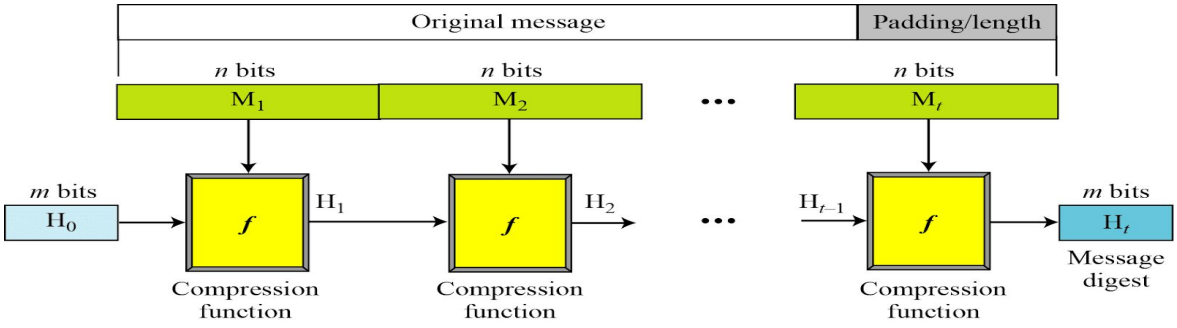
UNIT-IV

Iterated Hash Functions

Merkle-Damgard Scheme:

It is an iterated hash function that is collision resistant if the compression function is collision resistant.



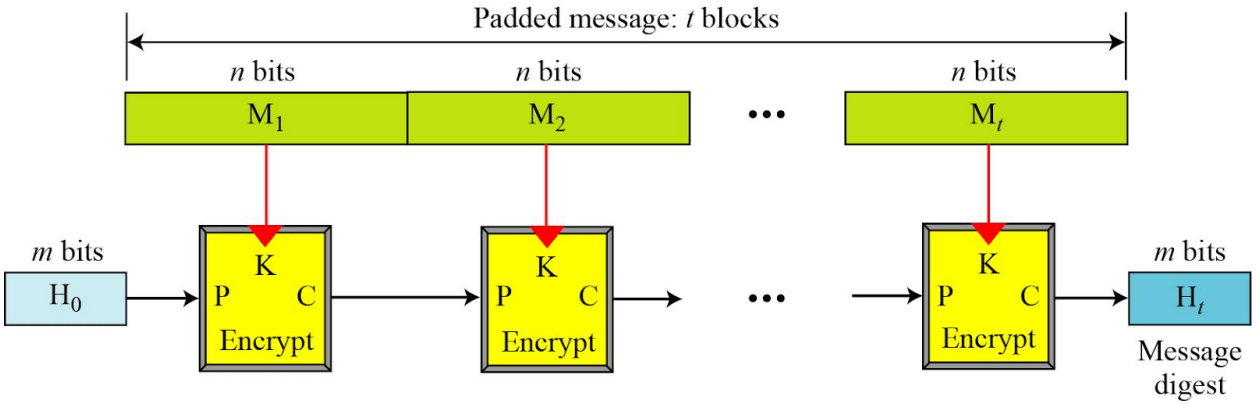
The scheme uses the following steps:

1. The message length and padding are appended to the message to create an augumented message which can be evenly divided into blocks of n bits, where n is the size of the block to be processed by te compression function.
2. The message is then considered as t blocks, each of n bits. We call each block M1,M2,....Mt. the digest created at t iterations is H1,H2....Ht.
3. Before starting the iteration, the digest H0 is set to a fixed value, normally called IV
4. The compression function at each iteration operates on H_{i-1} and Mi
5. Hi is the cryptographic hash function of the original message h(M).

Hash Functions based on Block Ciphers

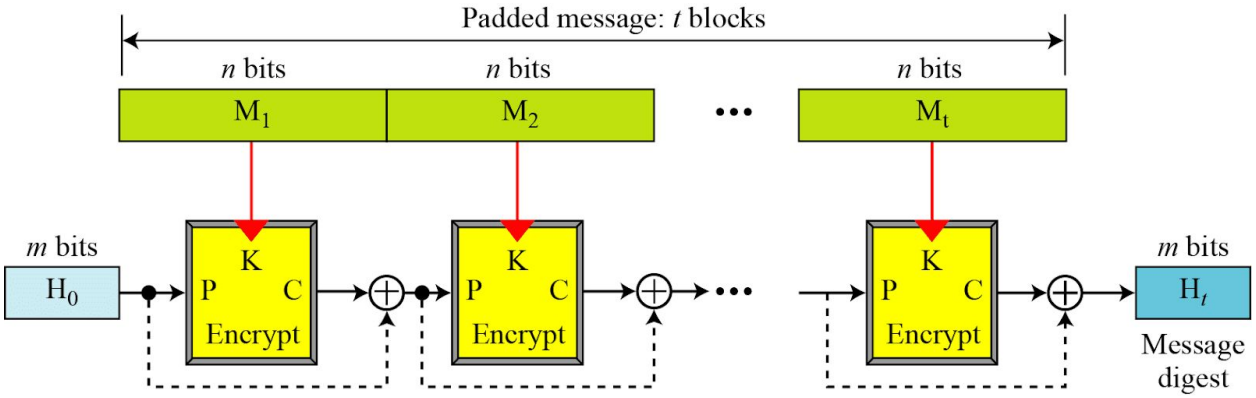
Rabin Scheme

The iterated hash function proposed b rabin is very simple. This scheme is based on Merkle-Damgard Scheme. The compression function is replaved by any encrypting cipher. The message block is used as the key, the previously created digest is used as the plaintext. The cipher text is the new message digest.



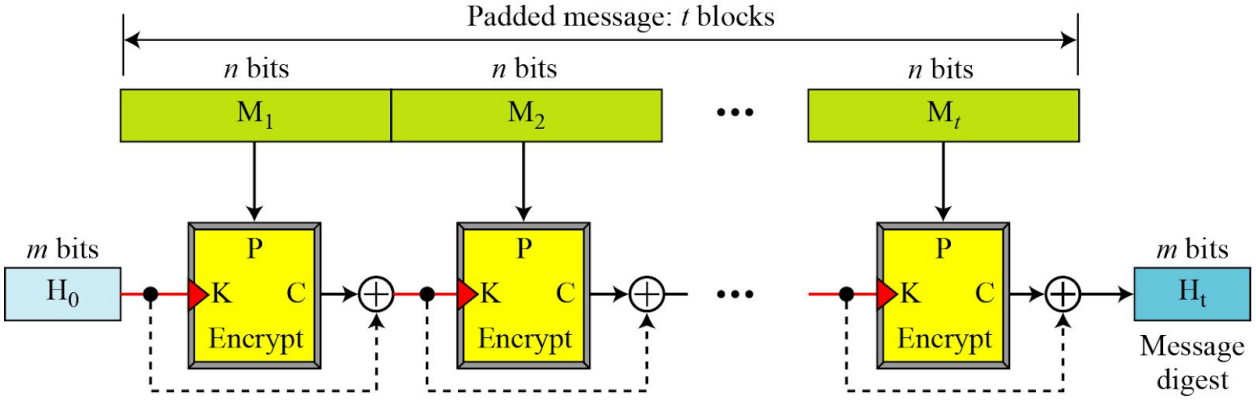
Davies- Meyer Scheme

It is basically same as the Rabin Scheme except that is uses forward feed to protect against meet-in-the middle attack.



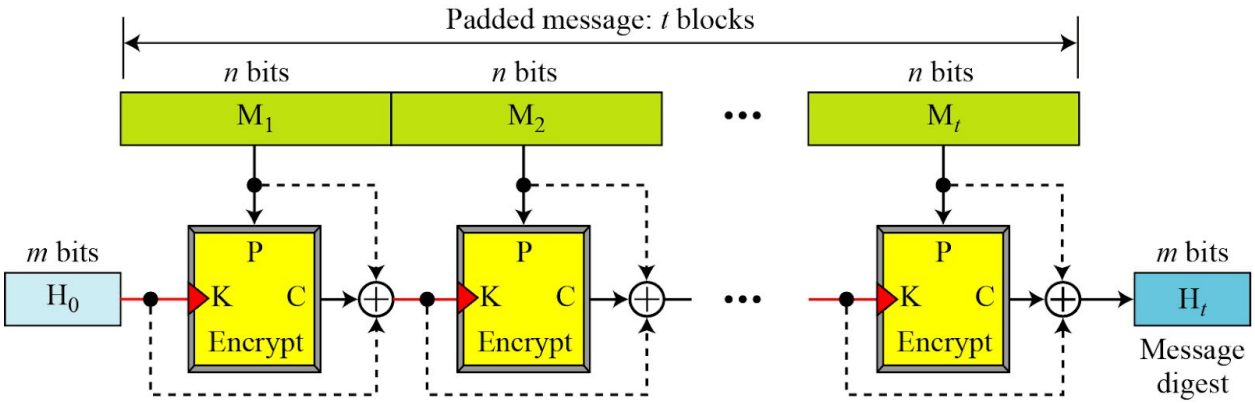
Matyas-Meyer-Oseas Scheme

It is a dual verision of the Davies-Meyer Scheme. The message block is used as the key to the cryptosystem. The scheme can be used if the data block and the cipher key are the same size. AES is the good candidate for this scheme



Miyaguchi-Praneel Scheme:

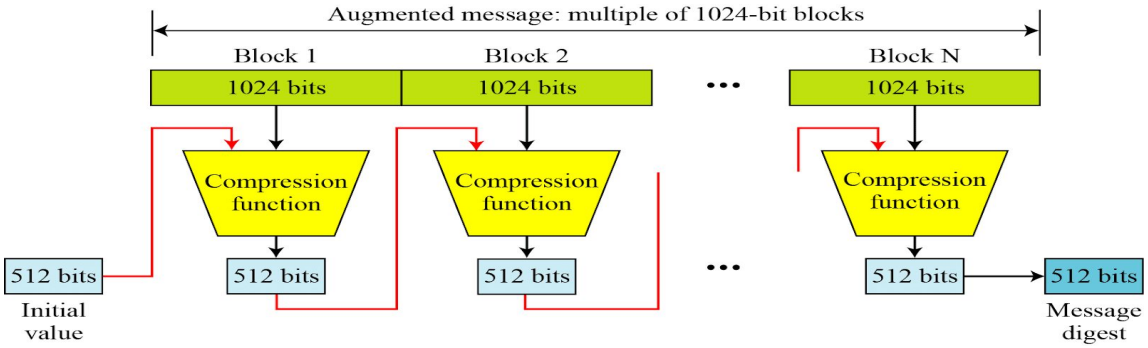
It is an extended version of Matyas-Meyer-Oseas. To make the algorithm stronger against attack, the plaintext, the cipher key and the cipher text are all exclusive ored together to create the new digest. This is the scheme used by the Whirlpool hash function.



SHA-512

SHA-512 is the version of SHA with a 512 bit message digest. This version is based on Merkle-Damgard Scheme.

SHA-512 creates a digest of 512 bits from a multiple-block message. Each block is 1024 bits in length.



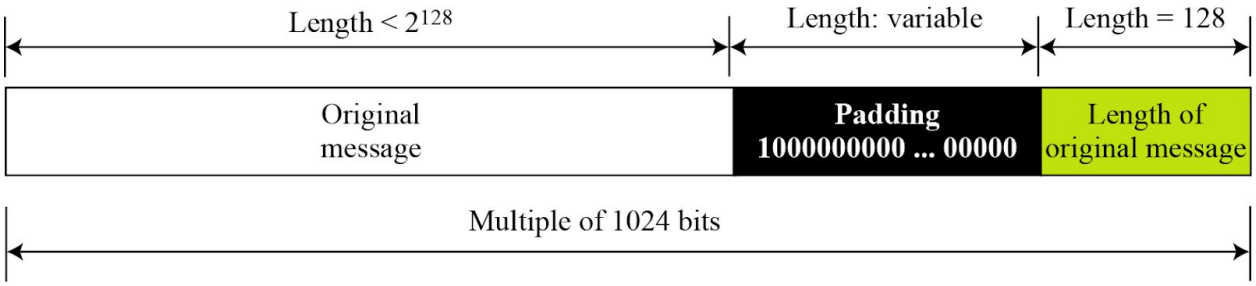
The digest is initialized to a predetermined value of 512 bits. The algorithm mixes this initial value with the first block of the message to create the first intermediate message digest of 512 bits. The digest is then mixed with the second block to create the second intermediate digest. Finally the N-1 th digest is mixed with the Nth block to create the Nth digetst. When the last block is processed, the resulting digest is the message digest for the entire message.

Message Preparation:

SHA-512 insists that the length of the original message be less than 2^{128} bits.

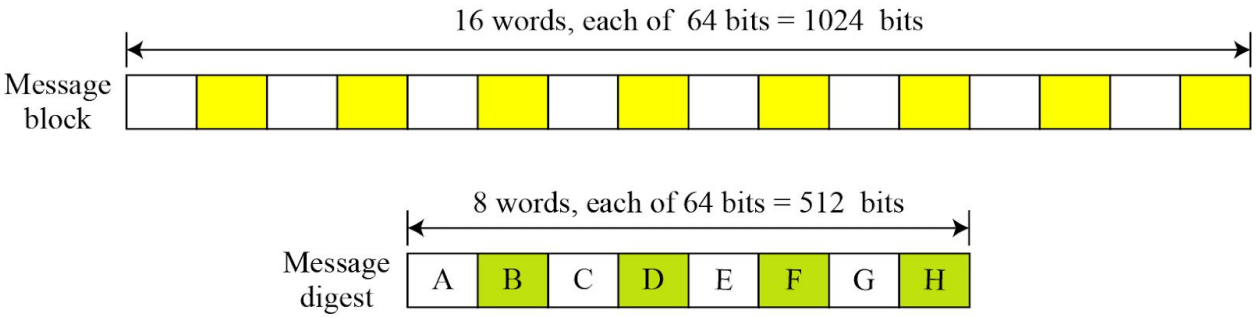
Length Field and Padding

Before the message digest can be created, SHA-512 requires the addition of a 128 bit unsigned integer length field to the message that defines the length of the message in bits.



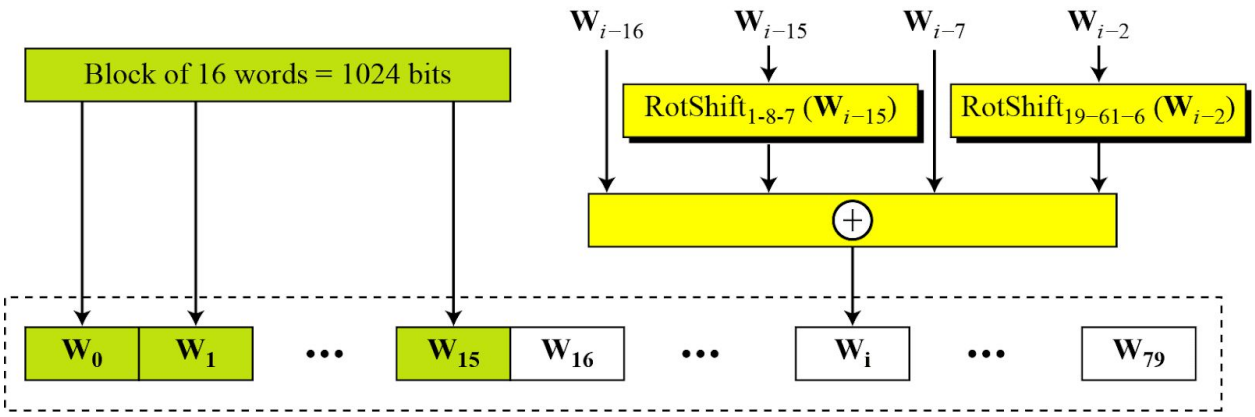
Words

SHA-512 operates on words, it is word oriented. A word is defined as 64 bits. After padding and the length field are added to themessage, each block of the message consists of sixteen 64-bit words. The message digest is also made of 64-bit words, but the message digest is only eight words are the words are named A,B.... H



Word Expansion

Before processing, each message block must be expanded. A block is made of 1024 bits, or sixteen 64 bit words. We need 80 words in the procession phase. So the 16 word block needs to be expanded to 80 words, from W0 to W79.

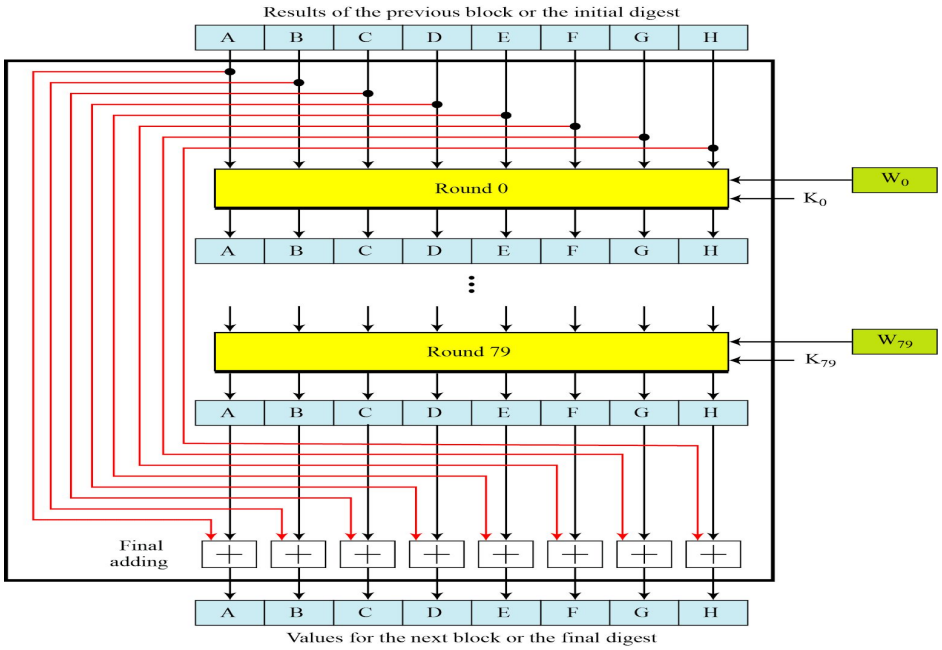


RotShift$_{l\text{-}m\text{-}n}$ (x): RotR$_l(x)$ $\oplus$ RotR$_m$ (x) $\oplus$ ShL$_n$ (x)

RotR$_i(x)$: Right-rotation of the argument x by i bits
ShL$_i$(x): Shift-left of the argument x by i bits and padding the left by 0's.

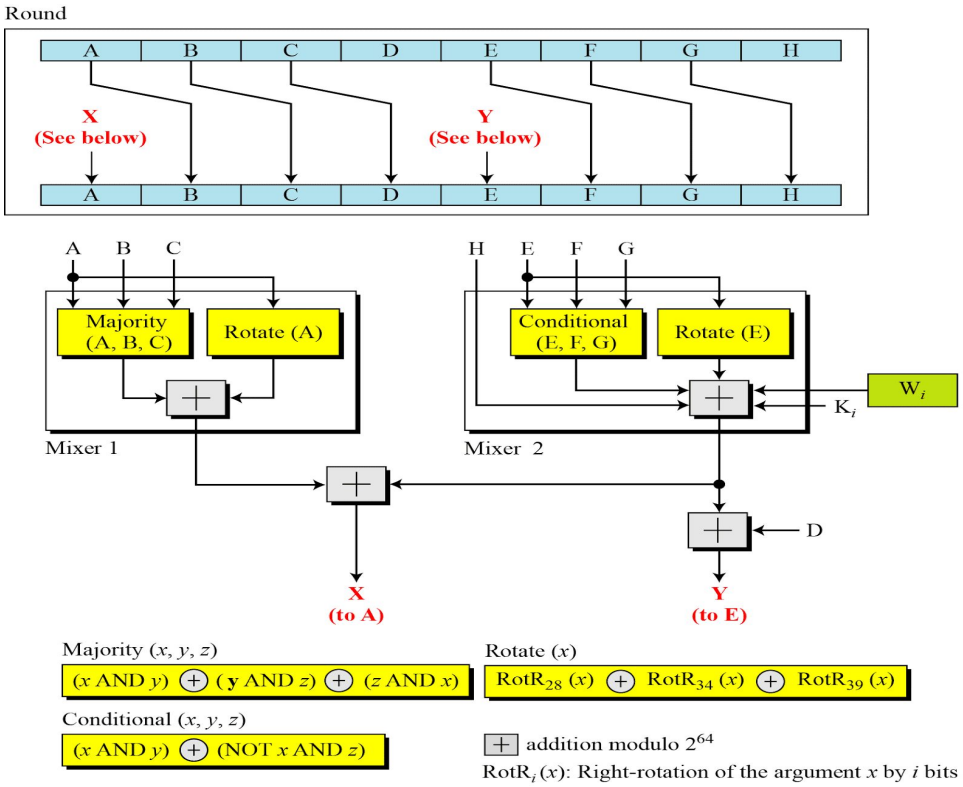
Compression function

SHA-512 creates a 512 bit message digest from a multiple block message when each block is 1024 bits. The processing of each block of data in SHA-512 involves 80 rounds. In each round, the contents of eight previous buffers, one word from the expanded block and one 64 bit constant are mixed together and then operated on to create a new set of eight buffers.



Structure of Each Round:

In each round, eight new values for the 64 bit buffers are creatd from the values of the buffers in the previous round.



