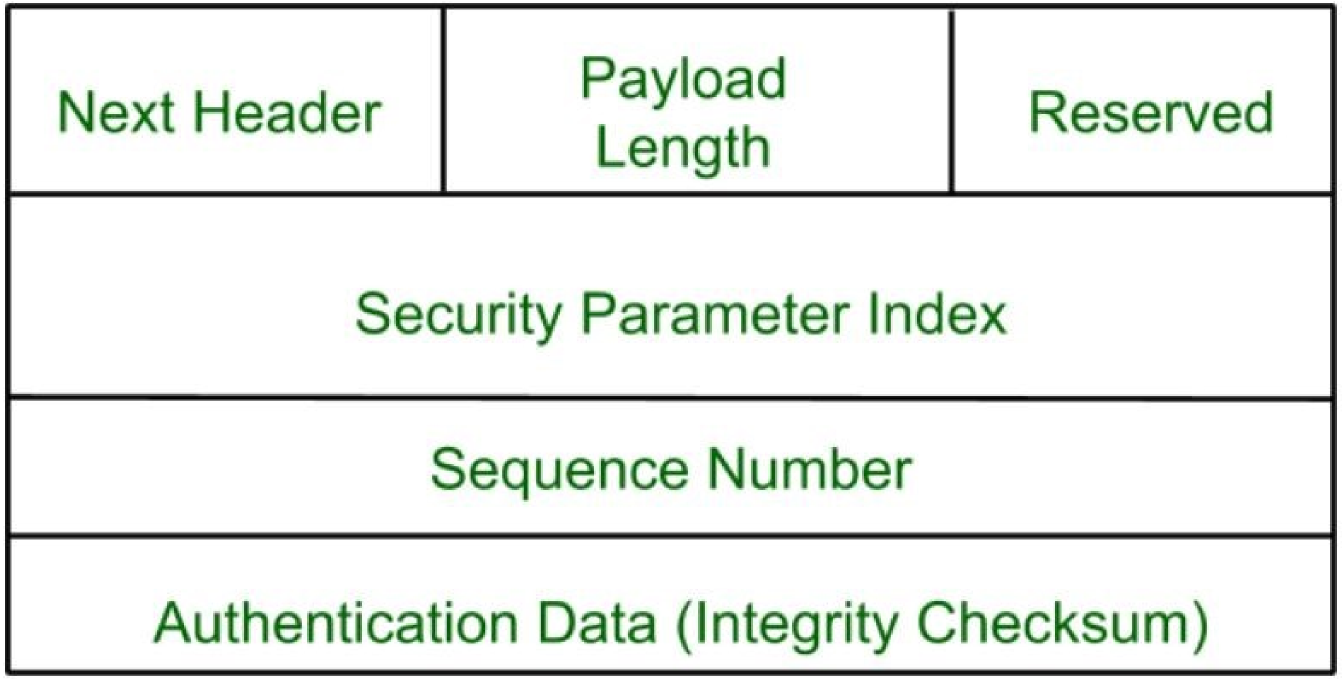
This field is optional in ESP protocol packet format.

3. Encryption algorithm:

Encryption algorithm is the document that describes various encryption algorithm used for Encapsulation Security Payload.

4. AH Protocol:

AH (Authentication Header) Protocol provides both Authentication and Integrity service. Authentication Header is implemented in one way only: Authentication along with Integrity. Authentication Header covers the packet format and general issue related to the use of AH for packet authentication and integrity.



Next Header	Payload Length	Reserved
Security Parameter Index		
Sequence Number		
Authentication Data (Integrity Checksum)		

5. Authentication Algorithm:
Authentication Algorithm contains the set of the documents that describe authentication algorithm used for AH and for the authentication option of ESP.

6. DOI (Domain of Interpretation):
DOI is the identifier which support both AH and ESP protocols. It contains values needed for documentation related to each other.

7. Key Management:
Key Management contains the document that describes how the keys are exchanged between sender and receiver.

DIFFERENCE BETWEEN IPSEC AND SSL

IPSec Protocol:
It is an Internet Engineering Task Force standard suite of protocols between two communication points. It can also be defined as the encrypted, decrypted and authenticated packets. It generally uses cryptographic security services to protect communications. It can be seen that network-level peer and data origin authentication, data integrity, data encryption, and protection are supported by IPsec.
For Example, IPSec can be used in between two routers in order to create a site-to-site VPN and between a firewall and windows host for a remote access VPN.

SSL:
It is a networking protocol that is used at the transport layer to provide a secure connection between the client and the server over the internet. It is a transparent protocol that requires little interaction from the end-user when establishing a secure session. SSL Tunneling involves a client that requires an SSL connection to a backend service or secure server via a proxy server.
For Example, For securing the communication between a web browser and a web server, he SSL is used.

IPSec	SSL
Internet protocol security (IPsec) is a set of protocols that provide security for Internet Protocol.	SSL is a secure protocol developed for sending information securely over the Internet.
It Work in Internet Layer of the OSI model.	It Work in Between the transport layer and application layer of the OSI model.
Configuration of IPsec is Complex	Configuration of SSl is Comparatively Simple
IPsec is used to secure a Virtual Private Network.	SSL is used to secure web transactions.
Installation process is Vendor Non-Specific	Installation process is Vendor Specific
Changes are required to OS for implementation. NO Changes are required to application	No changes are required to OS for implementation but Changes are required to application
IPsec resides in operating system space	SSL resides in user space

PGP AND S/MIME

PGP is an open source software package that is designed for the purpose of email security. It provides the basic or fundamental needs of cryptography.