There are two mixers, three functions, and several operators. Each mixer combines two functions. The description of the functions and operators follows:

$$(A_j \text{ AND } B_j) \oplus (B_j \text{ AND } C_j) \oplus (C_j \text{ AND } A_j)$$

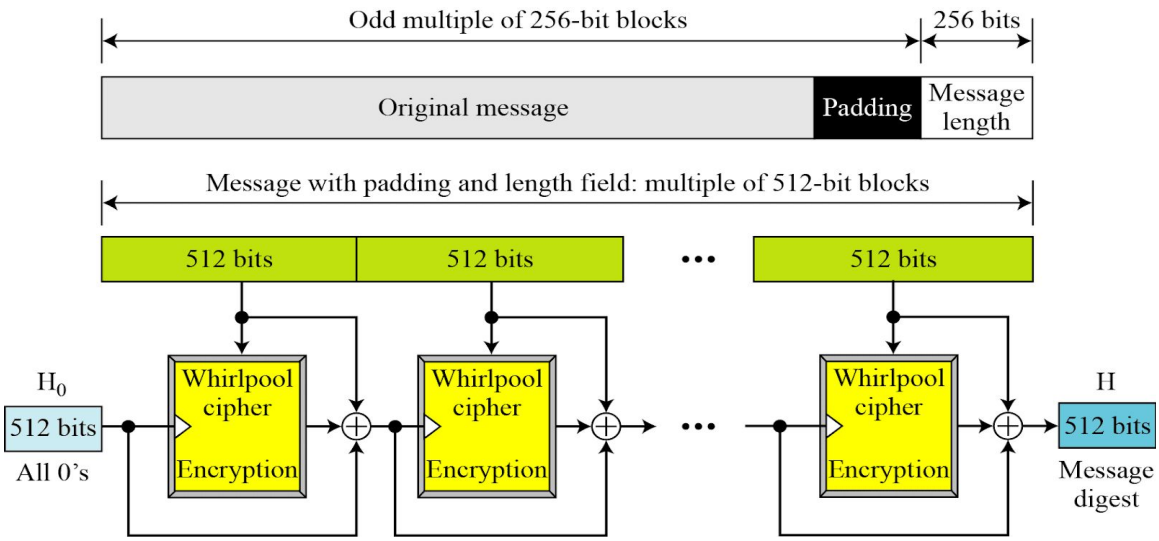
$$(E_j \text{ AND } F_j) \oplus (\text{NOT } E_j \text{ AND } G_j)$$

$$\text{Rotate (A): } RotR_{28}(A) \oplus RotR_{34}(A) \oplus RotR_{29}(A)$$
$$\text{Rotate (E): } RotR_{28}(E) \oplus RotR_{34}(E) \oplus RotR_{29}(E)$$

Whirlpool

Whirlpool is an iterated cryptographic hash function based on the Miyaguchi-Praneel Scheme, that uses a symmetric key block cipher in place of the compression function.

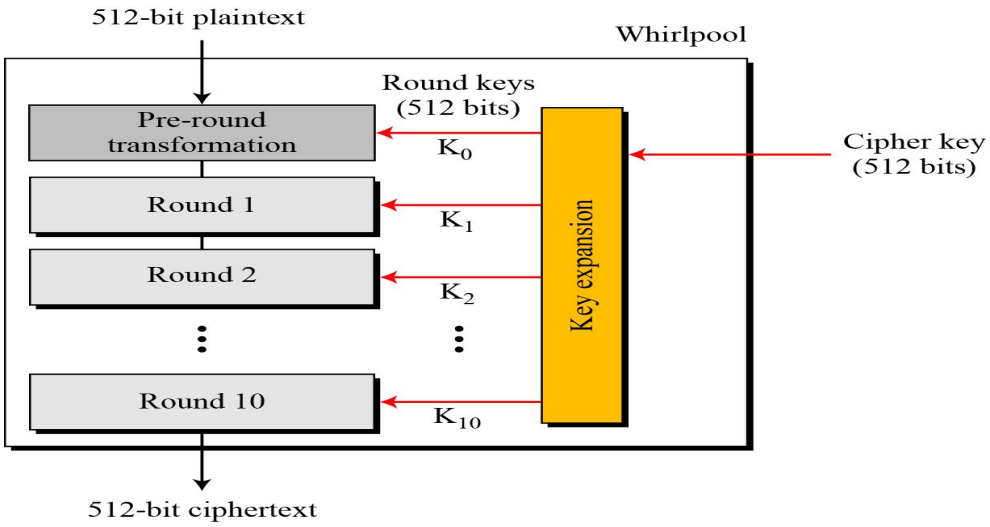
Whirlpool Hash Function



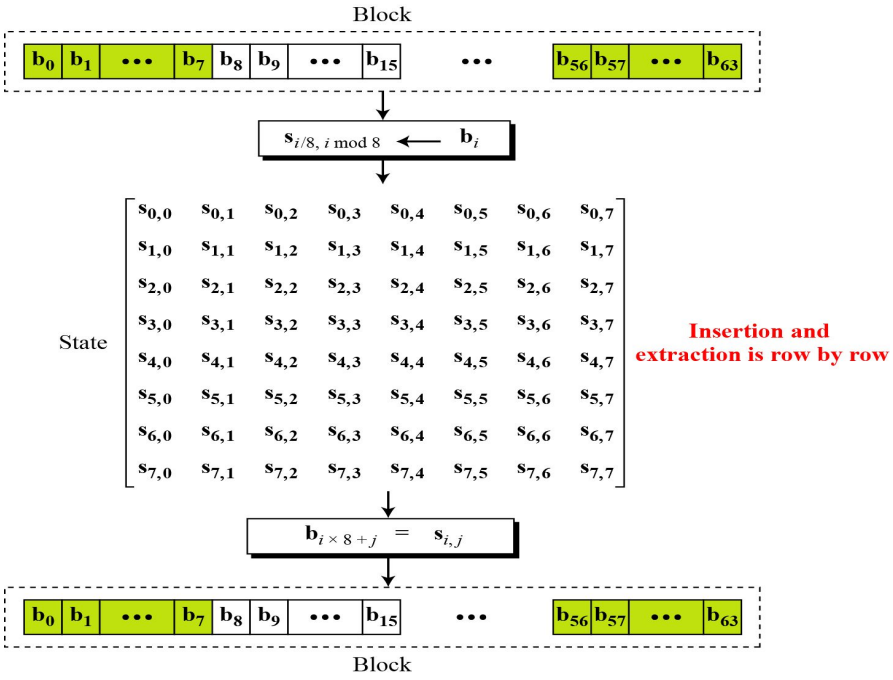
The message needs to be prepared for processing should be less than 2256 bits. A message needs to be padded before processed. The padding is a single 1-bit followed by the necessary number of 0-bits to make the length of the padding an odd multiple of 256 bits.

Rounds:

Whirlpool is a round cipher that uses 10 rounds. The block size and key size are 512 bits. The cipher uses 11 round keys, k0 to k10 each of 512 bits.

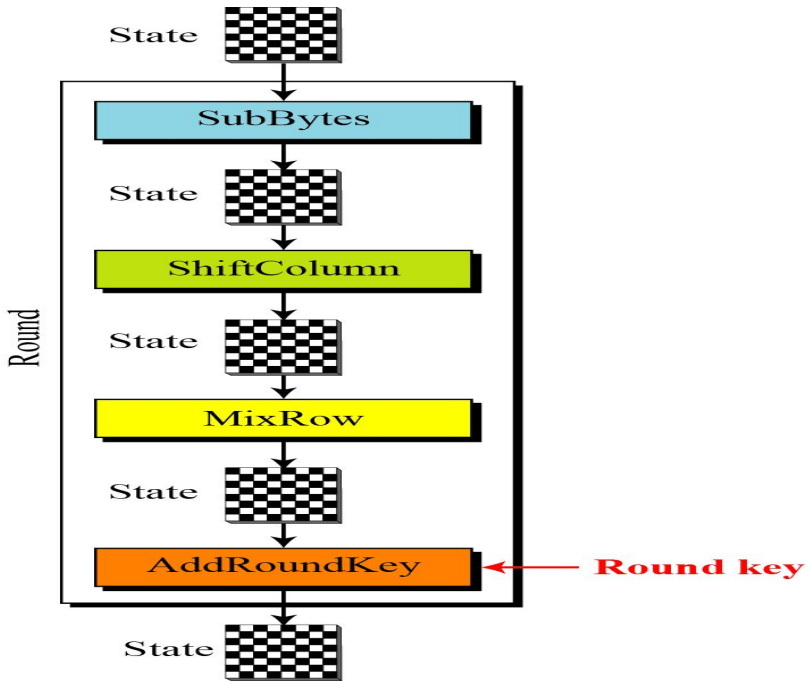


States and Blocks: Like AES cipher, the whirlpool cipher uses states and blocks. A block is considered as a row matrix of 64 bytes. A state in considered as a square matrix of 8 x 8 bytes.



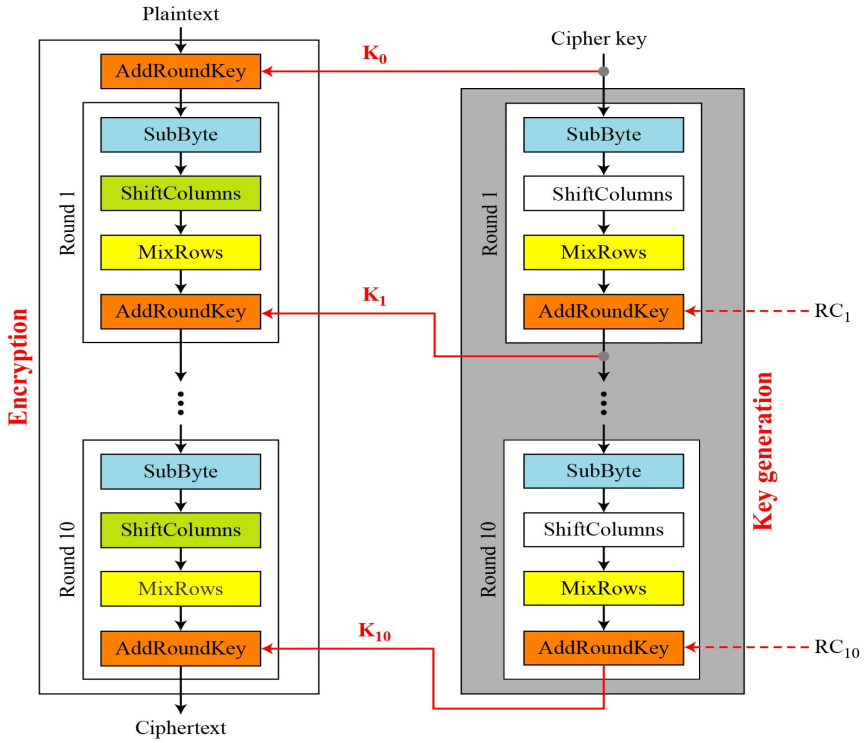
Structure of each round:

Each round uses four transformations.



Key Expansion:

Whirlpool uses a copy of the encryption algorithm to create the round keys. The output of each round in the encryption is the round key for that round.



Explain the services provided by Digital Signatures

Message Authentication

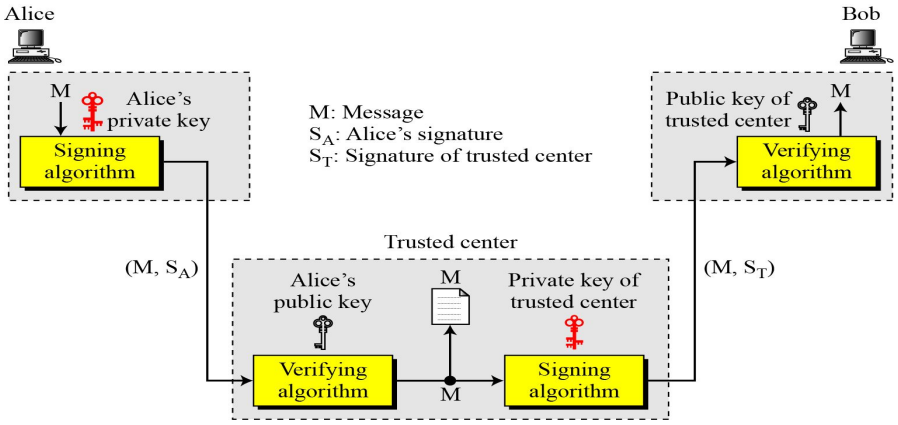
A secure digital signature scheme, like a secure conventional signature can provide message authentication.

Message Integrity

The integrity of the message is preserved even if we sign the whole message because we cannot get the same signature if the message is changed.

Nonrepudiation

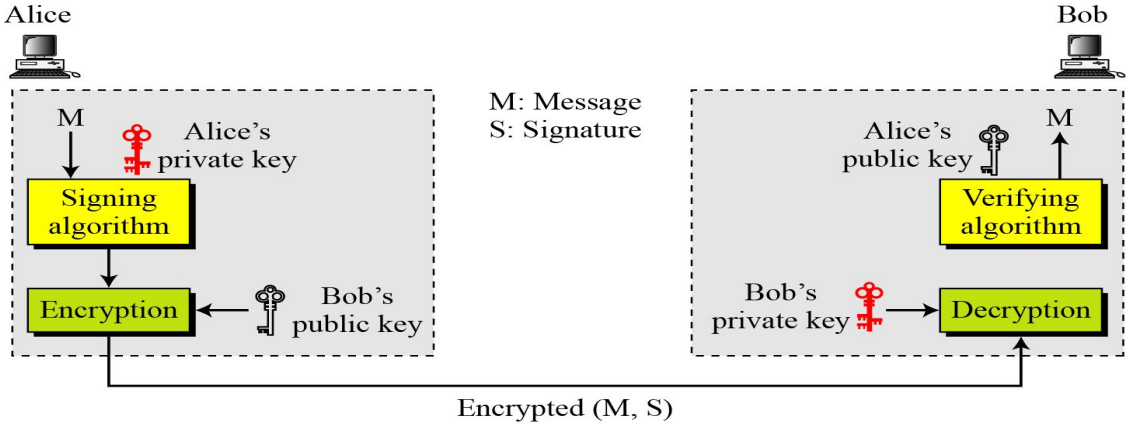
if alice signs a message and then denies it, can bob later prove that alice actually signed it. One solution is a trusted third party. People can create an established third party among themselves. It can solve many other problems concerning security services and key exchange



Alice creates a signature from her message and sends the message, her identity, Bobs identity and the signature to the center. The center after checking that alices public key is valid, verifies through alices public key that the message came from Alice. The center then saves a copy of message with the sender signature from the message. The center then sends the message, the new signature, alice identity and bobs identity to Bob. Bob verifies the message using the public key of the trusted center.

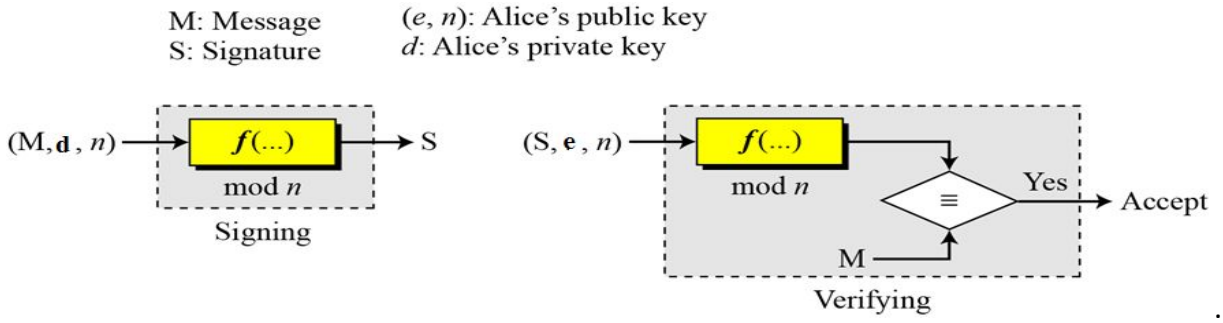
Confidentiality

A digital signature does not provide confidential communication. If confidentiality is required, The message and the signature must be encrypted using either a secret key or public key crypto system.



Explain RSA Digital Signature Schema

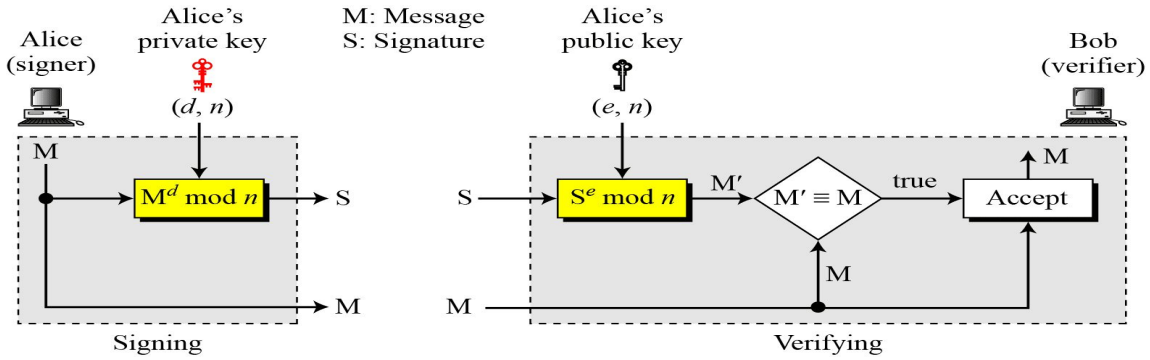
If RSA can be used for signing and verifying a message then it is called RSA Digital Signature Scheme. It changes the roles of private and public keys, First the private and public keys of the sender, not the receiver are used. Second, the sender uses her own private key to sign the document, the receivers uses the senders public key to verify it.



Key Generation

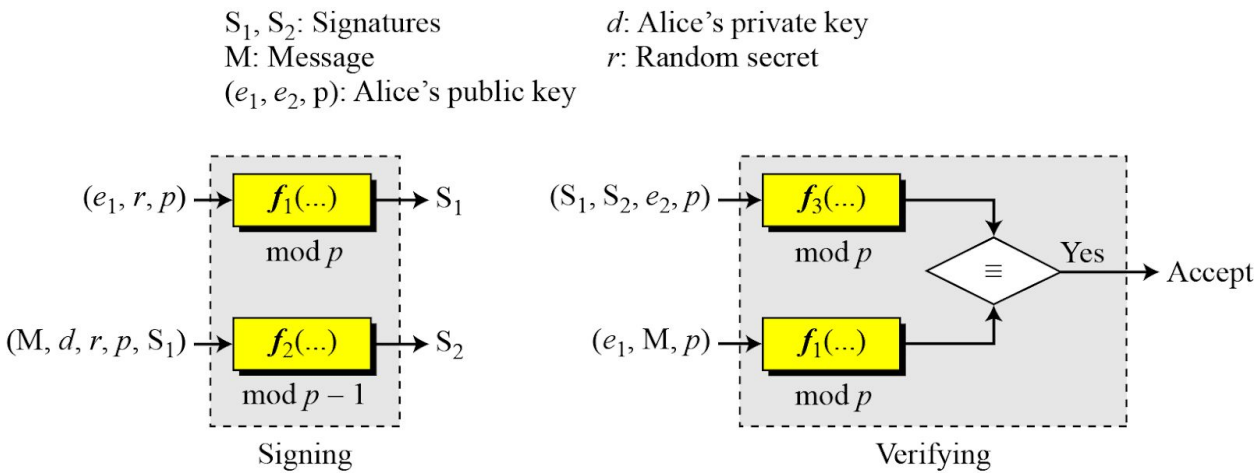
Key generation in the RSA digital signature scheme is exactly the same as key generation in the RSA cryptosystem. Alice chooses two prime numbers p and q and calculates n=pxq, alice calculates Ø(n)=(p-1)x(q-10. She then chooses e, the public exponent and calculates d, the private exponent such that e x d = 1 mod Ø(n). Alice keeps d, she publicly announces n and e.

Signing and Verifying



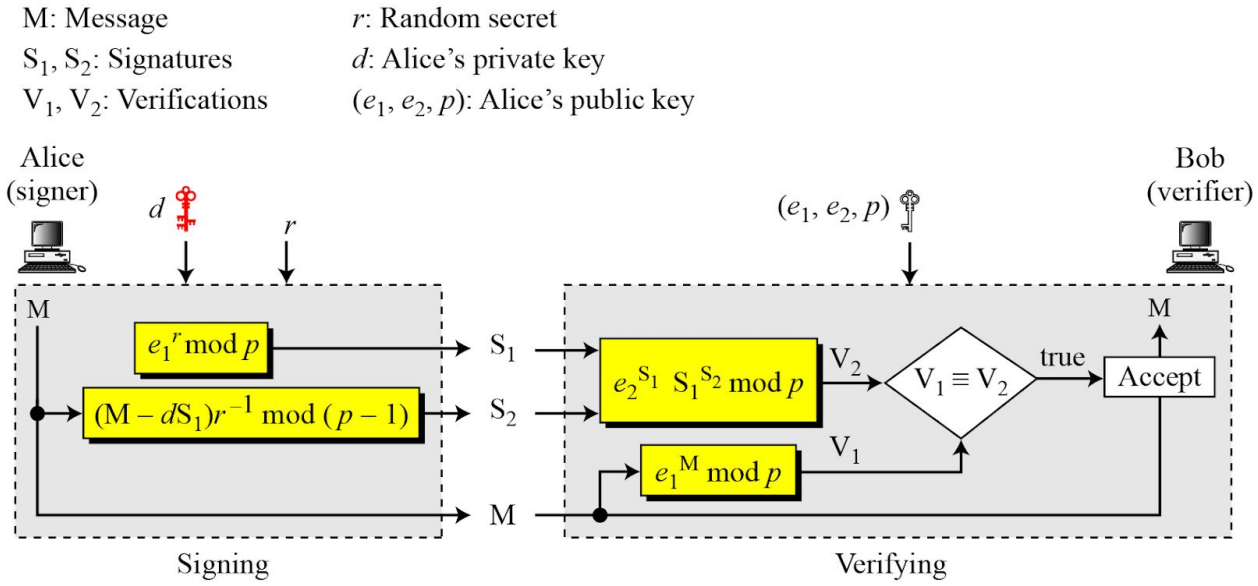
Explain Elgamal Digital Signature Scheme

Elgamal digital signature scheme uses Elgamal Cryptosystems . In signing process two functions create two signatures, in the verifying process the ouputs of two functions are compared for verification. Same function is used for both signing and verifying but the functions use two different inputs.



Key Generation: the Key generation procedure is exactly same as the one used in the elgamal cryptosystem. Let p be a prime number large enough. Let e1 be a primitive element in Zp^*. Alice selects her private key d to be less than p-1. She calculates $e2=e1^d$. Alice public key is the tuple (e1,e2,p) and private key is d.

Verifying and Signing



Signing

1. Alice chooses a secret random number r.
2. Alice calculates the first signatures S1=e1r mod p
3. Alice calculates the second signature S2=(M-dxS1) x r^{-1} mod (p-1)
4. Alice sends M, S1 and S2 to Bob

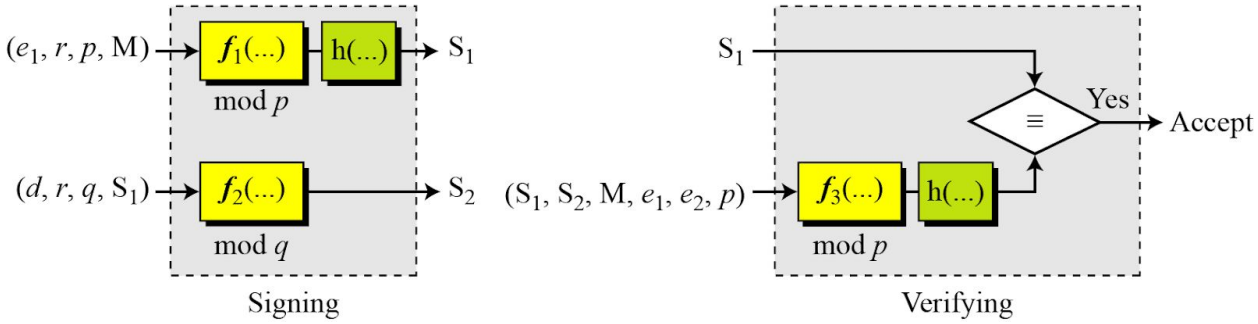
Verifying

1. Bob checks to see if 0< S1 < p
2. Bob checks to see if 0 < s2 < p-1
3. Bob calculates $V1=e1^{M} \bmod p$
4. Bob calculates $V2=e2^{S1} x S1^{S2} \bmod p$
5. If V1 is congruent to V2, the message is accepted, otherwise it is rejected.