S/MIME (Secure/Multipurpose Internet Mail Extension) is a security-enhanced version of MIME internet e-mail format, founded on the technology from RSA Data security. In this, public key cryptography is used for digital sign, encrypt or decrypt the email. User acquires a public-private key pair with a trusted authority and then makes appropriate use of those keys with email applications.

Definition of PGP

The PGP (Pretty Good Privacy) is an open source software package designed for email security. It was developed by Phil Zimmerman. The most important feature of the PGP is that it provides the basic requirements of the cryptography.

It uses various steps such as authentication, confidentiality, compression, e-mail compatibility, segmentation and reassembly for securing the email. To implement public key trust model and public key certificate management tools are included in the PGP to develop it.

Working of PGP

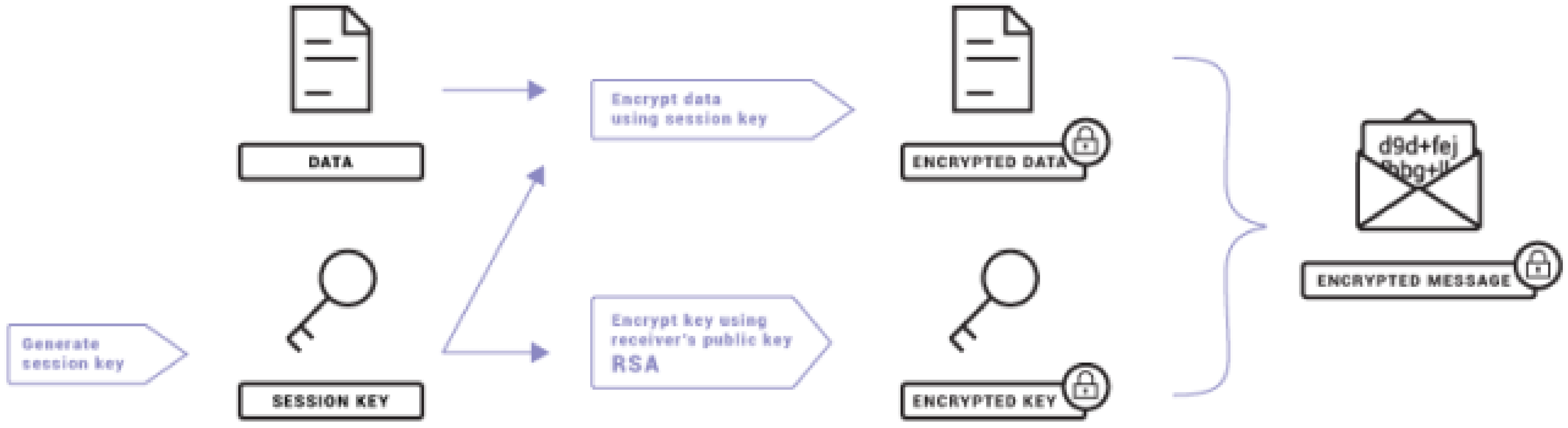
In the beginning, identifiers of the algorithm used in the message are required to be incorporated along with the values of the keys. The security options permitted by PGO while sending an email message are described below:

- Signature only
- Signature and Bade-64 encoding
- A signature, Encryption, Enveloping and Base-64 encoding.

Process of PGP step by step.

•**Digital Signature**: In this step, the message digest of the email message is produced using the SHA-1 algorithm and then the sender's private key is employed for encrypting the message digest.
•**Compression**: This step compresses the email message and the digital signature together in order to decrease the size of the resulting transit message. One of the examples of compression is the ZIP program based on the Lempel-Ziv algorithm.
•**Encryption**: It encrypts the compressed form of the message which we got in the previous step using a symmetric key.
•**Digital Enveloping**: In this step, the receiver's public key is used for encrypting the symmetric key used in the encryption step. The combination of step 3 and step 4 creates a digital envelope.
•**Base-64 encoding**: Here the output of step 4 is Base-64 encoded where a bunch of arbitrary binary inputs are translated into printable characters.

ENCRYPT



DATA

Encrypt data using session key

ENCRYPTED DATA

Generate session key

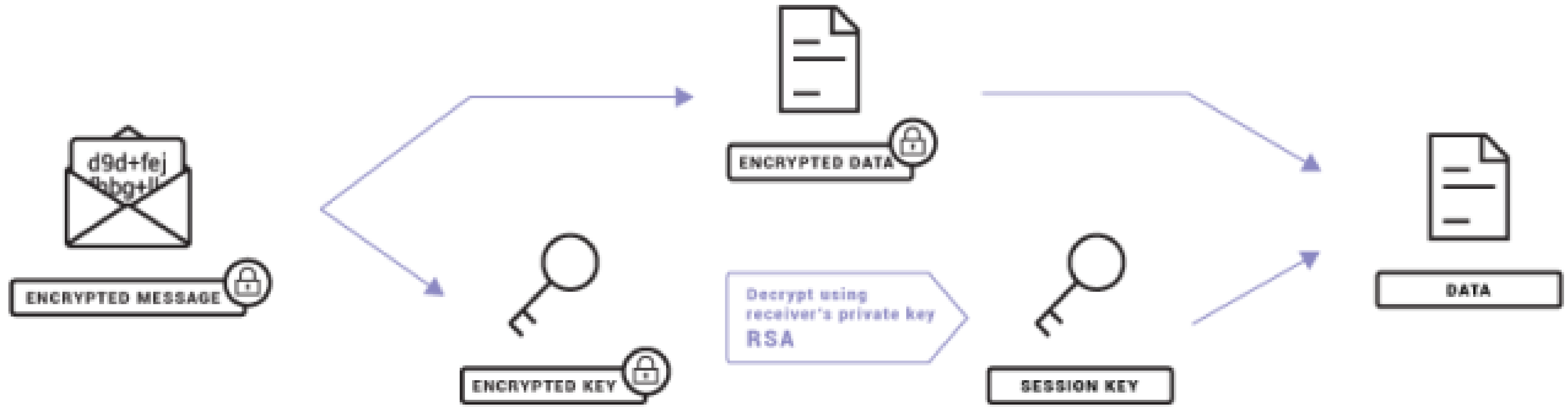
SESSION KEY

Encrypt key using receiver's public key RSA

ENCRYPTED KEY

ENCRYPTED MESSAGE

DECRYPT



ENCRYPTED MESSAGE

ENCRYPTED DATA

ENCRYPTED KEY

Decrypt using receiver's private key RSA

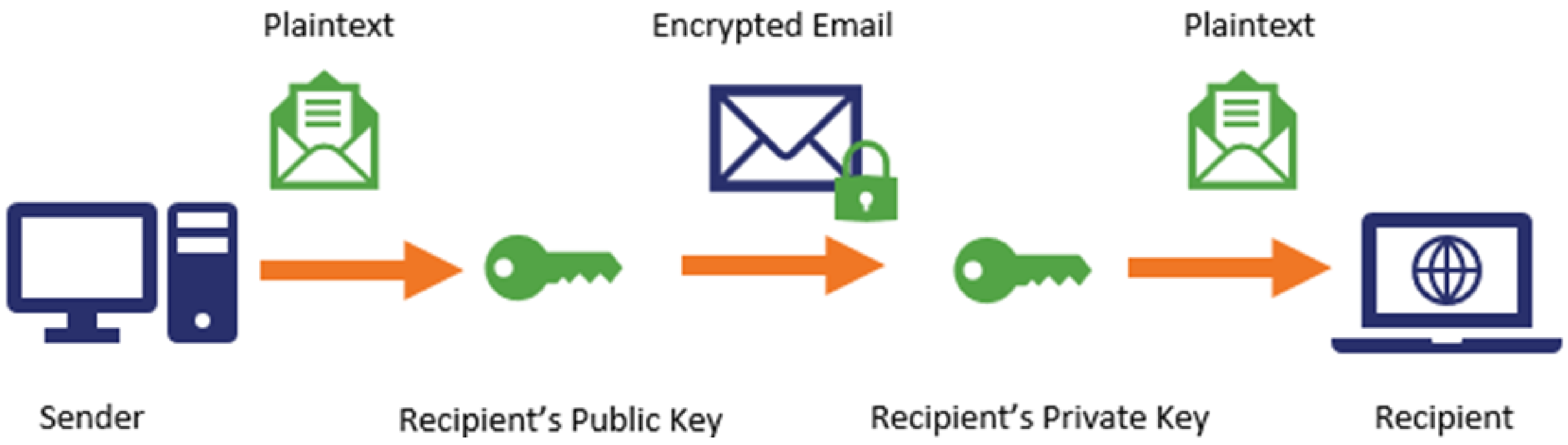
SESSION KEY

DATA

Definition of S/MIME

S/MIME (Secure/Multipurpose Internet Mail Extension) is a security-enhanced version of MIME internet e-mail format, founded on the technology from RSA Data security. MIME replaced the SMTP protocol as it was limited to just text where only text messages were allowed to transmit. The restricted protocol SMTP was not able to exchange the multimedia files, documents in various arbitrary formats because it uses 7-bit ASCII representation of characters of an email message which cannot represent the special characters having the value greater than 127.

Therefore, there was a critical need for a standard which can also exchange multimedia files. That is how the features of the primary email system are extended through MIME.



Our Email Server Is Already Encrypted. Isn't That Enough?

Sure, encrypting your email servers with Digital Certificates is a wise move. This also prevents outsiders from getting in between your email and mail servers and intercepting sensitive data. However, it can only do as much, as the Digital Certificates that encrypt the server don’t necessarily protect the emails themselves. Basically, your emails will be protected to and from the encrypted server but hackers can still get in your email system and open your messages from there or access them while they pass through other servers. So sure, your emails are well-protected on transit to your server but the emails at rest or in transit elsewhere are still up for grabs.

This was evident in a 2016 attack that stole almost 20,000 emails from the Democratic National Committee right in the middle of the US elections. The hacker forced his way into the DNC’s unencrypted inbox. The emails that revealed the DNC’s supposed bias towards Sen. Bernie Sanders were published in WikiLeaks, with some experts saying that this hack started Hillary’s downfall towards losing the elections. Encrypting the individual emails themselves, using S/MIME for example, would have kept the contents inaccessible.