Schnorr Digital Signature Scheme

The problem with ElGamal digital signature scheme is that p needs to be very large . The recommendation is a p of at least 1024 bits. To reduce the size of the signature, Schnorr proposed a new scheme based on ElGamal, but with a reduced signature size.

S_1, S_2: Signatures — (d): Alice's private key
M: Message — r: Random secret
(e_1, e_2, p, q): Alice's public key

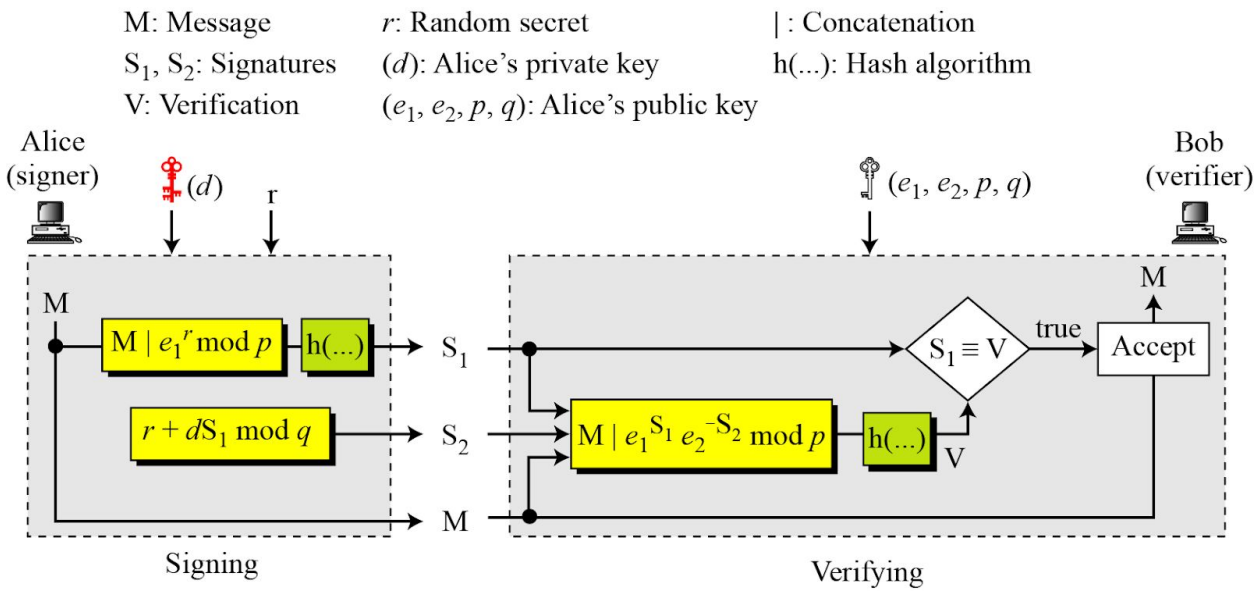


In the signing process, two functions create two signatures, in the verifying process, the output of one function is compared t the first signature for verification.

Key Generation:

1) ***Alice selects a prime p, which is usually 1024 bits in length.***
2) ***Alice selects another prime q.***
3) ***Alice chooses e_1 to be the qth root of 1 modulo p.***
4) ***Alice chooses an integer, d, as her private key.***
5) ***Alice calculates $e_2 = e_1^d \bmod p$.***
6) ***Alice's public key is (e_1, e_2, p, q); her private key is (d).***

Verifying and Signing



Signing

1. Alice chooses a random number r.

2. Alice calculates $S_1 = h(M|e_1^r \bmod p)$.

3. Alice calculates $S_2 = r + d \times S_1 \bmod q$.

4. Alice sends M, S_1, and S_2.

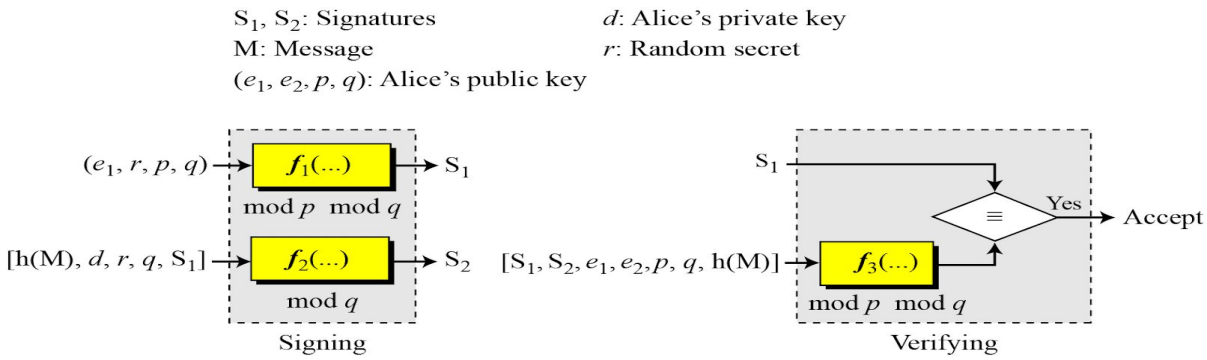
Verifying Message

1. Bob calculates $V = h(M \mid e_1^{S2} e_2^{-S1} \bmod p)$.

2. If S_1 is congruent to V modulo p, the message is accepted;

Explain Digital Signature Standard(DSS)

DSS was adopted by the NIST in 1994. DSS is based on the ElGamal scheme with some ideas from the Schnorr scheme. DSS has been criticized from the time it was designed regarding the secrecy of DSS Design. The second complaint regards the size of the prime 512 bits.

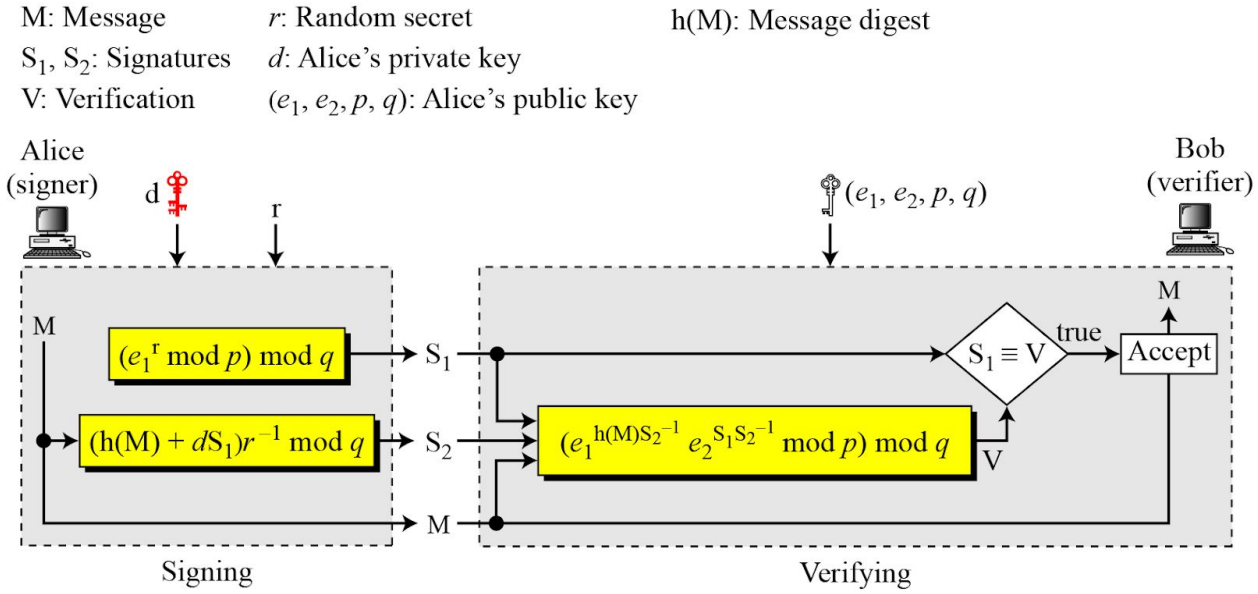


In Signing process, two functions create two signatures, in the verifying process, the output of one function is compared to the first signature for verification. This scheme uses tow public moduli p and q. functions 1 and 3 are both p and q function 2 uses only q.

Key Generation.

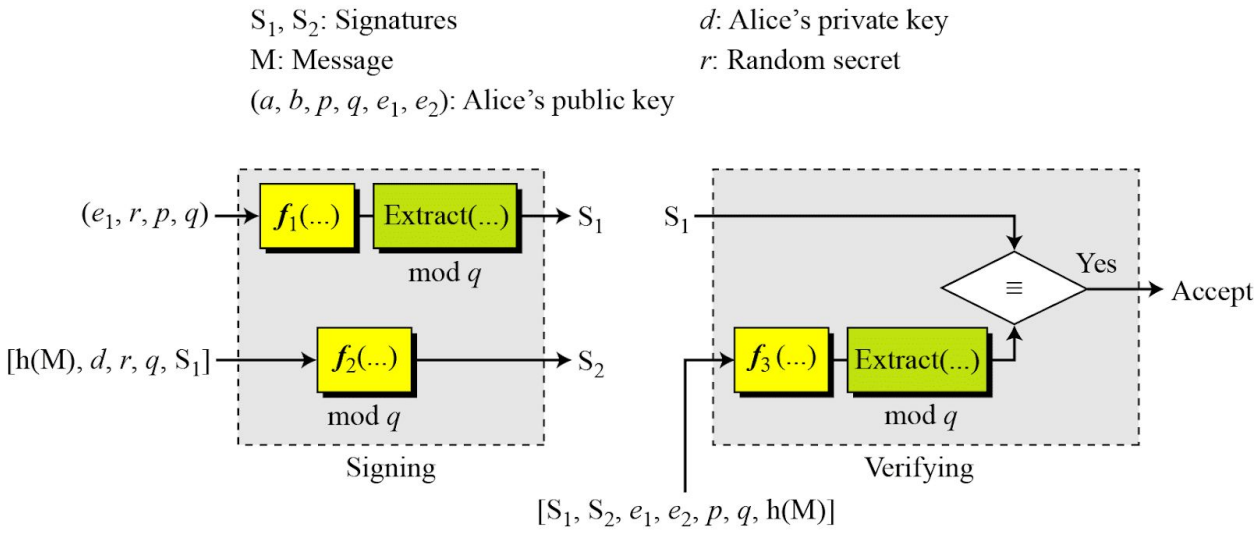
1) ***Alice chooses primes p and q.***
2) ***Alice uses $<Z_p^*, \times>$ and $<Z_q^*, \times>$.***
3) ***Alice creates e_1 to be the qth root of 1 modulo p.***
4) ***Alice chooses d and calculates $e_2 = e_1^d$.***
5) ***Alice's public key is (e_1, e_2, p, q); her private key is (d).***

Signing and Verifying



Elliptic Curve Digital Signature Scheme

It is based on DSA based on elliptic curves. The scheme sometimes is referred as ECDSA(Elliptic curve DSA). The general idea behind ECDSS

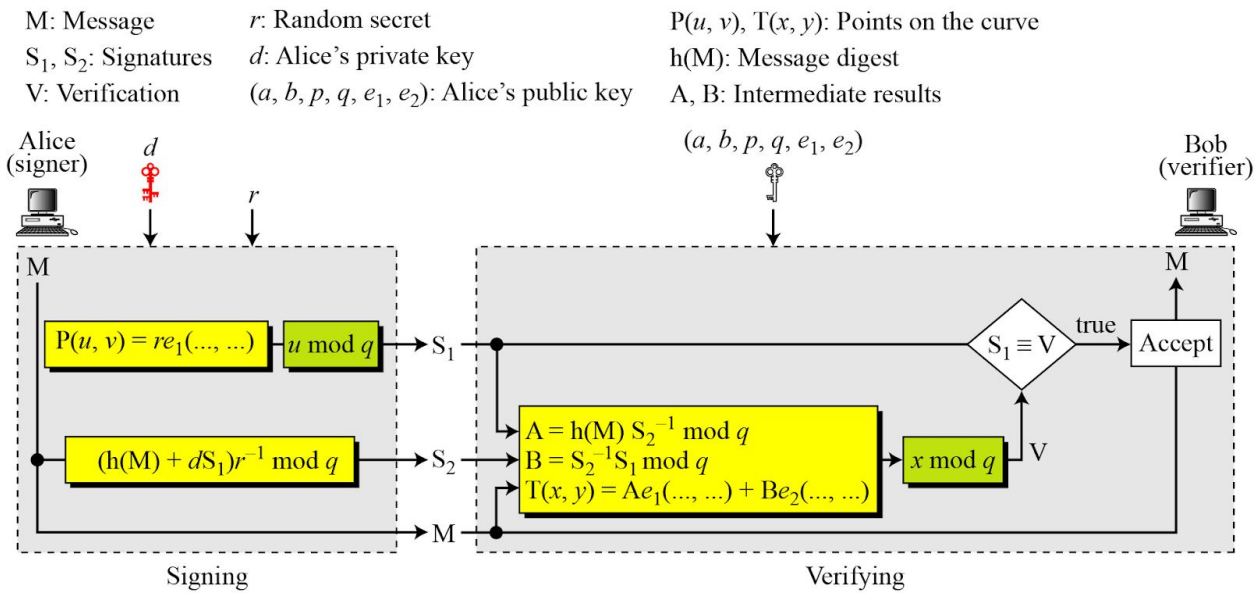


In the signing process, two functions and an extractor create two signatures, in the verifying process the output of one function is compared to the first signature for verification. Functions f1 and f3 acturally create points on the curve. The first creates a new point from the signers private key. The second creates a new point from the signers two public keys. Each extractor extracts the first coordinates of the corresponding pont in modular arithmetic

Key Generation

1) ***Alice chooses an elliptic curve $E_p(a, b)$.***
2) ***Alice chooses another prime q the private key d.***
3) ***Alice chooses $e_1(\ldots, \ldots)$, a point on the curve.***
4) ***Alice calculates $e_2(\ldots, \ldots) = d \times e_1(\ldots, \ldots)$.***
5) ***Alice's public key is (a, b, p, q, e1, e2); her private key is d.***

Signing and Verifying



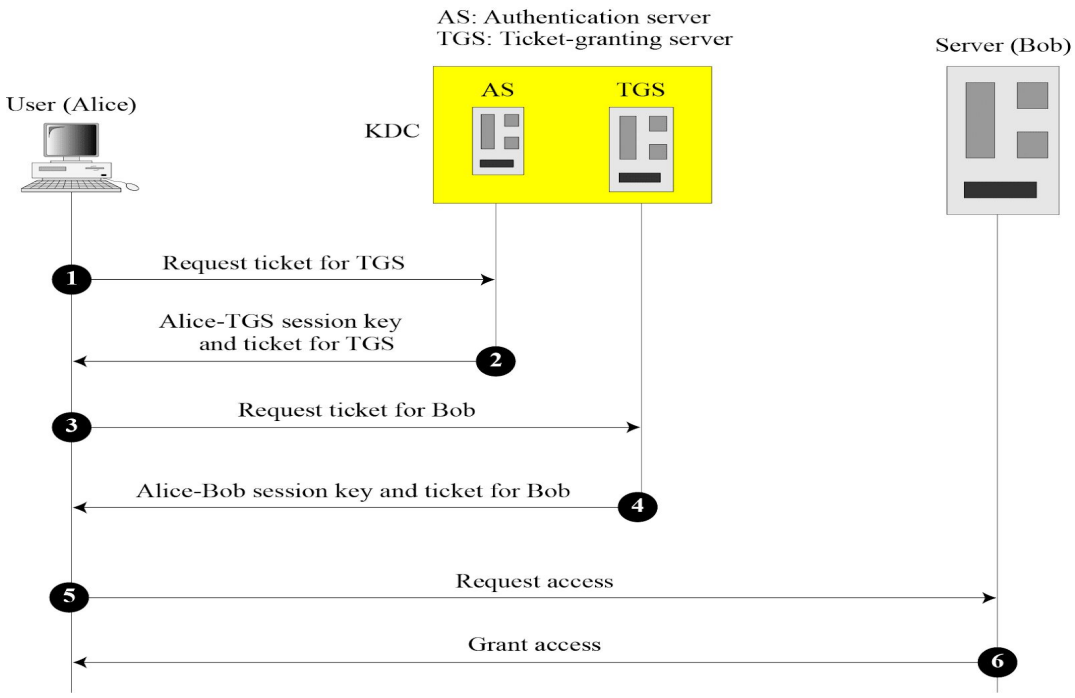
KERBEROS

Kerberos is an authentication protocol, and at the same time a KDC, that has become very popular. Several systems, including Windows 2000, use Kerberos. Originally designed at MIT, it has gone through several versions.

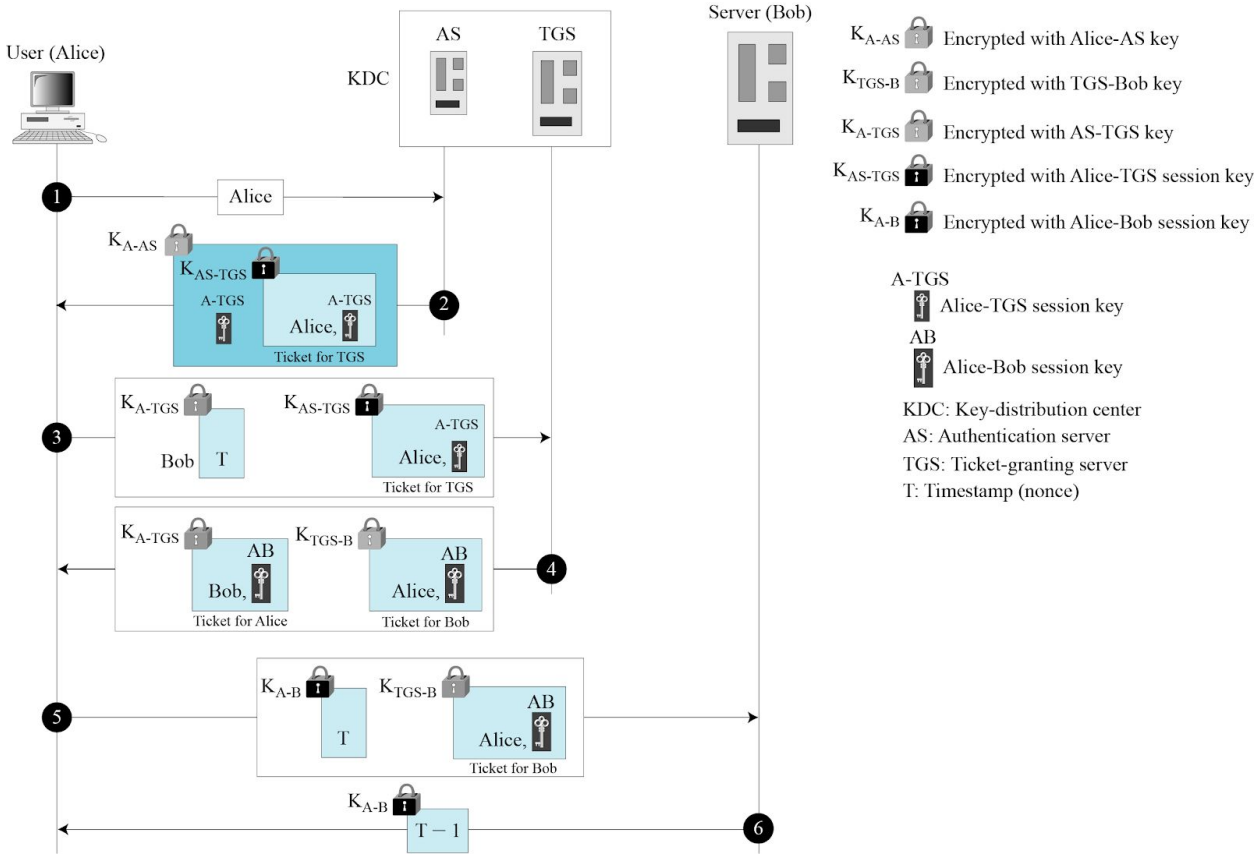
Servers

Three servers are involved in the Kerberos protocol:

1. Authentication Server(AS): It is the KDC in the kerboroes protocol. Each user registers with the AS and is granted a user identity and a password. The AS has a database with these identities and the corresponding passwords. The AS verifies the user, issues a session key to be used between Alice and the TGS, and sends a ticket for the TGS
2. Ticket Granting Server(TGS): TGS issues a ticket for the real server. It also provides the session key(K_{AB}) between Alice and Bob. Kerberos has separated user verification from the issuing of tickets. In the way, though Alice verifies her ID just once with the AS, she can contact the TGS multiple times to obtain tickets for different real servers.
3. Real Server: The real server(Bob) provides services for the user. Kerberos is designed for a client server program, such as FTP, in which a user uses the client process to access the server process. Kerboros is not used for person to person authentication.