S/MIME is based on asymmetric cryptography that uses a pair of mathematically related keys to operate – a public key and a private key. It is computationally infeasible to figure out the private key based on the public key. Emails are encrypted with the recipient’s public key. The email can only be decrypted with the corresponding private key, which is supposed to be in sole possession of the recipient.

MIME structure

A MIME email message is comprised of a text message along with some special headers and formatted sections of text. Each section could contain an ASCII-encoded part of data and also the mechanism of decoding the data at the receiver’s end. MIME headers include specification of – MIME version, Content-Type, Content-Transfer-Encoding, Content-ID, Content-Description.

S.NO	PGP	S/MIME
1.	It is designed for processing the plain texts	While it is designed to process email as well as many multimedia files.
2.	PGP is less costly as compared to S/MIME.	While S/MIME is comparatively expensive.
3.	PGP is good for personal as well as office use.	While it is good for industrial use.
4.	PGP is less efficient than S/MIME.	While it is more efficient than PGP.
5.	It depends on user key exchange.	Whereas it relies on a hierarchically valid certificate for key exchange.
6.	PGP is comparatively less convenient.	While it is more convenient than PGP due to the secure transformation of all the applications.
7.	PGP contains 4096 public keys.	While it contains only 1024 public keys.
8.	PGP is the standard for strong encryption.	While it is also the standard for strong encryption but has some drawbacks.
9.	PGP is also be used in VPNs.	While it is not used in VPNs, it is only used in email services.
10.	PGP uses **Diffie hellman digital signature**.	While it uses **Elgamal digital signature**.

SSH –SECURE SHELL PROTOCOL

Early network protocols like Telnet did not provide enough protection against malicious cyber attacks. The need for a more secure network communication method inspired the creation of the SSH protocol.

The original SSH is now considered outdated and less secure. SSH2 is the current version.

What is SSH?

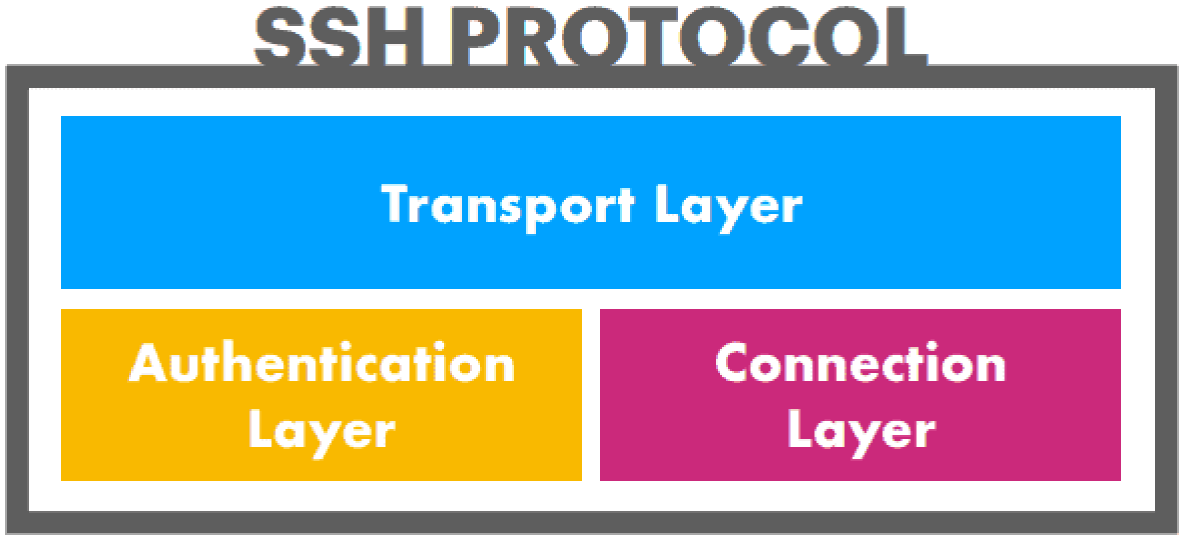
SSH (short for Secure Shell) is a network protocol that provides a secure way for two computers to connect remotely. SSH employs encryption to ensure that hackers cannot interpret the traffic between two connected devices.

SSH consists of three distinct layers:

- **The transport layer** establishes safe and secure communication between a client and a server during and after authentication. It oversees data encryption, decryption, and integrity protection. Furthermore, it helps speed up data exchange by providing data compression and caching.

- **The authentication layer** communicates the supported authentication methods to the client. It also conducts the entire user authentication process.

- **The connection layer** manages the communication between the machines after the authentication succeeds. It handles the opening and closing of communication channels and allows multiple channels for multiple sessions.



What is SSH Used for?

SSH provides a layer of security for information transfer between machines. Some important use cases for SSH are:

Remote access – SSH ensures encrypted remote connections for users and processes.

File transfers – SFTP, a secure file transfer protocol managed by SSH, provides a safe way to manipulate files over a network.

X11 Forwarding – Users can run server-hosted X applications from their client machines.

Port Forwarding – By mapping a client's port to the server's remote ports, SSH helps secure other network protocols, such as TCP/IP.

Tunneling – This encapsulation technique provides secure data transfers. Tunneling is useful for accessing business-sensitive online materials from unsecured networks, as it can act as a handy VPN alternative.

Network management – The SSH protocol manages network infrastructure and other parts of the system.

How Does SSH Work?

SSH is a client-server based protocol. This means the protocol allows a device requesting information or services (the client) to connect to another device (the server).

When a client connects to a server over SSH, the machine can be controlled like a local computer.

The server has a designated TCP port over which it monitors the network, waiting for clients to initialize the connection. Before a client connects and starts issuing SSH commands, it needs to pass the authentication process.

Run the following command on a client machine to initiate an SSH connection:

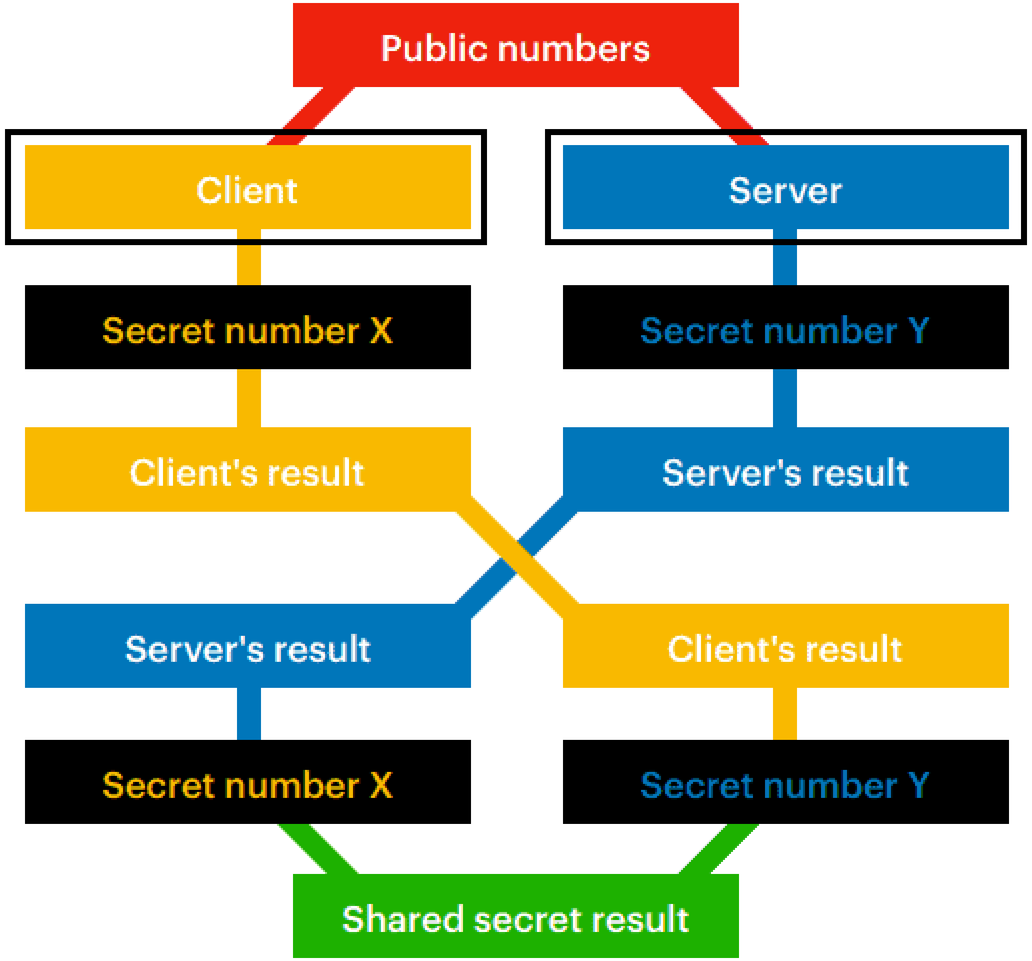
```
ssh [username]@[server_ip_or_hostname]
```

When the server receives the requests, a session encryption negotiation begins.

Session Encryption Negotiation

Upon receiving a connection request, the server sends the client a set of supported encryption protocols. The server uses the public key as the authentication method. The client compares the protocols to its own set. If there are matching protocols, the machines agree to use one to establish the connection.

The client compares the server's public key to the stored private key stored in its system on the first connection attempt. If the keys match, the client and the server agree to use symmetric encryption to communicate during the SSH session. For this purpose, they communicate using an asymmetrically encrypted process that employs the Diffie-Hellman (DH) key exchange algorithm.



The Diffie-Hellman (DH) algorithm enables machines to work together and securely create a cryptographic key over a public network. To generate a key, the machines perform the following steps:

•**The machines agree on two numbers**: a modulus and a base number. To prevent brute force key decryption, the chosen modulus is a prime number of at least 600 digits.

•**The machines separately choose one number** and apply it to the equation involving the two public numbers.

•The server and the client **exchange the calculated values**.

•Each machine now **performs a calculation** using the result received from the other.

By performing the steps above, both machines calculate the same value, their secret key. Finally, the server then attempts to authenticate the user who requests access.

User Authentication

The two most common SSH user authentication methods used are passwords and SSH keys. The clients safely send encrypted passwords to the server. However, passwords are a risky authentication method because their strength depends on the user’s awareness of what makes a strong password.

Asymmetrically encrypted SSH public-private key pairs are a better option. Once the client decrypts the message, the server grants the client access to the system.

The system generates and stores the keys.

SSH Encryption Technologies

SSH uses three data encryption types during the communication between the machines. These are:
•Symmetric encryption
•Asymmetric encryption
•Hashing.