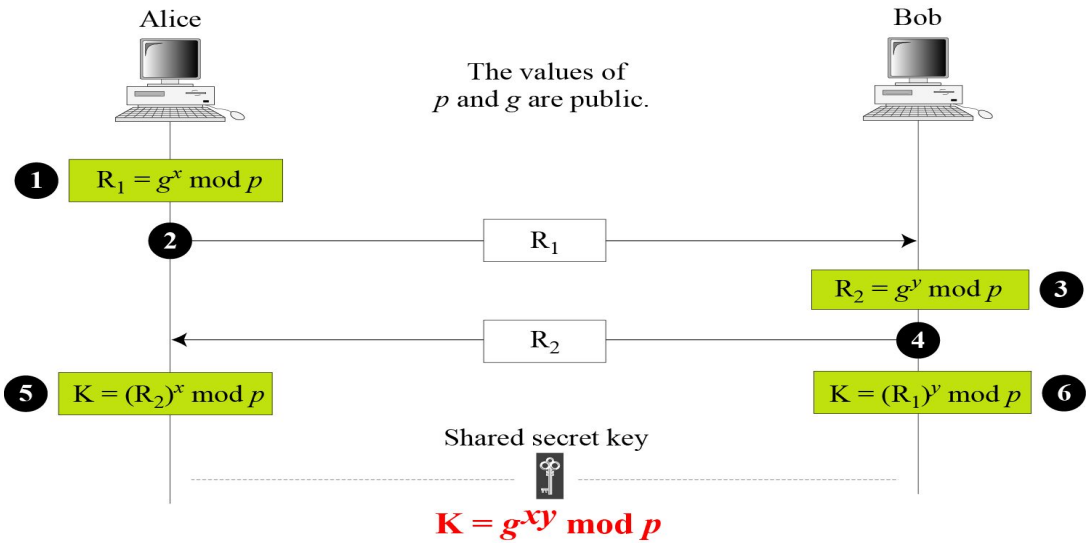

AS: Authentication server
TGS: Ticket-granting server
Server (Bob)
User (Alice)
KDC
AS
TGS
Request ticket for TGS
Alice-TGS session key
and ticket for TGS
Request ticket for Bob
Alice-Bob session key and ticket for Bob
Request access
Grant access

Operation:


User (Alice)
KDC
AS
TGS
Server (Bob)
Alice
Ticket for TGS
Bob
T
Ticket for Alice
Ticket for Bob
Encrypted with Alice-AS key
Encrypted with TGS-Bob key
Encrypted with AS-TGS key
Encrypted with Alice-TGS session key
Encrypted with Alice-Bob session key
A-TGS
Alice-TGS session key
AB
Alice-Bob session key
KDC: Key-distribution center
AS: Authentication server
TGS: Ticket-granting server
T: Timestamp (nonce)

Deffie Hellman Key Agreement

Deffie Hellman protocol create a symmetric session key without the need of a KDC. Before establishing a symmetric key, the two parties need to coose two numbers p and g. the first number, p is a large prime number on the order of 300 decimal digits(1024 bits). The second number g is a generator of order p-1 in the group <Zp*,x>. these two do not need to be confidential. They can be sent through the internet, they can be public.



The steps are as follows:

1. Alice chooses a large random number x such that 0<= x<= p-1 and calculates R1=g^x mod p.
2. Bob chooses another large random number y such that 0<= y <= p-1 and calculates R2=g^y mod p.
3. Alice sends R1 to Bob. Note that Alice does not send the value of x she sends only R1
4. Bob sends R2 to alice. Again note that bob does not send the value of y, he sends only R2.
5. Alice calculates K=$(R2)^x$ mod p.
6. Bob also calculates K=$(R1)^y$ mod p

Examle: Assume that *g* = 7 and *p* = 23. The steps are as follows:

1. **Alice chooses *x* = 3 and calculates R_1 = 73 mod 23 = 21.**

2. **Bob chooses y = 6 and calculates R_2 = 76 mod 23 = 4.**
3. **Alice sends the number 21 to Bob.**
4. **Bob sends the number 4 to Alice.**
5. **Alice calculates the symmetric key K = 43 mod 23 = 18.**
6. **Bob calculates the symmetric key K = 216 mod 23 = 18.**
7. **The value of K is the same for both Alice and Bob; g^{xy} mod p = 718 mod 35 = 18.**

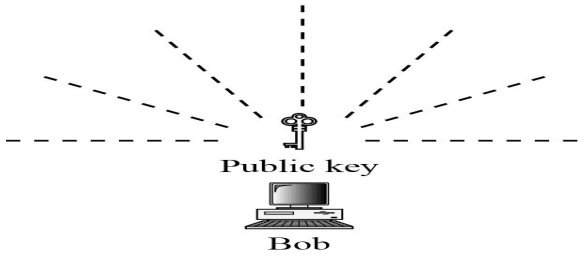
Explain the ways of distributing public Keys

In public key cryptography every one has access to every ones public key. There fore public keys must be available public. The ways of distributing public keys are

1.Public Announcement

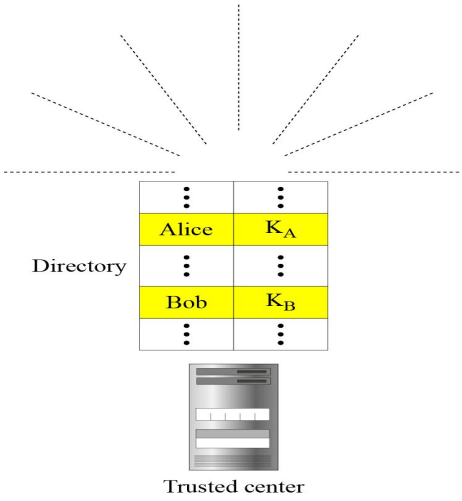
He naïve approach is to announce public keys publicly. Bob can put his public key on his website or announce it in a local or national news paper. When alice wants to send a confidential message to Bob, she can obtain Bob's public key from his site.

This approach is not secure and it is subject to forgery. Eve could make such a public announcement, before bob can react, damage could be done. Even can fool Alice into sending a message that is intended for Bob. This approach is also vulnerable if Alice requests Bob public key. Even can intercept Bob's response and substitute her own forged public key.



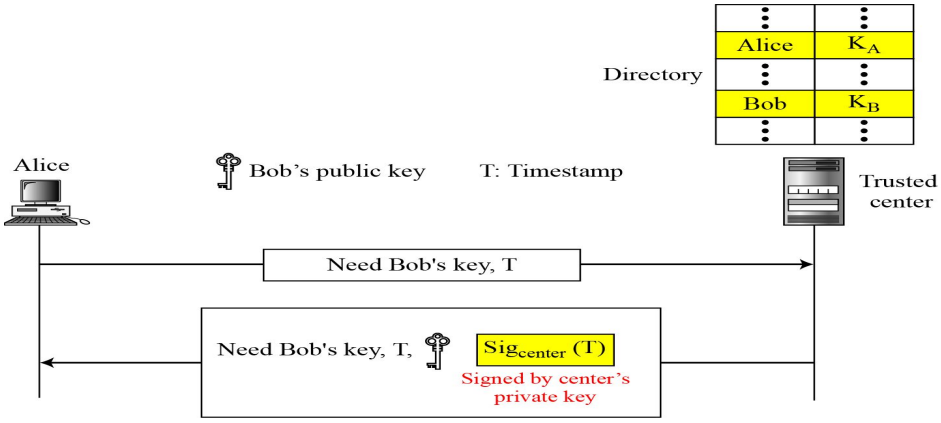
2.Trusted Center

A more secure approach is to have a trusted center retain a directory of public keys which is update dynamically. Each user can select a private and public key, keep the private key, and deliver the public key for insertion into the directory. The center requires that each user register in the center and prove his or identity. The directory can be publicly advertised by the trusted center. The center can also respond to any inquiry about a pubic key.



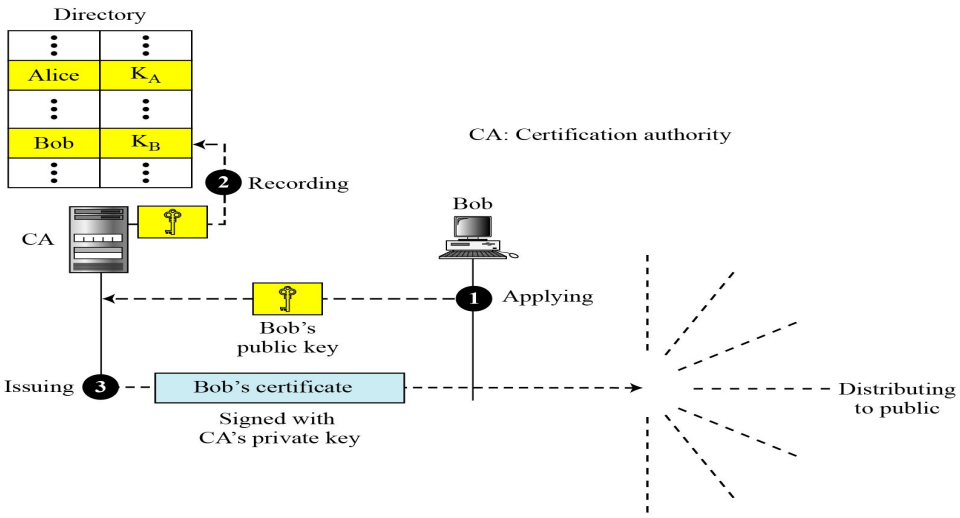
3.Controlled Trusted Center

A higher level of security can be achieved if there are added controls on the distribution of the pubic key. The public key announcements can include a time stamp and be signed by an authority to prevent interception and modification of the response. If Alice needs to know Bobs public key, she can send a request to that center including Bobs name and timestamp signed with the private key of the center. The center responds with Bobs public key, the original request and the timestamp signed with the private key of the center.



4.Certificaiton authority

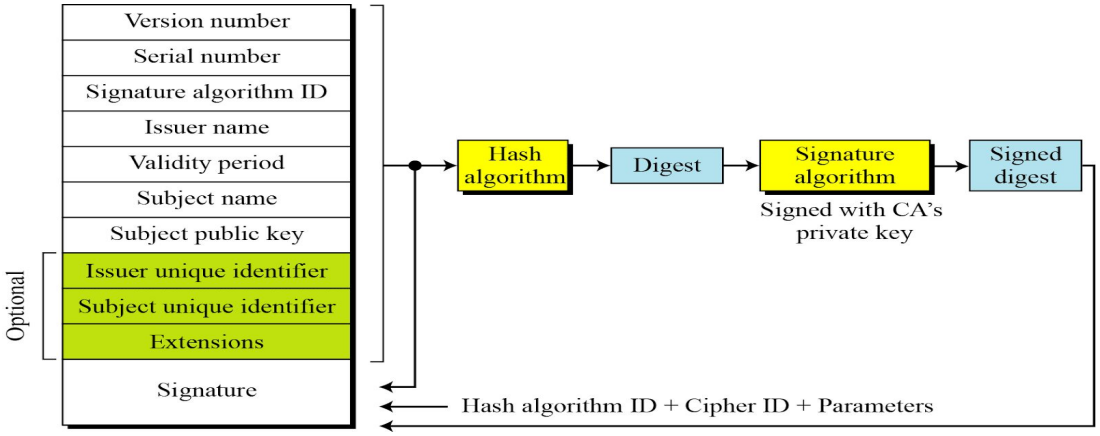
When any one needs a public key he goes to a certification authority(CA), a federal or state organization that binds the public key to an entity and issues a certificate. The CA checks the Bobs identification and issues the certificate which cannot be forged.



X.509

ITU has designed X.509, which is a way to describe the certificate in a structured way. It uses well known protocol called ASN.1(Abstract Syntax Notation 1).

Format of a Certificate



Version Number: the version number started at 0 and the current version is 2

Serial Number: This field defines a number assigned to each certificate. The value is unique for each certificate

Signature algorithm ID: it identifies the algorithm used to sign the certificate

Issuer Name:it identifies the certification authority that issued the certificate.

Validity Period: this field defines the earliest name and latest time the certificate is valid

Subject Name: it defines the entitiy to which the public key belongs

Subject public key: this field defines the owners public key

Issuers unique identifier: this optional field allows two issuers to have the same issuer field value

Subject unique identifier: this optional field allows two different subjects to have the same subject field value.

Extensions: this optional field allows issuers to add more private information to the certificate

Signature: it contains three sections. The first contains all other fields in the certificate. The second contains the digest of the first section and third section contains the algorithm identifier used to create the second section

Certificate Renewal

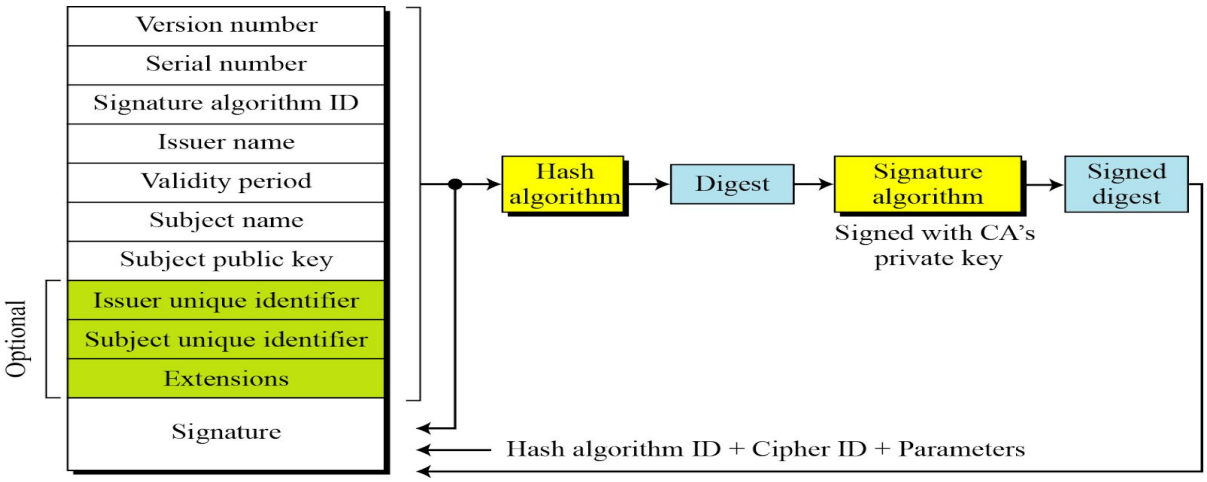
Each certificate has a period of validity. If there is no problem with the certificates, the CA issues a new certificate before the old one expires.

Certificate Revocation:

In some cases a certificate must be revoked before it expires. The certificate can be revoked in some case like

1. The users private key might have compromised
2. The CA is no longer willing to certify the user
3. The CA private key may be compromised.

The certificate revocation list format is



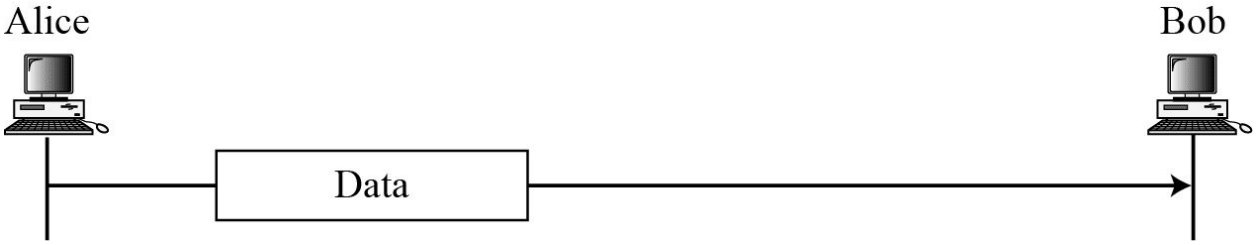
UNIT-V

PGP

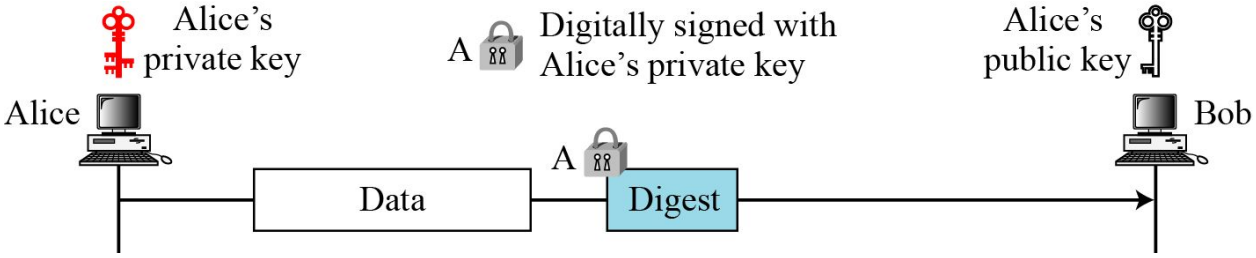
Pretty Good Privacy was invented by Phil Zimmermann to provide email with privacy, integrity and authentication. PGP can be used to create a secure email message or to storage a file securely for future reference

Scenarios of PGP

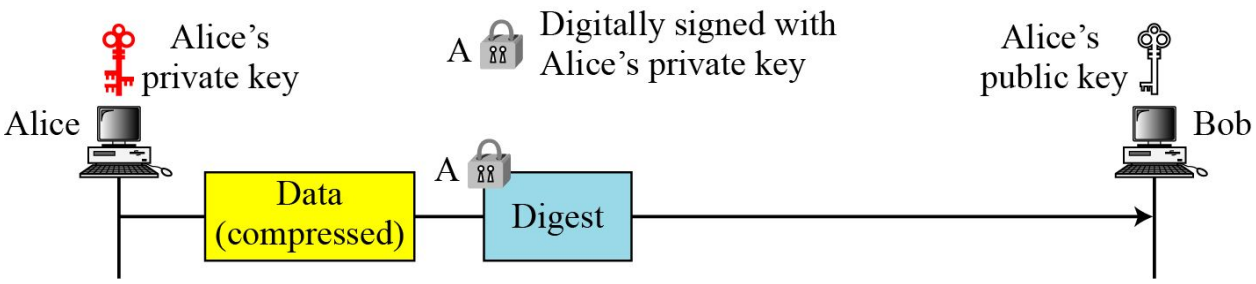
1.Plaintext: the simplest scenario s to send the email message in plain text. There is no message integrity or confidentiality.



2.Message Integrity: In this method digest of the message is created and signts it with his private key. When Bob receives the message he verifies the message by using with Alice Public key.

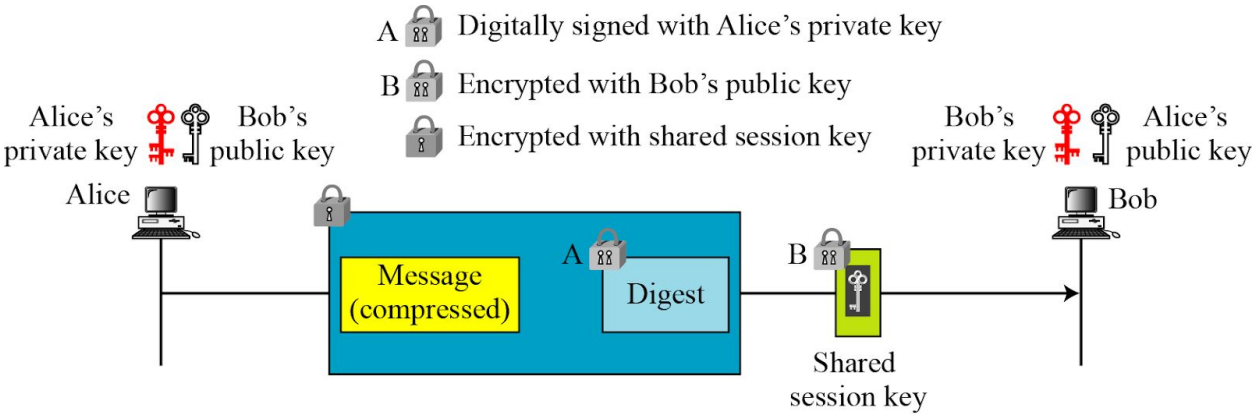


3.Compression: A further improvement is to compress the message and digest to make the packet more compact. This improvement has no security benefit but it eases the traffic



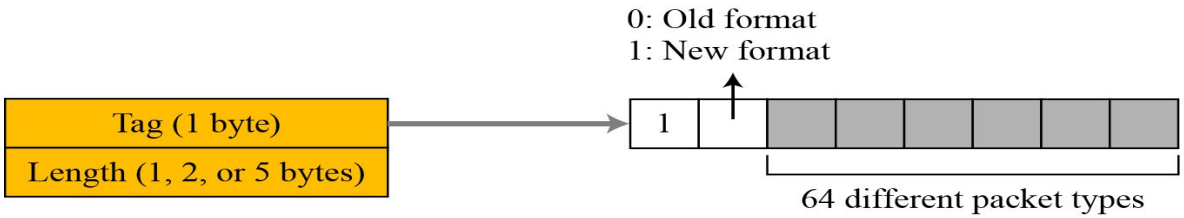
4.Confidentiality with one time Session Key

Confidentiality in an email system can be achieved using conventional encryption with a one time session key, Alice can create a session key and use the session key to encrypt the message and digest, and send the key to itself with the message



Explain the packet types of PGP

A message in PGP consists of one or more packets. The format of the packet header is



Tag: it is a 8 bit flag. The first bit is always 1. The second bit is 1 if we are using the latest version. The remaining six bits can define up to 64 different packet types.

Table 16.12 *Some commonly used packet types*

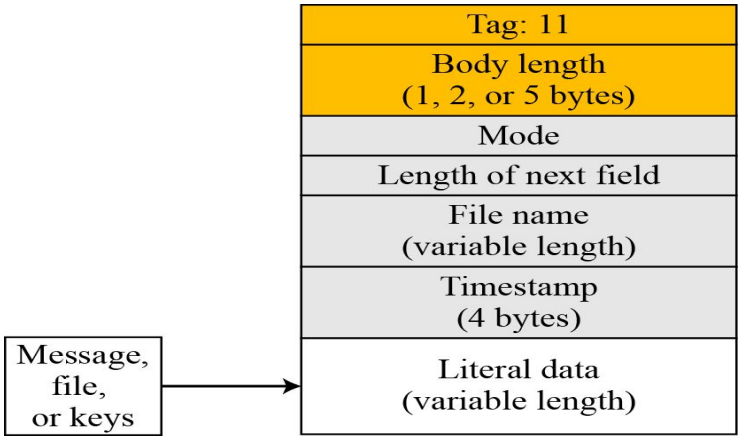
Value	*Packet type*
1	Session key packet encrypted using a public key
2	Signature packet
5	Private-key packet
6	Public-key packet
8	Compressed data packet
9	Data packet encrypted with a secret key
11	Literal data packet
13	User ID packet

Length:

The length field defines the length of the entire packet in bytes. The size of this field is variable it can be 1,2 or 5 bytes.

Literal Data Packet:

The literal data packet is the packet that carries or holds the actual data that is being transmitted or stored. The packet is the most elementary type of the message. It cannot carry any other packet.



Mode: the one byte field defines how data is written to the packet. The value of this field can be “b” for binary, “t” for text.