Symmetric Encryption

Symmetric encryption generates a single key that two machines exchange. Then, the machines use the key for both encryption and decryption. This method is quick, it is not resource-intensive, and SSH uses it for each session.
Whenever the client and the server negotiate which algorithm to use for an SSH session, they always choose the first algorithm on the client's list that the server supports.

Asymmetric Encryption

Data is asymmetrically encrypted when machines use two different but mathematically related keys, public and private, to perform the encryption. The client machine that participated in setting up the encryption can decrypt the information using the private key.
SSH uses temporal asymmetric keys to exchange symmetric keys, such as during the user authentication process.

Hashing

SSH uses hashing to validate if the data packets come from the source they appear to come from. Hashing algorithms used to produce hashes in SSH are Message Authentication Code (MAC) and Hashed Message Authentication Code (HMAC).
A hashing algorithm uses a data packet to create a unique hash string. The machine sending the packet always sends the packet together with the hash value.
The receiving machine knows the algorithm used to create the hash and can apply it to the data. The purpose is to see if the calculated hash value will be the same. If the obtained hash value differs from the sender's hash, the data got corrupted during the transfer.

KERBEROS

Kerberos is a network authentication protocol. It is designed to provide strong authentication for client/server applications by using secret-key cryptography. A free implementation of this protocol is available from the Massachusetts Institute of Technology. Kerberos is available in many commercial products as well. Initially developed by the Massachusetts Institute of Technology (MIT) for Project Athena in the late '80s, Kerberos is now the default authorization technology used by Microsoft Windows.

The protocol derives its name from the legendary three-headed dog Kerberos (also known as Cerberus) from Greek myths, the canine guardian to the entrance to the underworld. Kerberos had a snake tail and a particularly bad temper and, despite one notable exception, was a very useful guardian.
But in the protocol's case, the three heads of Kerberos represent the client, the server, and the Key Distribution Center (KDC). The latter functions as the trusted third-party authentication service.

The Internet is an insecure place. Many of the protocols used in the Internet do not provide any security. Tools to "sniff" passwords off of the network are in common use by malicious hackers. Thus, applications which send an unencrypted password over the network are extremely vulnerable. Worse yet, other client/server applications rely on the client program to be "honest" about the identity of the user who is using it. Other applications rely on the client to restrict its activities to those which it is allowed to do, with no other enforcement by the server.
Some sites attempt to use firewalls to solve their network security problems. Unfortunately, firewalls assume that "the bad guys" are on the outside, which is often a very bad assumption. Most of the really damaging incidents of computer crime are carried out by insiders. Firewalls also have a significant disadvantage in that they restrict how your users can use the Internet. In many places, these restrictions are simply unrealistic and unacceptable.

Kerberos was created by MIT as a **solution to these network security problems**. The Kerberos protocol uses **strong cryptography** so that a client can prove its identity to a server (and vice versa) across an insecure network connection. After a client and server has used Kerberos to prove their identity, they can also encrypt all of their communications to assure privacy and data integrity as they go about their business.

Users, machines, and services that use Kerberos depend on the KDC alone, which works as a single process that provides two functions: authentication and ticket-granting. KDC "tickets" offer authentication to all parties, allowing nodes to verify their identity securely. The Kerberos authentication process employs a conventional shared secret cryptography that prevents packets traveling across the network from being read or altered, as well as protecting messages from eavesdropping and replay (or playback) attacks.

KERBEROS PROTOCOL FLOW

Here are the principal entities involved in the typical Kerberos workflow:

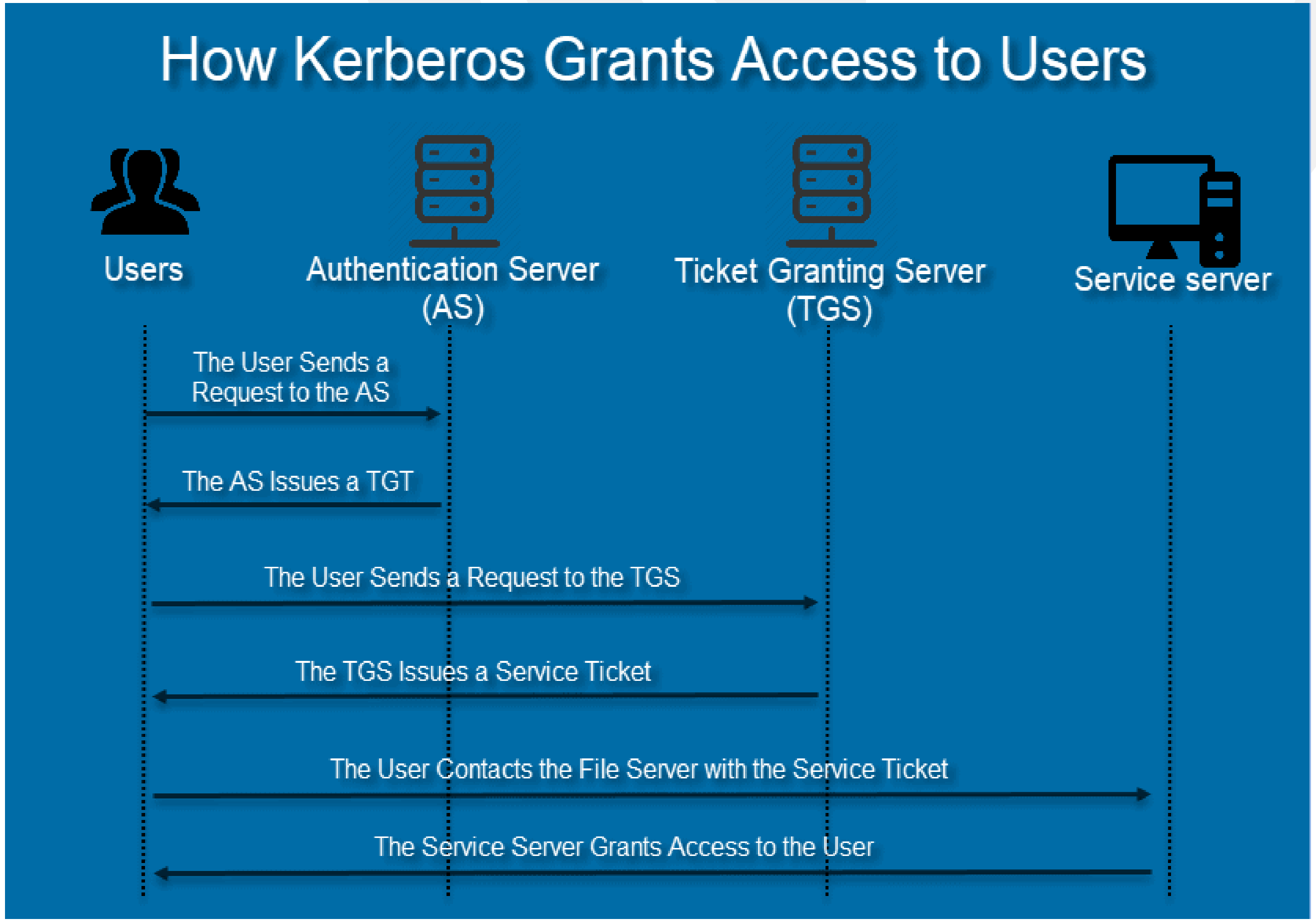
•**Client.** The client acts on behalf of the user and initiates communication for a service request

•**Server.** The server hosts the service the user wants to access

•**Authentication Server (AS).** The AS performs the desired client authentication. If the authentication happens successfully, the AS issues the client a ticket called **TGT (Ticket Granting Ticket)**. This ticket assures the other servers that the client is authenticated

•**Key Distribution Center (KDC)**. In a Kerberos environment, the authentication server logically separated into three parts: A database (db), the Authentication Server (AS), and the **Ticket Granting Server (TGS)**. These three parts, in turn, exist in a single server called the Key Distribution Center

•**Ticket Granting Server (TGS).** The TGS is an application server that issues service tickets as a service


How Kerberos Grants Access to Users
Users
Authentication Server (AS)
Ticket Granting Server (TGS)
Service server
The User Sends a Request to the AS
The AS Issues a TGT
The User Sends a Request to the TGS
The TGS Issues a Service Ticket
The User Contacts the File Server with the Service Ticket
The Service Server Grants Access to the User

PROTOCOL FLOW

First, there are three crucial secret keys involved in the Kerberos flow. There are unique secret keys for the client/user, the TGS, and the server shared with the AS.

- Client/user. Hash derived from the user's password
- TGS secret key. Hash of the password employed in determining the TGS
- Server secret key. Hash of the password used to determine the server providing the service.

Step 1: Initial client authentication request. The user asks for a Ticket Granting Ticket (TGT) from the authentication server (AS). This request includes the client ID.

Step 2: KDC verifies the client's credentials. The AS checks the database for the client and TGS's availability. If the AS finds both values, it generates a client/user secret key, employing the user's password hash.

The AS then computes the TGS secret key and creates a session key (SK1) encrypted by the client/user secret key. The AS then generates a TGT containing the client ID, client network address, timestamp, lifetime, and SK1. The TGS secret key then encrypts the ticket.

Step 3: The client decrypts the message. The client uses the client/user secret key to decrypt the message and extract the SK1 and TGT, generating the authenticator that validates the client's TGS.