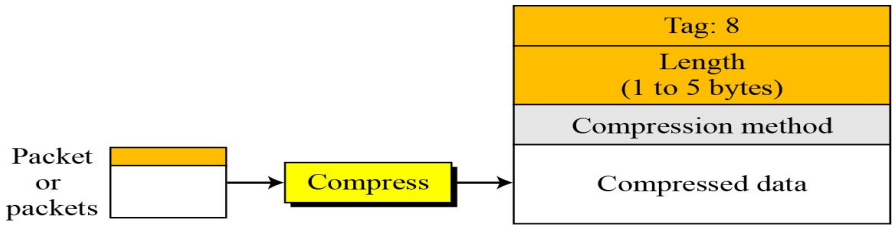
Length of the next field: the one byte field defines the length of the next field.

File Name: the variable length field defines the name of th file or message as an ASCII string.

TimeStamp: the four byte field defines the use of creation or last modification of the message. The value can be o, which means that the user chooses not to specify a time.

Literal Data: this variable length field carries the actual data in text or binary

Compressed Data packet: This packet carries compressed data packets.. The format of a compressed data packet is

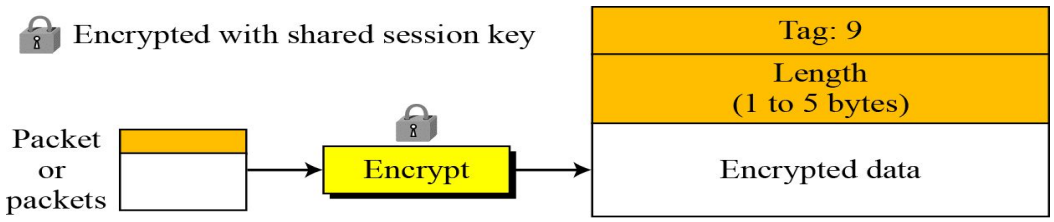


Compression method: this one byte field defines the compression method used to compress the data. The values defined for this field so far are 1(ZIP) and 2(ZLIP).

Compressed Data: This variable length field carries the data after compression.the data in this field can be one packet or the concatenation of two or more packets.

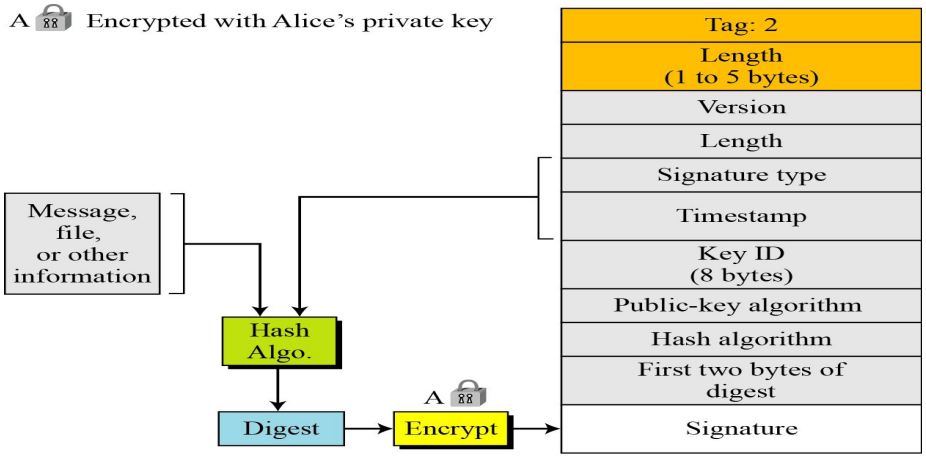
Data Packet Encrypted with Secret Key

This packet carries data from one packet or a combination of packets that have been encrypted using a conventional symmetric key algorithm. The packet carrying the one time session key must be sent before the packet.



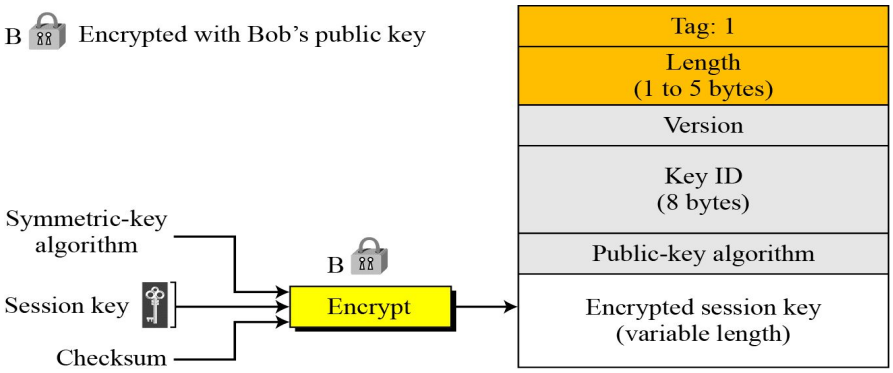
Signature Packet

A signature packet protects the integrity of the data.



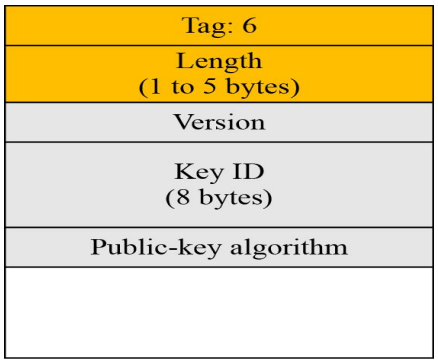
Session Key Packet encrypted with public key

This packet is used to send the session key encrypted with the receiver public key.



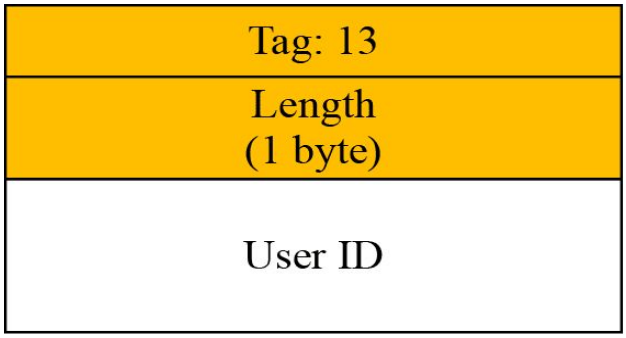
Public Key Packet:

This packet contains the public key of the sender. The format of the packet is



User ID Packet

This packet identifies a user and can normally associate the user ID with a public key of the sender.



User ID defines the user ID of the sender. It is normally the name of the user followed by an e-mail address and length field of the general header is only one byte.

MIME

EIectronic mail has a simple structure. It can send message NVT 7-bit ASCII format. In other words, it has some limitations. For example, it cannot be used for languages that are not supported by 7-bit ASCII characters (such as Arabic, Chinese, French,German, Hebrew, Japanese, and Russian). Also, it cannot be used to send binary files or video or audio

Multipurpose Internet Mail Extensions (MIME) is a supplementary protocol that allows non-ASCI data to be sent through e-mail. MIME transforms non-ASCII data at the sender site to NVT ASCII data and delivers it to the client MTA to be sent through the Internet. The message at the receiving side is transformed back to the original data.

MIME defines five headers that can be added to the original e-mail header section to define the transformation parameters:

l. MIME-version

2. Content-Type

3. Content-Transfer-Encoding

4. Content-Id

5. Content-Description

Data types and Sub types in MIME

MIME-Version

This header defines the version of MIME used. The current version is 1.1.

Content-Type

The content type and the content subtype are separated by a slash. Depending on the subtype, the header may contain other parameters.

Table 16.14 *Data types and subtypes in MIME*

Type	*Subtype*	*Description*
	Plain	Unformatted.
	HTML	HTML format.
Multipart	Mixed	Body contains ordered parts of different data types.
	Parallel	Same as above, but no order.
	Digest	Similar to Mixed, but the default is message/RFC822.
	Alternative	Parts are different versions of the same message.
Message	RFC822	Body is an encapsulated message.
	Partial	Body is a fragment of a bigger message.
	External-Body	Body is a reference to another message.
Image	JPEG	Image is in JPEG format.
	GIF	Image is in GIF format.
Video	MPEG	Video is in MPEG format.
Audio	Basic	Single channel encoding of voice at 8 KHz.
Application	PostScript	Adobe PostScript.
	Octet-stream	General binary data (eight-bit bytes).

Content Transfer Encoding

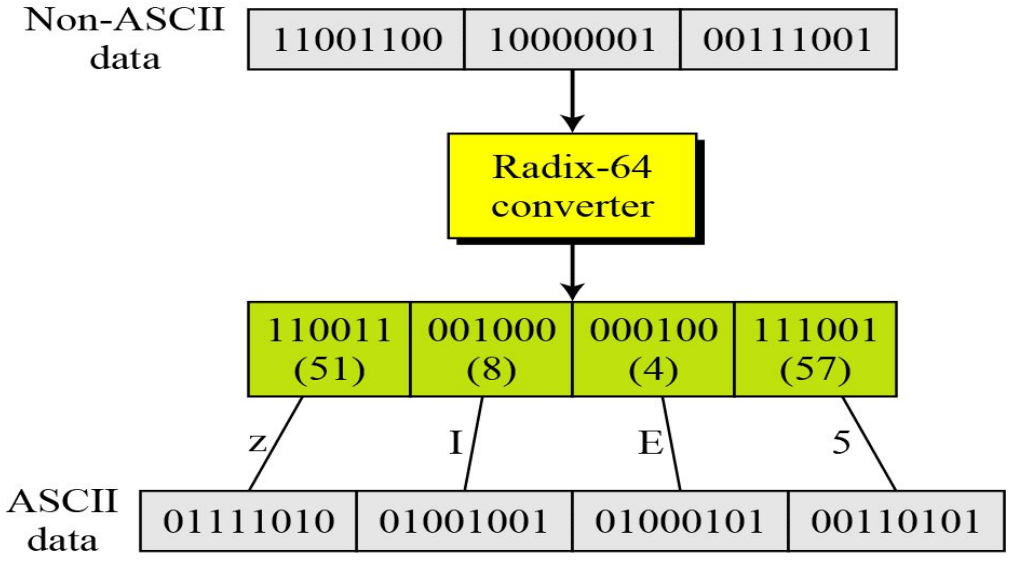
The five types of encoding methods are

Table 16.15 *Content-transfer-encoding*

Type	*Description*
7bit	NVT ASCII characters and short lines.
8bit	Non-ASCII characters and short lines.
Binary	Non-ASCII characters with unlimited-length lines.
Radix-64	6-bit blocks of data are encoded into 8-bit ASCII characters using Radix-64 conversion.
Quoted-printable	Non-ASCII characters are encoded as an equal sign followed by an ASCII code.

Radix 64

It transforms the type of data to printable characters, which can then be sent as ASCII characters or any type of character set supported by the underlying mail transfer mechanism. Radix 64 divides the binary data into 24-bit blocks. Each block is then divided into four sections, each mad of 6 bits



Quoted Printable: If a character is ASCII, it is sent as is. If a character is not ASCII, it is sent as three characters first character is = sign. The next two characters are the hexadecimal representation of the byte.

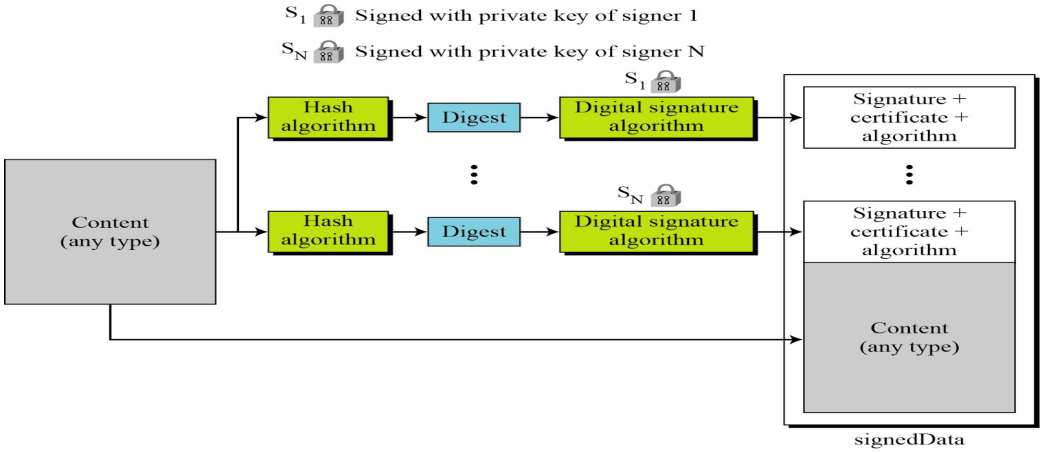
Explain how S/MIME provides authentication, privacy and confidentiality?

Signed Data Content Type:

This type provides only integrity of data. It contains any type and zero or more signature values. The encoded result is an object called signed data.

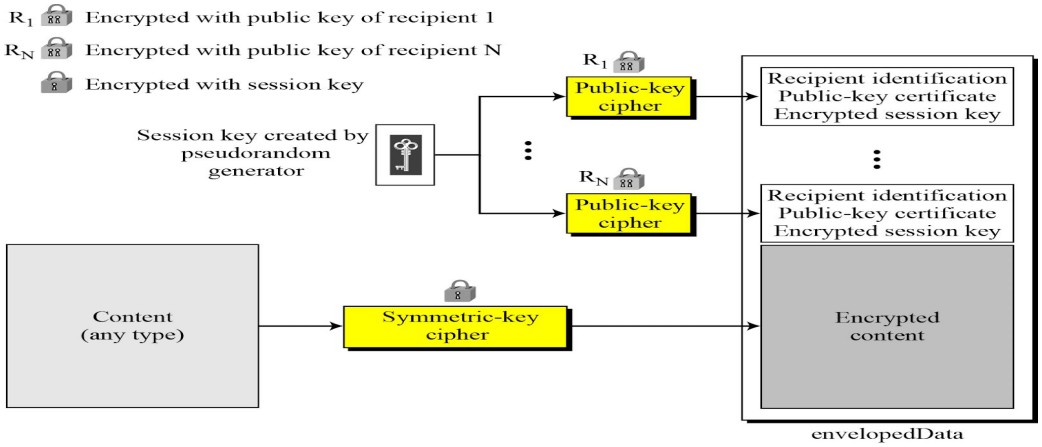
The following are the steps in the process.

1. For each signer, a message digest is created from the content using the specific hash algorithm
2. Each message digest is signed with the private key of the signer
3. The content, signature values, certificates and algorithms are then collected to create the signed data object



Enveloped Data Content type:

This type is used to provide privacy for the message. It contains any type and zero or more encrypted keys and certificates. The encoded result is an object called enveloped data.

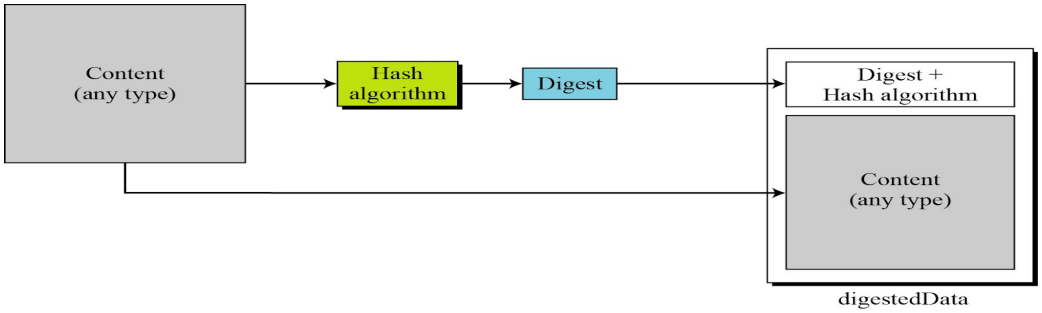


Digested Data Content Type

This type is used to provide integrity for the message. The result is normally used as the content for the enveloped data content type. The encoded result is an object called digested data.

The process of creating an object of this type is

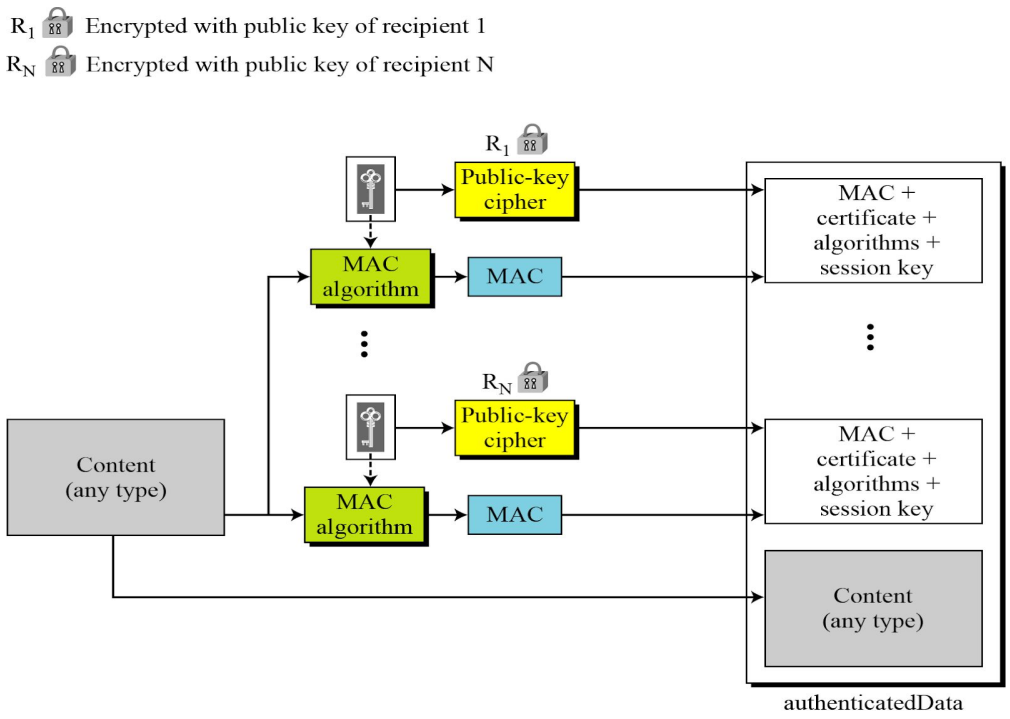
1. A message digest is calculated from the content
2. The message digest, the algorithm and the content are added together to create the digested data object



Authenticated Data Content Type

This type is used to provide authentication of the data. The object is called AuthenticatedData. The process is

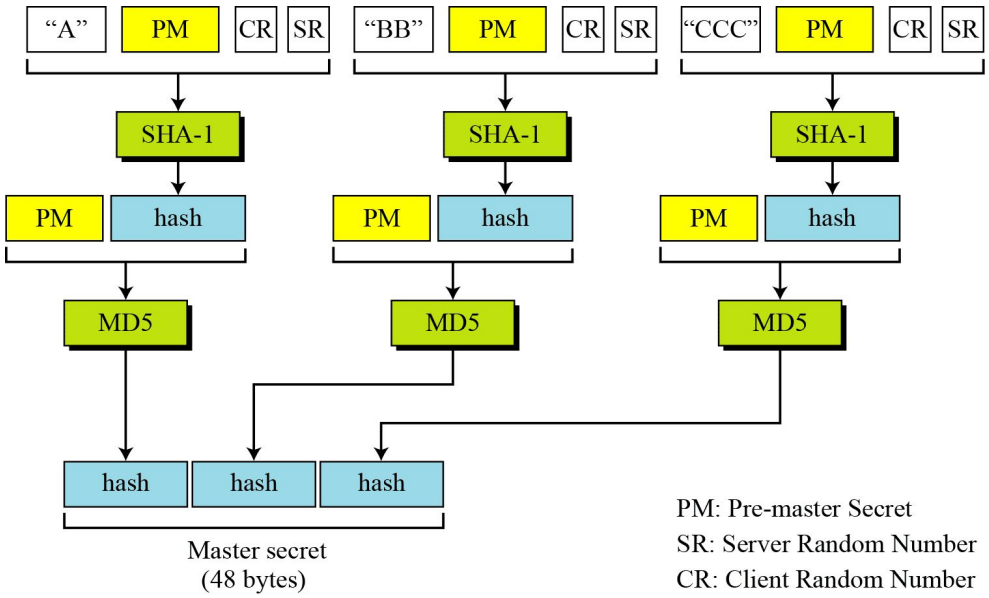
1. Using a pseudo random generator, a MAC key is generated for each recipient
2. The MAC key is encrypted with the public key of the receipient
3. The content, MAC, algorithms and other informations are collected together to form the authenticatedData Object.



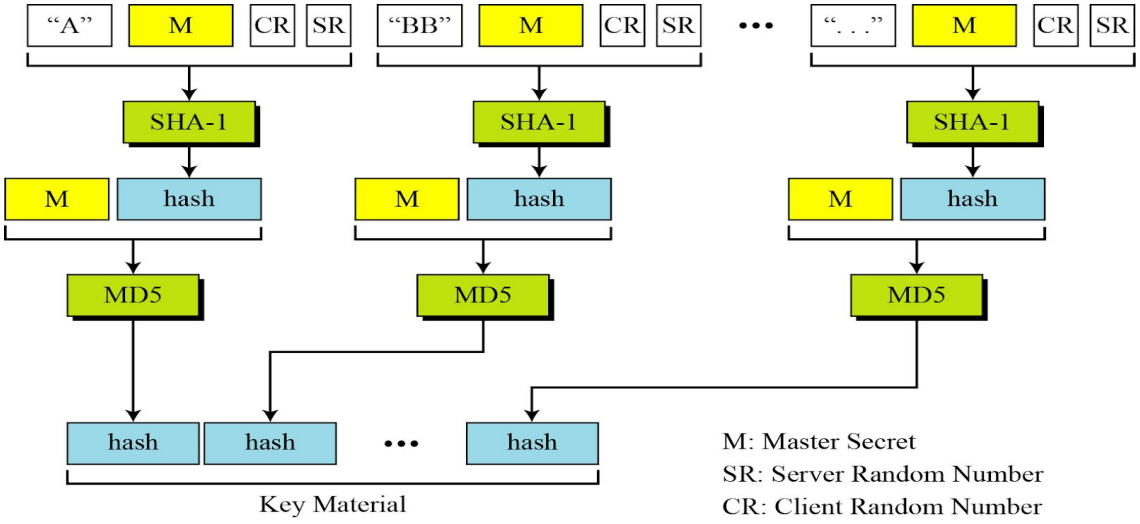
Explain Cryptographic parameter generation in SSL

To achieve message integrity and confidentiality, SSL needs six cryptographic secrets, four keys and two IV's. the client needs one key for message authentication(HMAC), one key for encryption and one IV for block encryption. The server needs the same. The parameters are generated using the following procedure.

1. The client and server exchange two randm numbers, one is created by the client and the other by the server.
2. The client and server exchange one pre master secret using one of the key exchange algorithms.
3. A 48 byte master secret is created from the pre master secret by applying two hash functions SHA-1 and MD5

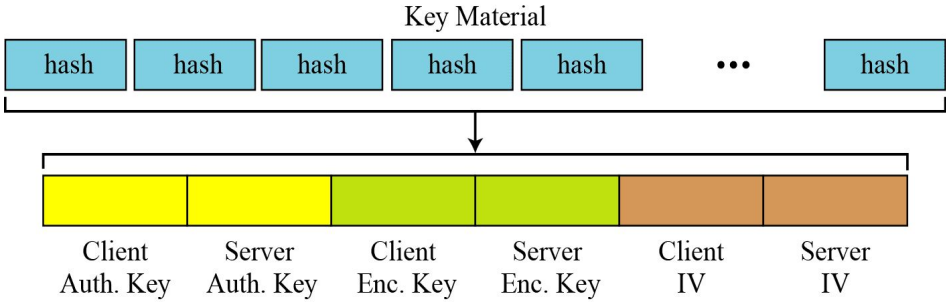


4. The master secret is used to create variable length key material by applying the same set of hash functions and prepending with the different constant



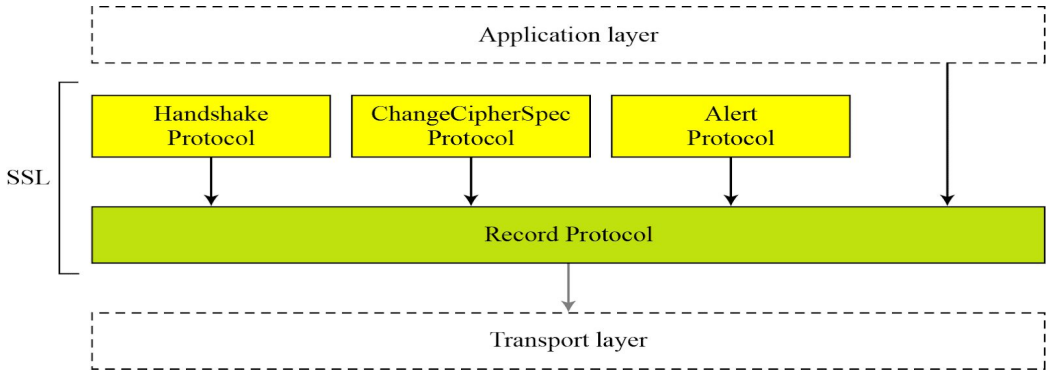
5. Six different keys are extracted from the key material

Auth. Key: Authentication Key
Enc. Key: Encryption Key
IV: Initialization Vector



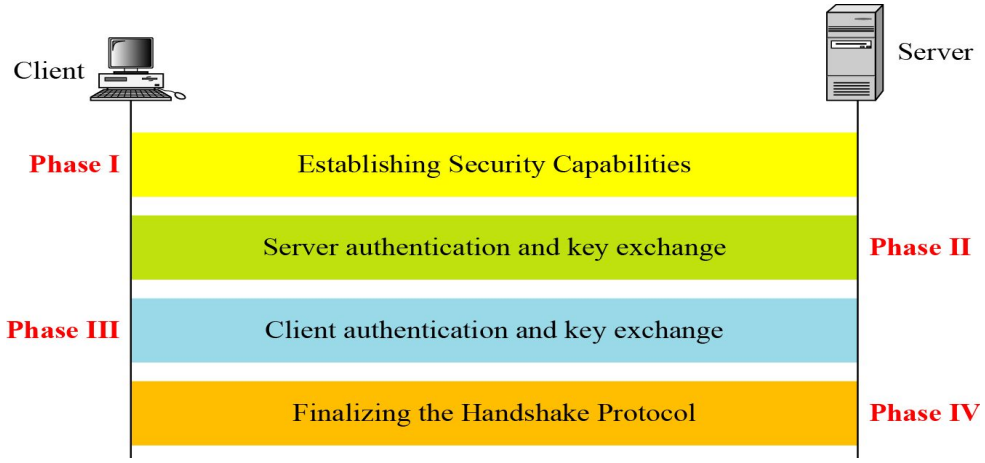
Explain the Hand shaking protocol in SSL

The four protocols used in the SSL are



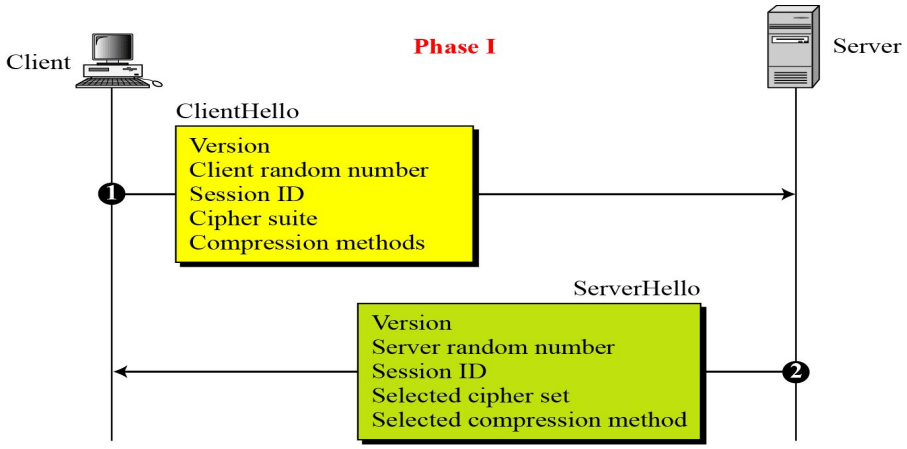
Handshake Protocol

The Handshake protocol uses messages to negotiate the cipher suite, to authenticate the server to the client and the client to the server if needed and to exchange information for building the cryptographic secrets.



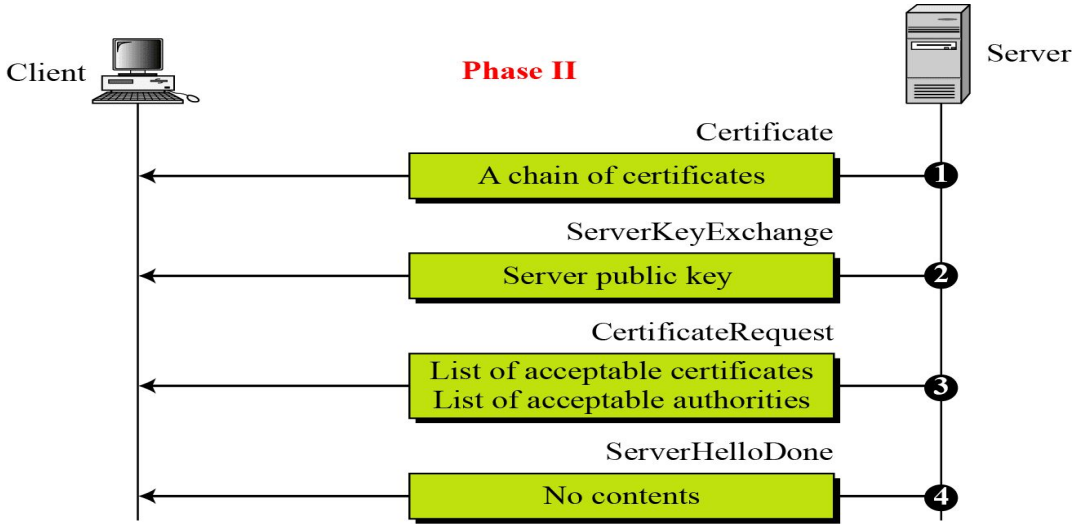
Phase I: Establishing Security Capability

In Phase I, the client and the server announce their security capabilities and choose that are convenient for both. In this phase a session ID is established and the cipher suite is chosen. The paties agree upon a particular compression method. Finally two random numbers are selected, one by the client and one by the server, to be used for creating a master secret.



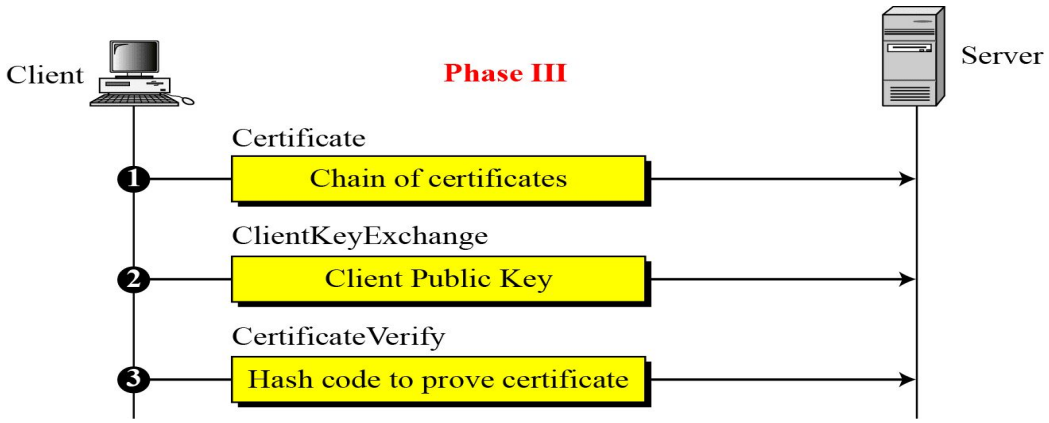
Phase II: Server Key Exchange and Authentication

In this phase the server authenticates itself if needed. The sender may send its certificate, its public key and may also request certificates from the client. At the end, the server announces that the serverHello process is done



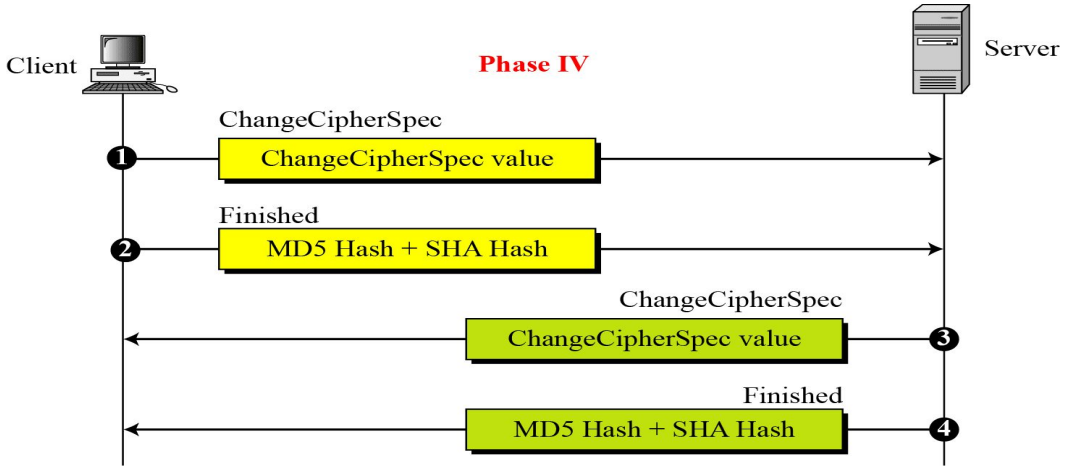
Phase III: Client Key Exchange and Authentication

This phase is designed to authenticate the client. Three messages can be sent from the client to the server.



Phase IV: Finalizing and Finishing

In the phase the client and server send messages to change cipher specification and to finish the handshaking protocol. Four messages are exchanged in this phase.



ChangeCipherSpec: the client sends a changeCipherSpec message to show that it has moved all of the cipher suite ser and the parameters from the pending state to the active state.

Finished: the next message is also sent by the client. It is a Finished message that announces the end of the handshaking protocol by the client.

ChangeCipherSpec: the server sends a ChangeCipherSpec message to show that it has also moved all of the cipher suite set and parameters from the pending state to the active state.

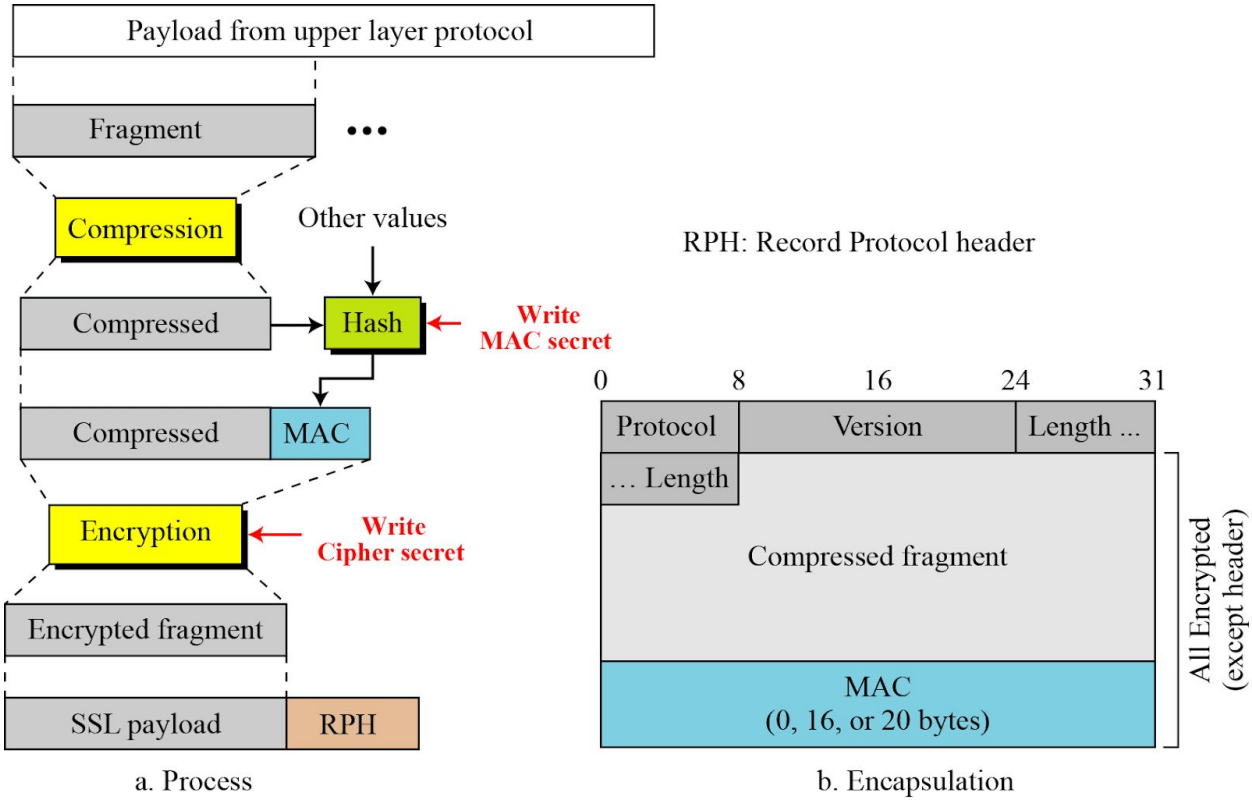
Explain the Record Protocol of SSL

The Record Protocol carries messages from the upper layer (Handshake Protocol. ChangeCipherSpec Protocol, Alert Protocol. or application layer). The message is fragmented and optionally compressed; a MAC is added to the compressed message using the negotiated hash algorithm. The compressed fragment and the MAC are encrypted using the negotiated encryption algorithm. Finally. the SSL header is added to the encrypted message.

This process can only be done when the cryptographic parameters are in the active State. Messages sent before the movement from pending to active are neither signed nor encrypted.

Fragmentation/Combination

At the sender, a message from the application layer is fragmentedof 2^{14} bytes, with the last block possibly less than this size. At the receiver. the fragments are together to make a replica of the original message.



Compression/Decompression

At the sender, all application layer fragments are compressed by the compression method negotiated during the handshaking. The compression method needs to be loss less. The size of the fragment must not exceed 1024 bytes. Some compression methods work only on a predefined block size and if the size of the block is less than this, some padding is added. Therefore. the size of the compressed fragment may be greater than the size of the original fragment. At the receiver, the compressed fragment decompressed to create a replica of the original. If the size of the decompressed fragment exceeds 2^{14}. a fatal decompressoon Alert message is issued.

Signing/Verifying

At the sender, the authentication method defined during the handshake (NULL,MD5, or SHA) creates a signature (MAC). The hash algorithm is applied twice. First, a hash is created from the concatenations of the following values

1.The MAC write secret (authentication key for the outbound message)

b Pad-1. which is the byte 0x36 repeated 48 times for MDS and 40 times for SHA-I

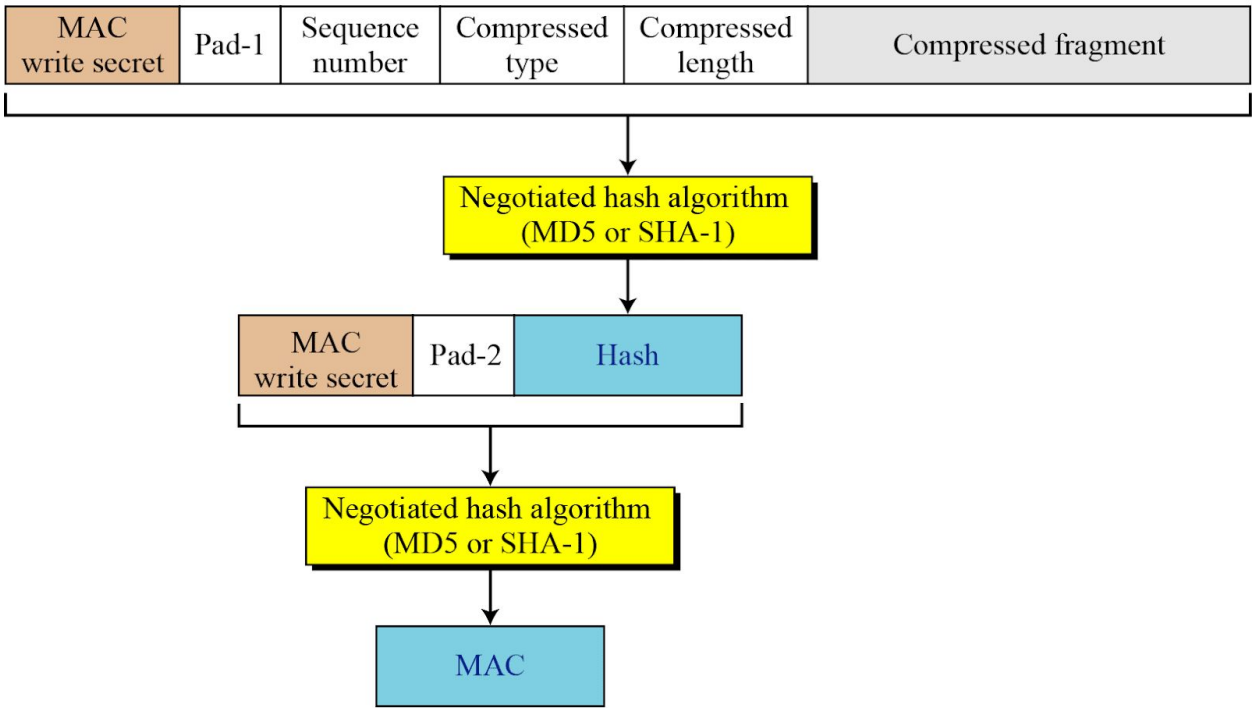
c The sequence number for this message

d the compressed type

d The compressed length, Which is the length of the compressed fragment

e The compressed fragment it self

Pad-1: Byte 0x36 (00110110) repeated 48 times for MD5 and 40 times for SHA-1
Pad-2: Byte 0x5C (01011100) repeated 48 times for MD5 and 40 times for SHA-1



Encryption/Decryption:

At the sender, the compressed fragment and the hash are encrypted using the cipher write secret. At the receiver the received message is decrypted using the cipher read secret.

Framing/Deframing: After the encryption, the record protocol header is added at the sender. The header is removed at the receiver before decryption.

UNIT-VI

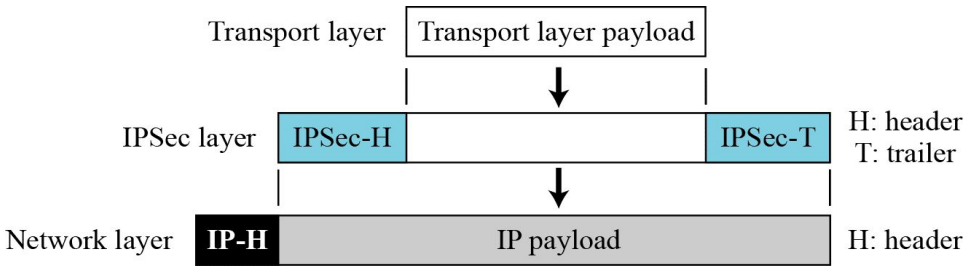
IP Security

IP Security is a collection of protocols designed by the IETF to provide security for a packet at the network level. The network layer in the Internet is referred as IP Layer. IPSec helps to create authenticated and confidential packets for the IP Layer.

Explain the Modes of IPSec

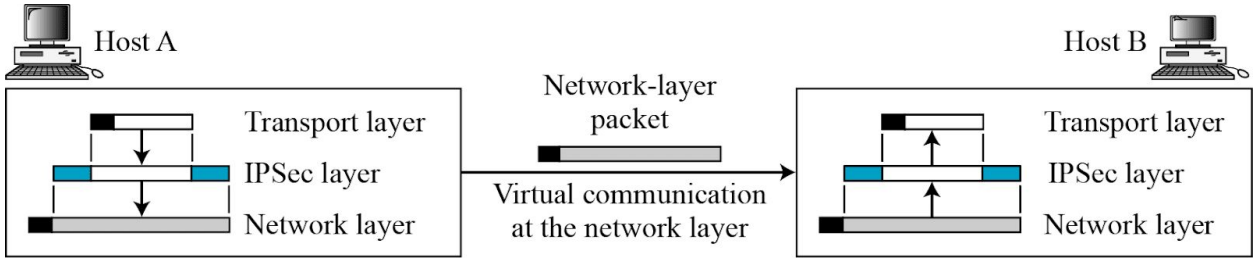
1. Transport Mode

   In transport mode, IPSec protects what is delivered from the transport layer to the network layer. Transport mode protects the network layer payload, the payload to be encapsulated in the network layer



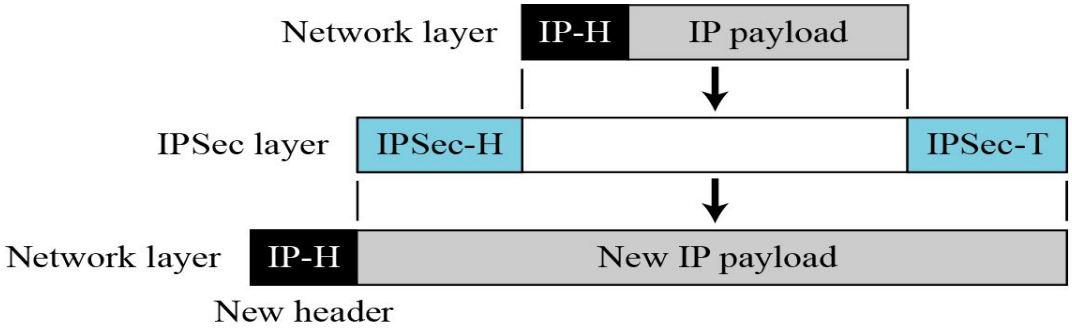
Transport mode does not protect the whole IP Packet, it protects only the packet from the transport layer. In this mode, the IPSEC header are added to the information coming from the transport layer

Transport mode is normally used when we need host to host protection of data. The sending host uses IPSec to authenticate or encrypt the payload delivered from the transport layer. The receiving host uses IPSec to check the authentication and decrypt the IP packet and deliver it to the transport layer.