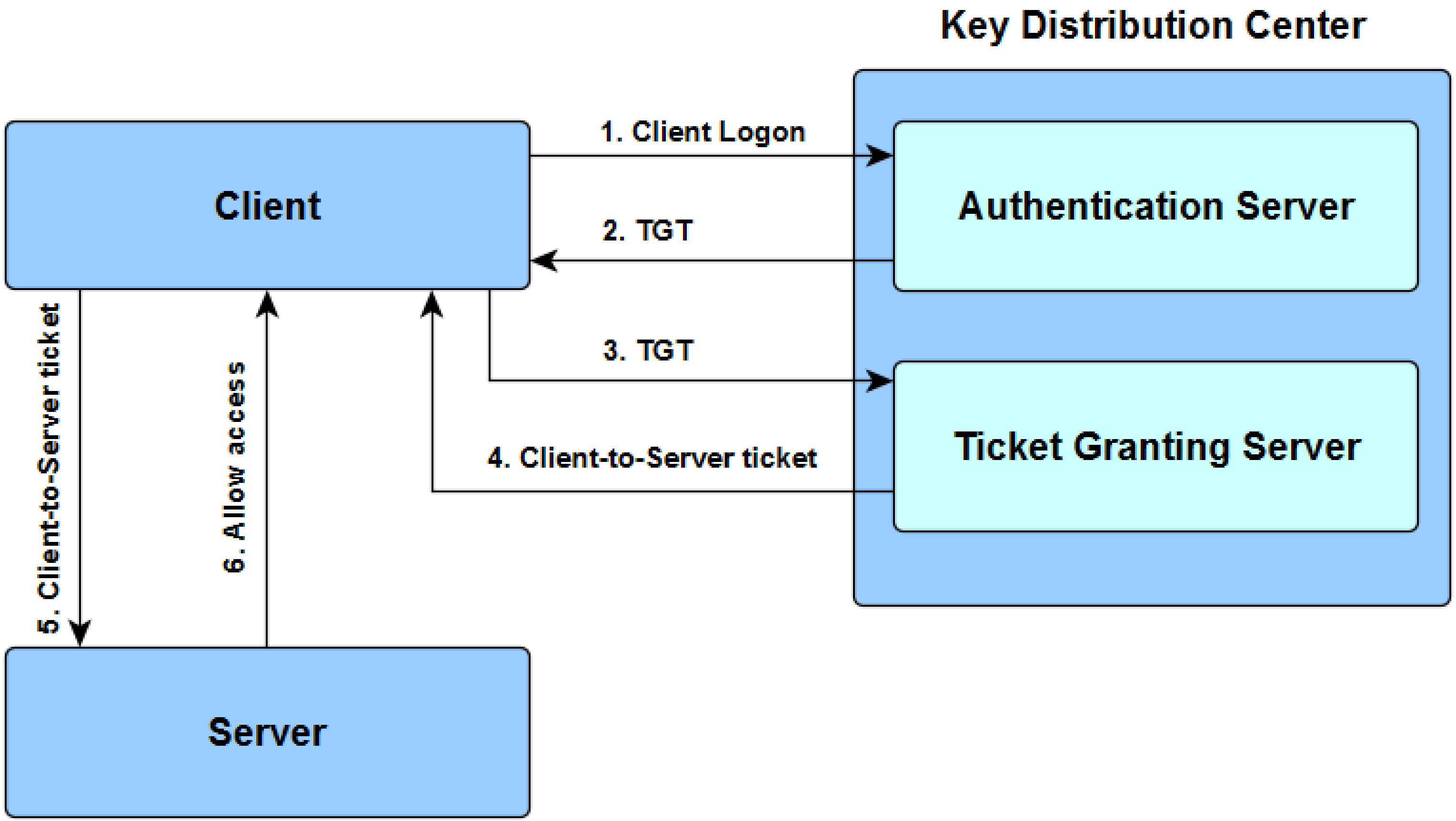
Step 4: The client uses TGT to request access. The client requests a ticket from the server offering the service by sending the extracted TGT and the created authenticator to TGS.

Step 5: The KDC creates a ticket for the file server. The TGS then uses the TGS secret key to decrypt the TGT received from the client and extracts the SK1. The TGS decrypts the authenticator and checks to see if it matches the client ID and client network address. The TGS also uses the extracted timestamp to make sure the TGT hasn't expired.
If the process conducts all the checks successfully, then the KDC generates a service session key (SK2) that is shared between the client and the target server.
Finally, the KDC creates a service ticket that includes the client id, client network address, timestamp, and SK2. This ticket is then encrypted with the server's secret key obtained from the db. The client receives a message containing the service ticket and the SK2, all encrypted with SK1.

Step 6: The client uses the file ticket to authenticate. The client decrypts the message using SK1 and extracts SK2. This process generates a new authenticator containing the client network address, client ID, and timestamp, encrypted with SK2, and sends it and the service ticket to the target server.

Step 7: The target server receives decryption and authentication. The target server uses the server's secret key to decrypt the service ticket and extract the SK2. The server uses SK2 to decrypt the authenticator, performing checks to make sure the client ID and client network address from the authenticator and the service ticket match. The server also checks the service ticket to see if it's expired.
Once the checks are met, the target server sends the client a message verifying that the client and the server have authenticated each other. The user can now engage in a secure session.

KERBEROS



1.Client sends a request to Authentication Server (AS) with plaintext user ID and asking for a server access on behalf of the user. This request is partially encrypted with a secret key which is the password of the client user.

2.Authentication Server (AS) retrive the secret key (user’s password) from the user DB based on the user ID and use his password as a key to decrypt the request. That is how the user is verified. Then the AS sends a Ticket-Granting Ticket (TGT) encrypted with another secret key which is shared between AS and Ticket-Granting Server (TGS).

3.Client send the encrypted TGT to the Ticket-Granting Server (TGS) requesting the access for the server.

4.Ticket-Granting Server (TGS) decrypt the TGT with shared secret key with AS and issue a kerberos token to the client which is encrypted with another shared secret key with TGS and server.

5.Client sends a request to server with the encrypted kerberos token.

6.Then the server allow the access to the requested resources to the client for a certain period of time specified in the token.

KERBEROS SECURITY

Since it's been around for so long, hackers have had the opportunity over the years to find ways around it, usually by forging tickets, making repeated attempts to guess passwords (brute force/credential stuffing), and using malware to downgrade the encryption.

Despite this, Keberos is still the best security access protocol available today. The protocol is flexible enough to employ more robust encryption algorithms to help combat new threats, and if users practice good password choice policies, they are better protected.