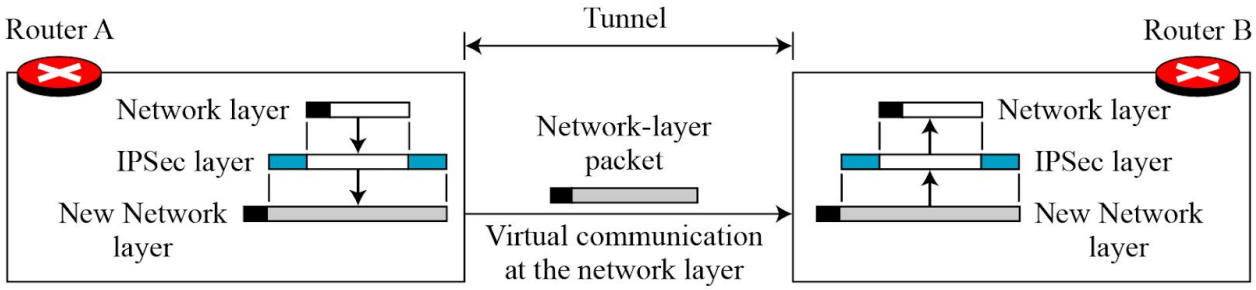


2. Tunnel Mode

In this mode IPSec protects the entire IP Packet. It takes an IP packet, including the header, applies IPSec security methods to the entire packet, and then adds a new IP haeader.

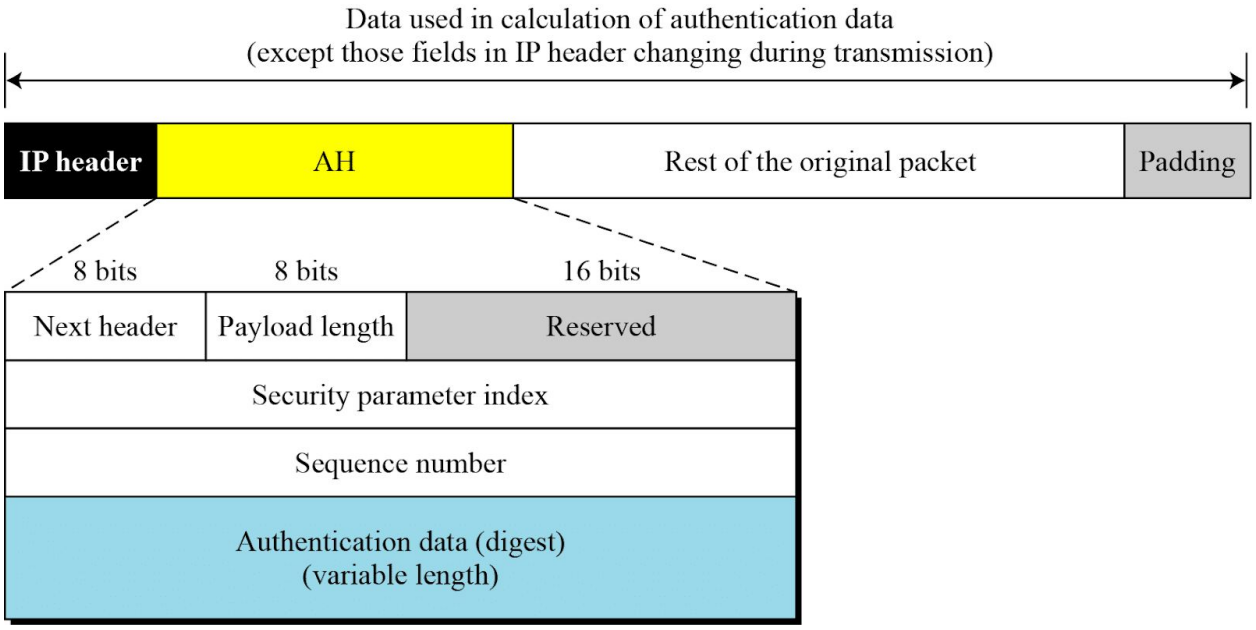


Tunnel mode is normally used between two routers, between a host and a router, or between a router and a host. Tunnel mode is used when either the sender or the receiver is not a host. The entire original packet is protected from intrusion between the sender and the receiver as if the whole packet goes through an imaginary tunnel



Explain Authentication Header and Encapsulating Security payload

Authentication Header (AH)



Next Header (8 bits): Identifies the type of header immediately following this header.

• **Payload Length (8 bits):** Length of Authentication Header in 32-bit words, minus 2. For example, the default length of the authentication data field is 96 bits, or three 32-bit words. With a three-word fixed header, there are a total of six words in the header,and the Payload Length field has a value of 4.

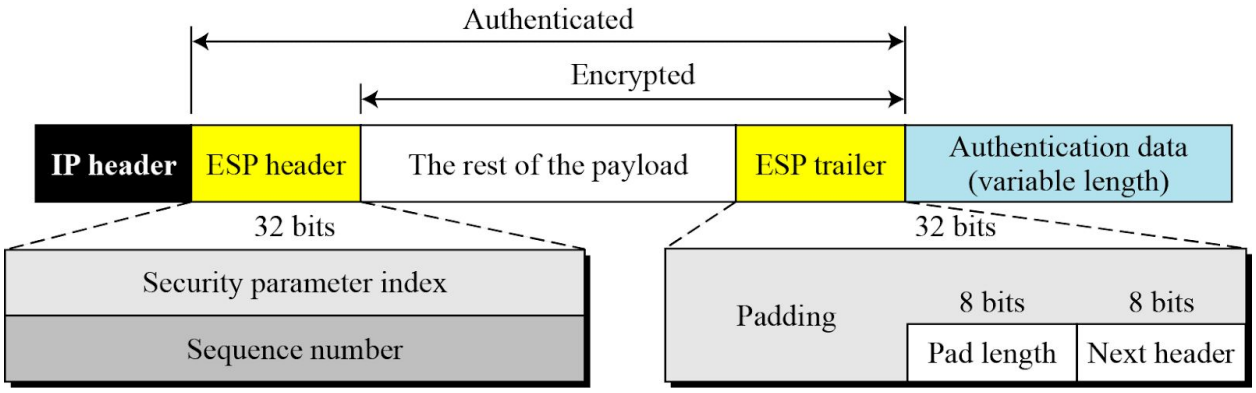
• **Reserved (16 bits):** For future use.

• **Security Parameters Index (32 bits):** Identifies a security association.

• **Sequence Number (32 bits):** A monotonically increasing counter value

• **Authentication Data (variable):** A variable-length field (must be an integral number of 32-bit words) that contains the Integrity Check Value (ICV), or MAC, for this packet.

Encapsulating Security Payload (ESP)



Security Parameters Index (32 bits): Identifies a security association.

• **Sequence Number** (32 bits): A monotonically increasing counter value; this provides an anti-replay function, as discussed for AH.

• **Payload Data (variable):** This is a transport-level segment (transport mode) or IP packet (tunnel mode) that is protected by encryption.

• **Padding (0-255 bytes):** This field is used to make the length of the plaintext to be a multiple of some desired number of bytes. It is also added to provide confidentiality.

• **Pad Length (8 bits):** Indicates the number of pad bytes immediately preceding this field.

• **Next Header (8 bits):** Identifies the type of data contained in the payload data field by identifying the first header in that payload (for example, an extension header in IPv6, or an upper-layer protocol such as TCP).

• **Authentication Data (variable):** A variable-length field (must be an integral number of 32-bit words) that contains the Integrity Check Value computed over the ESP packet minus the Authentication Data field.

Adding encryption makes ESP a bit more complicated because the encapsulation *surrounds* the payload rather than *precedes* it as with AH: ESP includes header and trailer

Explain the Security Policy of IPSec

Security Policy (SP), which defines the type of security applied to a packet when it is to be sent or when it has arrived

Security Policy Data base.

Each host that is using the IPSec protocol needs to keep a Security Policy database(SPD). There is a need for an inbound SPD and an outbound SPD. Each entry in the SPD can be accessed using a sixtuple index: source address, destination address, name, protocol, source port and destination port.

Index	Policy
< SA, DA, Name, P, SPort, DPort >	
< SA, DA, Name, P, SPort, DPort >	
< SA, DA, Name, P, SPort, DPort >	
< SA, DA, Name, P, SPort, DPort >	

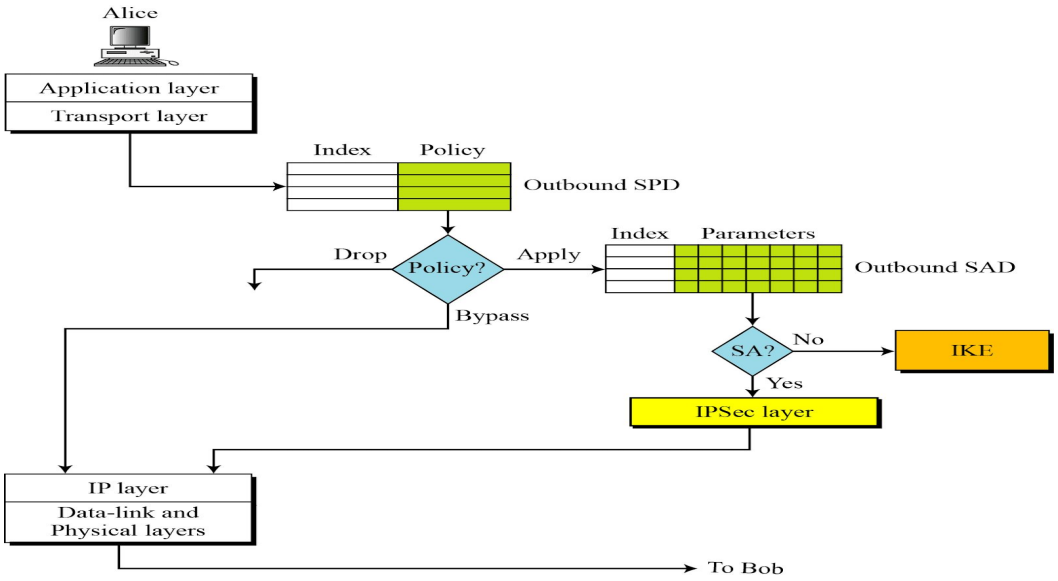
Legend:

SA: Source Address SPort: Source Port
DA: Destination Address DPort: Destination Port
P: Protocol

Source and destination addresses can be unicast, multicast or wildcard addresses. The name usually defines a DNS entity. The protocol is either AH or ESP. the source and destination addresses for the process running at the source and destination hosts.

Outbound SPD

When a packet is to be sent out, the outbound SPD is consulted

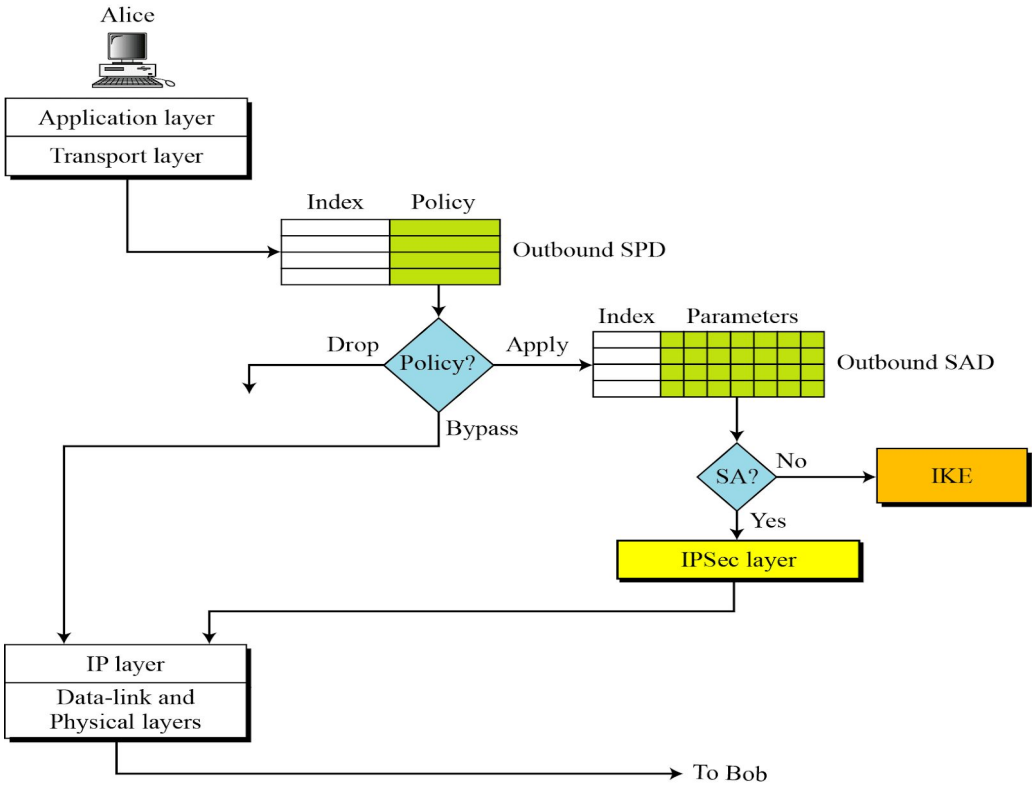


The output is one of the three following cases

1. Discard- this means that the packet defined by that policy must be dropped.
2. Bypass- this means that there is no policy for a packet with this policy index,the packet is sent , bypassing the security header application.
3. Apply- in this case the security header is applied. Two situations may occur.
   a. If an outbound SA is already established, the triple SA index is returned that selects the corresponding SA from the outbound SAD. The AH or ESP header is formed. Encryption, authentication or both are applied based on the SA selected
   b. If an outbound SA is not yet established the IKE is called called to create an outbound and inbound SA for this traffic.

Inbound SPD

When a packet arrives, the inbound SPD is consulted. Each entry in the inbound SPD is also accessed using the sam sixtuple index.



The output is one of the three following cases

1. Discard- this means that the packet defined by that policy must be dropped.

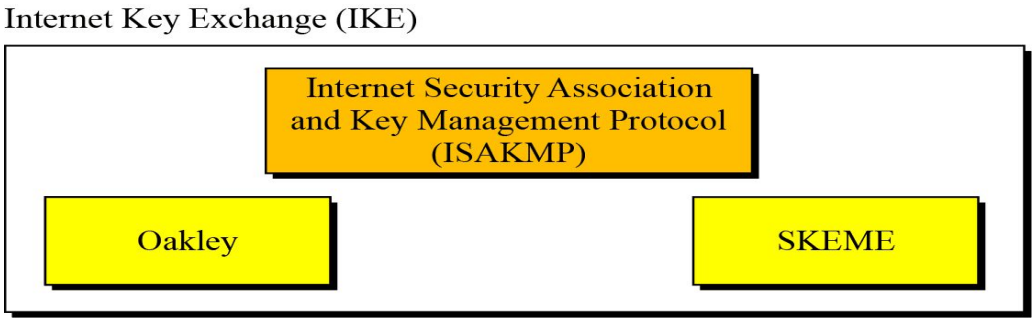
2. Bypass- this means that there is no policy for a packet with this policy index,the packet is processed, ignoring the information form AH or ESP header. The packet is delivered to the transport layer.
3. Apply- in this case the security header must be processed. Two situations may occur.
   c. If an inbound SA is already established, the triple SA index is returned that selects the inbound SA from the inbound SAD. Decryption, authentication or both are applied. If the packet passes the security criteria, the AH or ESP header is discarded and the packet is delivered to transport layer
   d. If an SA is not yet established the packet may be discarded.

INTERNET KEY EXCHANGE (IKE)

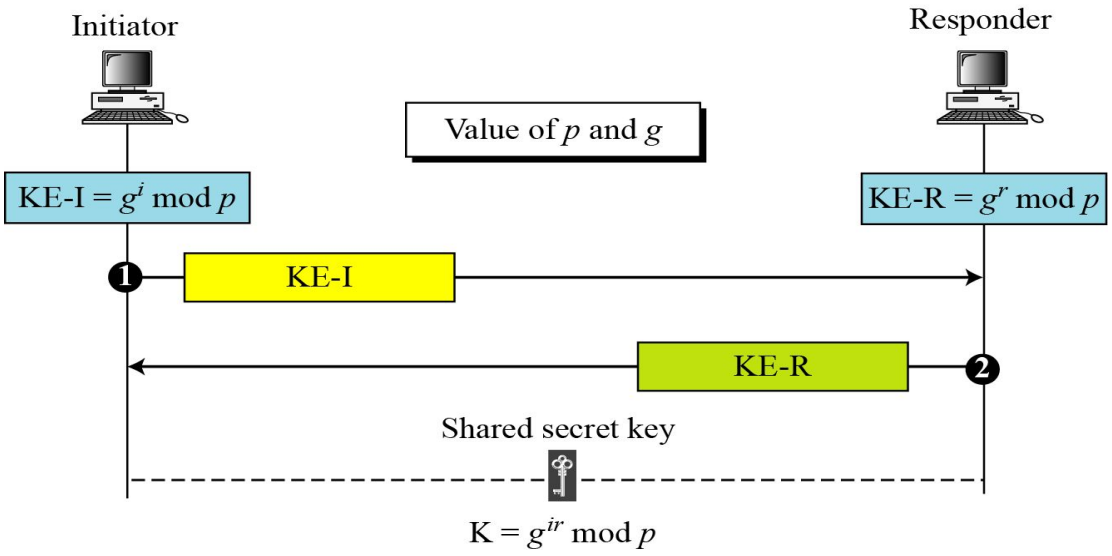
The Internet Key Exchange (IKE) is a protocol designed to create both inbound and outbound Security Associations.

IKE creates SAs for IPSec.

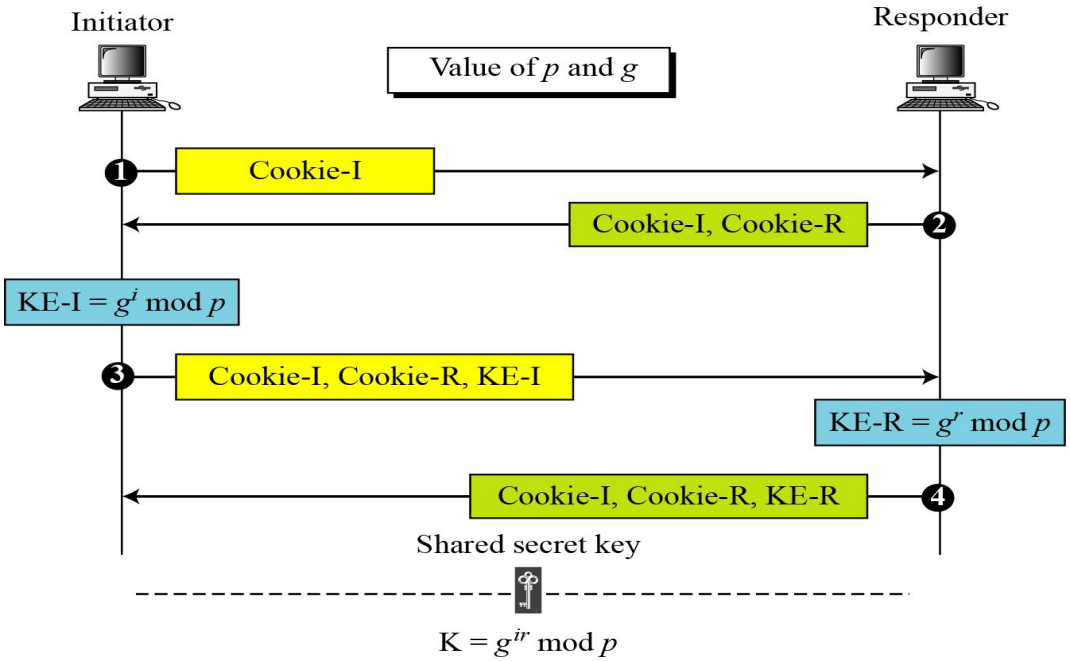
IKE components



Improved Deffie Hellman Key Exchange

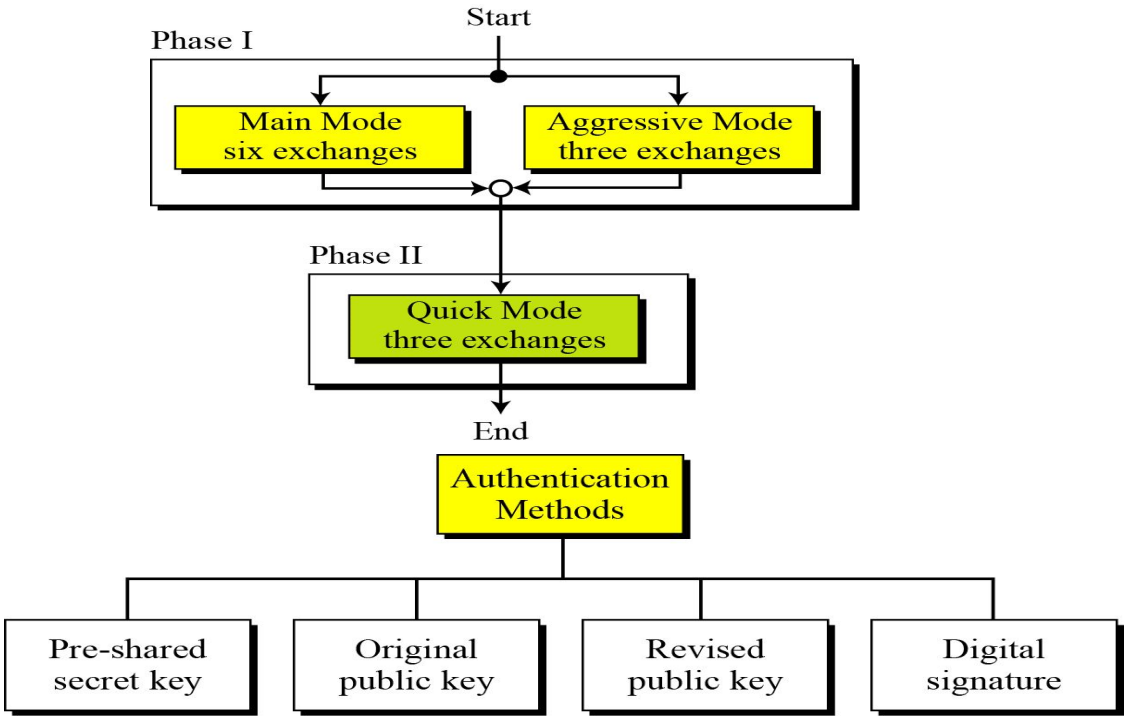


Clogging Attack



Explain the Phases in IKE

IKE is divided into two phases: phase I and phase II. Phase I creates SAs for phase II; phase II creates SAs for a data exchange protocol such as IPSec..



Phase 1: Main Mode

In main mode, the initiator and respond exchange six messages. In the first two messages, they exchange cookies and negotiate the SA parameters. The initiator sends a series of proposals; the responder selects one of them. When the first two messages are exchanged, the initiator and the responder know the SA parameters and are confident that the other party exists

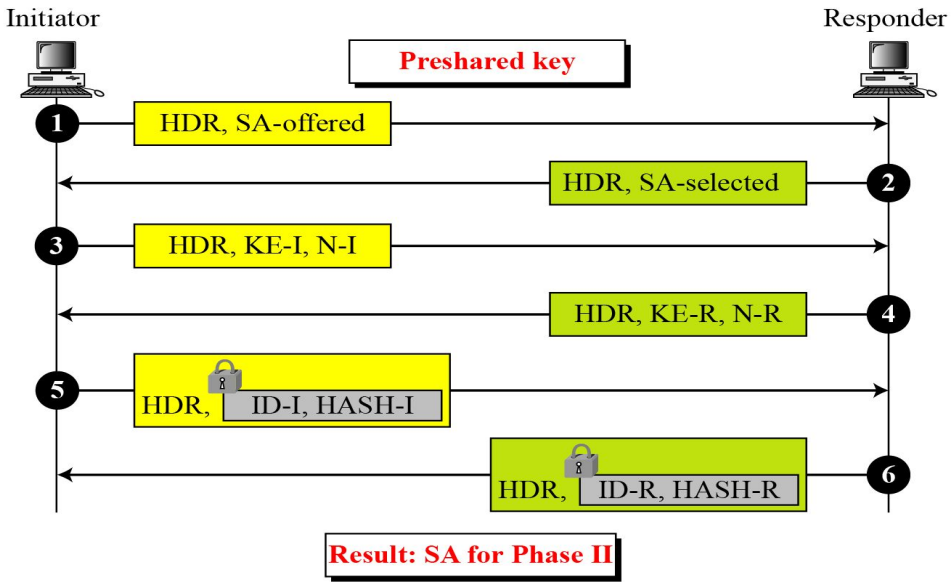
In the third and fourth messages, the initiator and responder usually exchange their half keys and their nonces. After exchanging each party calculate the common secret between them in addition to its individual hash digest.

Preshared secret key method

In this method a secret key is used for authentication of the peers to each other.

KE-I (KE-R): Initiator's (responder's) half-key
N-I (N-R): Initiator's (responder's) nonce
ID-I (ID-R): Initiator's (responder's) ID
HASH-I (HASH-R): Initiator's (responder's) hash

HDR: General header including cookies
Encrypted with SKEYID_e

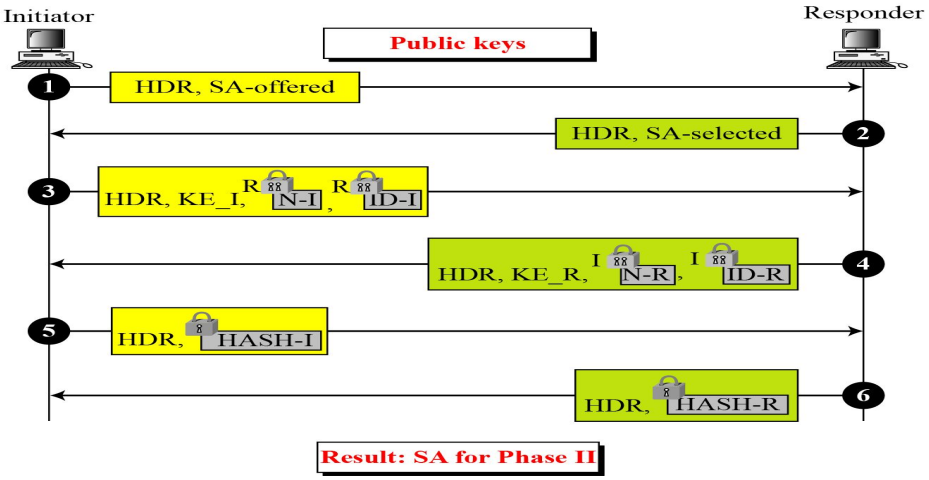


Original Public Key Method:

In this method the initiator and the responder prove their identities by showing that they possess a private key related to their announced public key

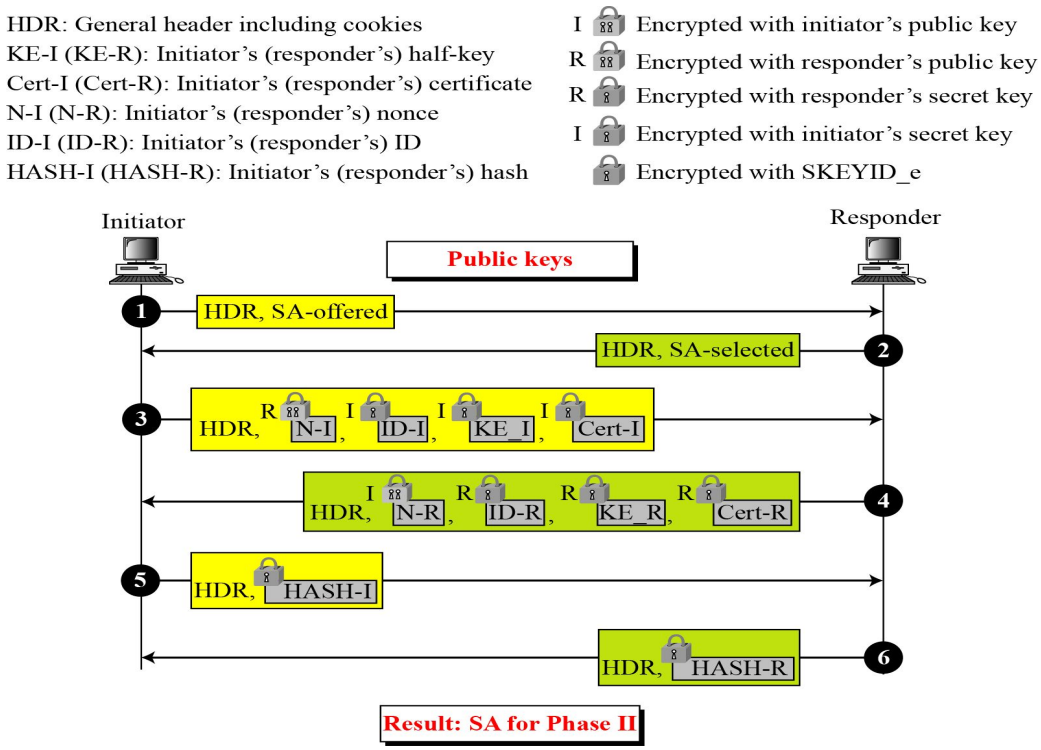
HDR: General header including cookies
KE-I (KE-R): Initiator's (responder's) half-key
N-I (N-R): Initiator's (responder's) nonce
ID-I (ID-R): Initiator's (responder's) ID
HASH-I (HASH-R): Initiator's (responder's) hash

I Encrypted with initiator's public key
R Encrypted with responder's public key
Encrypted with SKEYID_e



Revised Public key method

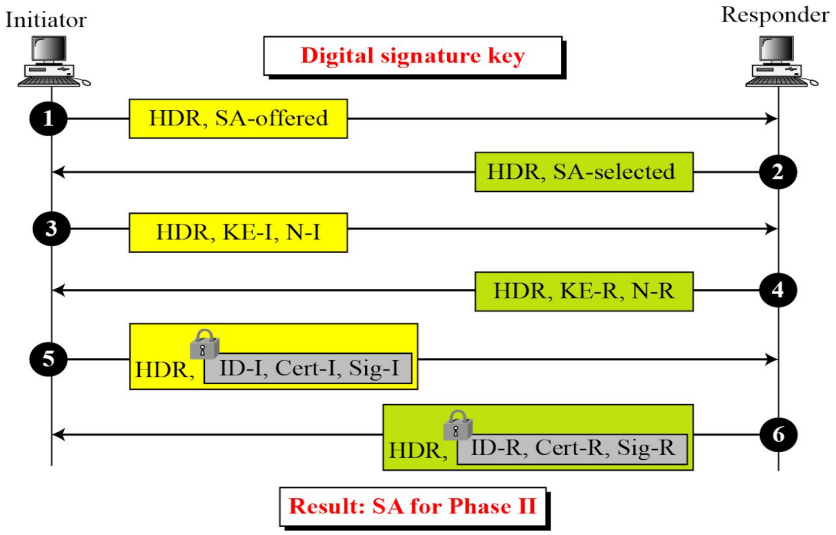
The original pubic key method has some drawbacks. First, two instances of public key encryption/decryption place a heavy load on the initiator and responder. Second the initiator cannot send its certificate encrypted by the public key of the responder. The method is revised so that the public key is used only to create temporary secret key.



Digital Signature method:

In this method each party shows that each party possesses the certified private key related to the digital signature.

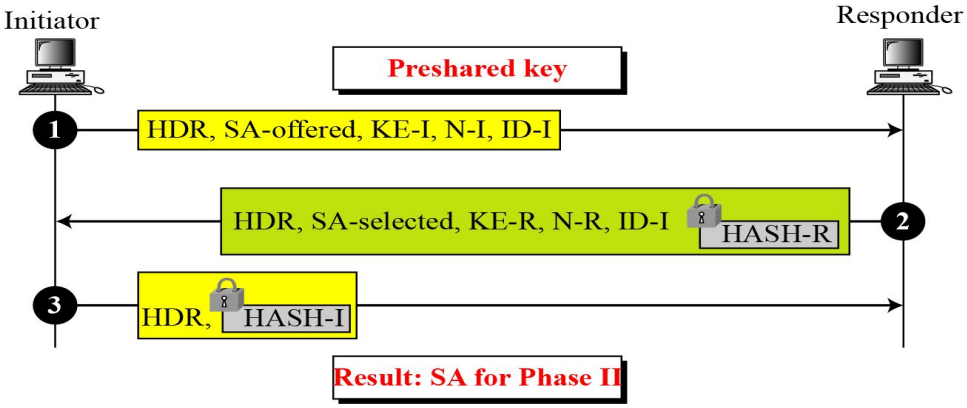
HDR: General header including cookies
Sig-I: Initiator's signature on messages 1-4
Sig-R: Initiator's signature on messages 1-5
Cert-I (Cert-R): Initiator's (responder's) certificate
N-I (N-R): Initiator's (responder's) nonce
KE-I (KE-R): Initiator's (responder's) half-key
ID-I (ID-R): Initiator's (responder's) ID
Encrypted with SKEYID_e



Phase I: Aggressive mode

Pre Shared Key Method

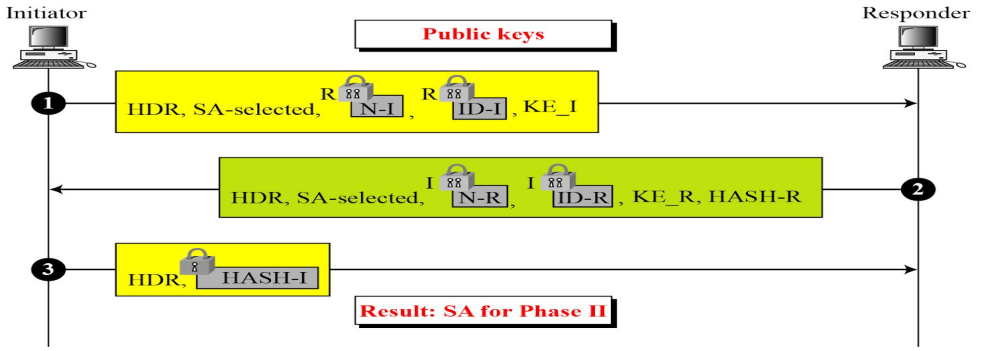
KE-I (IK-R): Initiator's (responder's) half-key
N-I (N-R): Initiator's (responder's) nonce
HASH-I (HASH-R): Initiator's (responder's) hash
HDR: General header including cookies
Encrypted with SKEYID_e
ID-I (ID-R): Initiator's (responder's) ID



Original Public Key Method

HDR: General header including cookies
KE-I (KE-R): Initiator's (responder's) half-key
N-I (N-R): Initiator's (responder's) nonce
ID-I (ID-R): Initiator's (responder's) ID

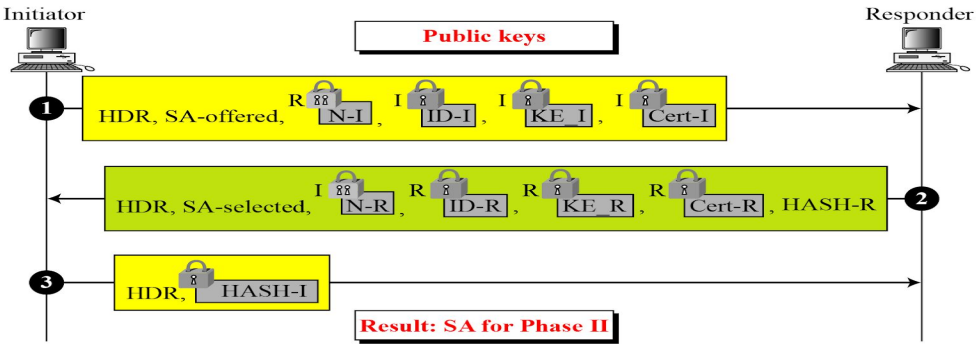
I Encrypted with initiator's public key
R Encrypted with responder's public key
Encrypted with SKEYID_e
HASH-I (HASH-R): Initiator's (responder's) hash



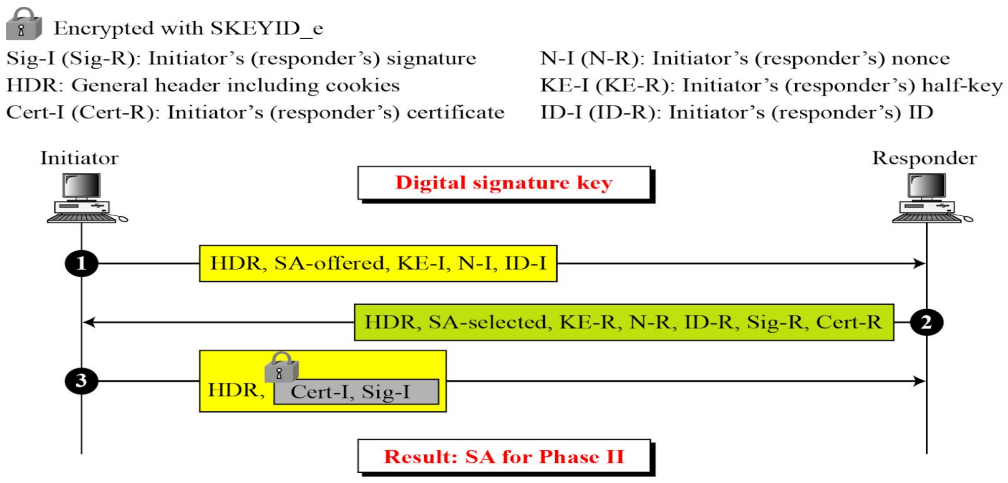
Revised Public Key Method

HDR: General header including cookies
KE-I (KE-R): Initiator's (responder's) half-key
Cert-I (Cert-R): Initiator's (responder's) certificate
N-I (N-R): Initiator's (responder's) nonce
ID-I (ID-R): Initiator's (responder's) ID
HASH-I (HASH-R): Initiator's (responder's) hash

I Encrypted with initiator's public key
R Encrypted with responder's public key
R Encrypted with responder's secret key
I Encrypted with initiator's secret key
Encrypted with SKEYID_e

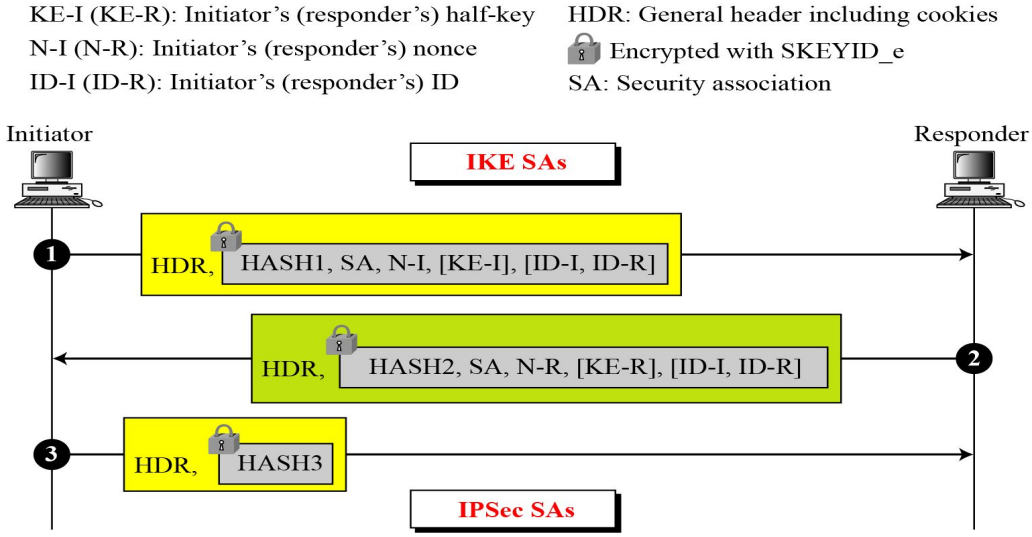


Digital Signature method



Phase II: Quick mode

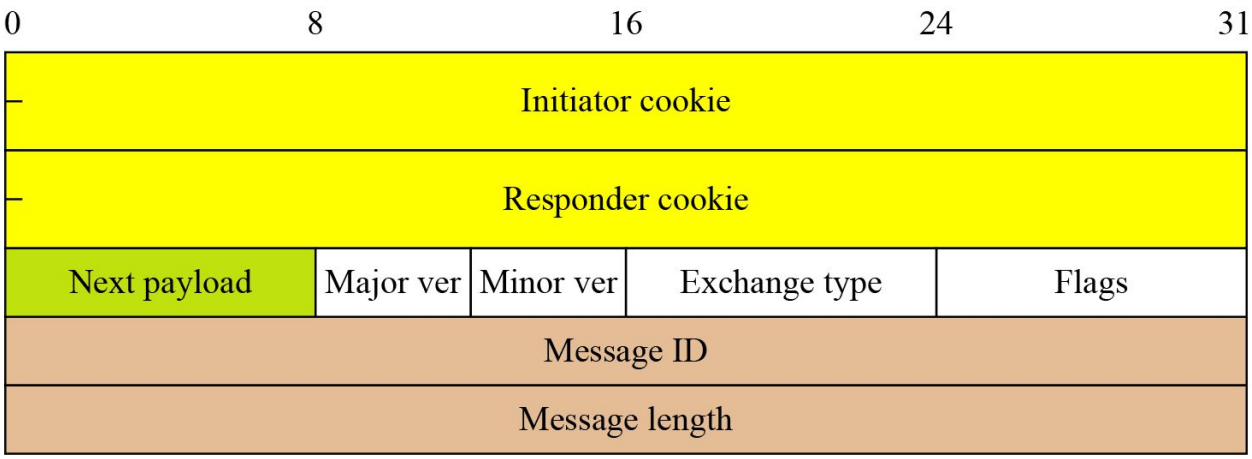
After As have been created in either the main mode or the aggressive mode, phase II can be started. There is only one mode defined for phase II that is quick mode.



ISAKMP

The ISAKMP IS designed to carry messages for the IKE Exchange

The general header is



Initiator Cookie:Tthis 32 bit field defines the cookie of the entity that initiates the SA Establishment

Responder cookie: This 32-bit field defines the cookie of the responding entity. The value of this field is 0 when the initiator sends the first message,

Next payload: This 8-bit field defines the type of payload that immediately follows the header.

Major version: This 4-bit version defines the major version of the protocol. Currently, the value of this field is 1.

Minor version: This 4-bit version defines the minor version of the protocol. Currently. the value of this field is 0.

Exchange type: This 8-bit field defines the type of exchange that is being carried by the ISAKMP packets.

Flags: This is an 8-bit field in which each bit defines an option for the exchange. So far only the three least significant bits are defined. The encryption bit, when set to l, specifies that the rest of the payload will be encrypted using the encryption key and the algorithm defined by SA. The commitment bit, when set to l, specifies that encryption material is not received before the establishment of the SA. The authentication bit, when set to l, specifies that the rest of the payload, though not encrypted, is authenticated for integrity.

Message ID: This 32-bitfield is the unique message identity that defines the protocol state. This

field is used only during the second phase of negotiation and is set to 0 during the first phase,

Message length: Because different payloads can be added to each packet, the length of a message can be different for each packet. This 32-bit field defines the length of the total message, including the header and all payloads.

Payloads

Types	*Name*	*Brief Description*
0	None	Used to show the end of the payloads
1	SA	Used for starting the negotiation
2	Proposal	Contains information used during SA negotiation
3	Transform	Defines a security transform to create a secure channel
4	Key Exchange	Carries data used for generating keys
5	Identification	Carries the identification of communication peers
6	Certification	Carries a public-key certificate
7	Certification Request	Used to request a certificate from the other party
8	Hash	Carries data generated by a hash function
9	Signature	Carries data generated by a signature function
10	Nonce	Carries randomly generated data as a nonce
11	Notification	Carries error message or status associated with an SA
12	Delete	Carries one more SA that the sender has deleted
13	Vendor	Defines vendor-specification extensions