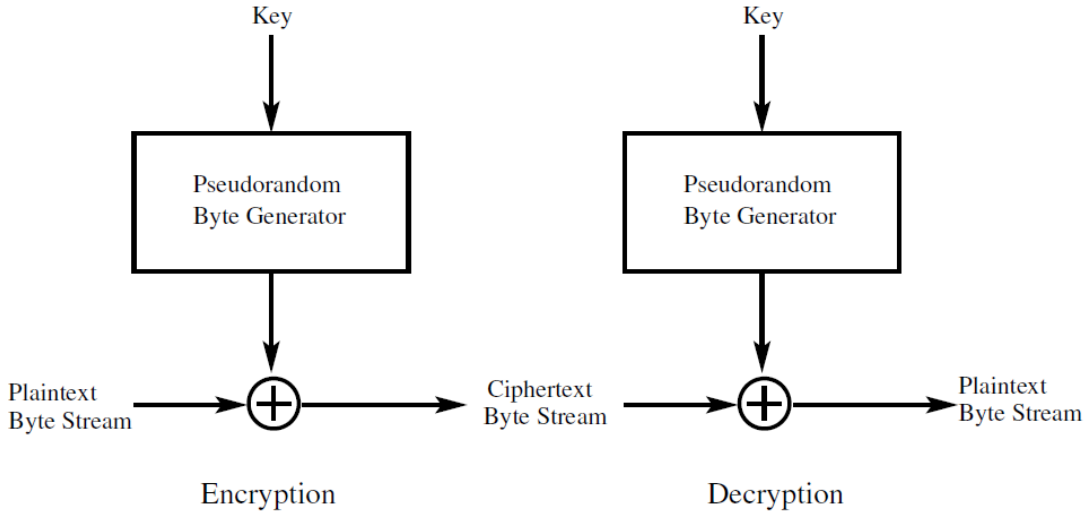
■STREAM CIPHERS: a typical stream cipher encrypts plaintext one byte at a time.

1. A simple stream cipher: as each byte of the plaintext becomes available, you XOR it with a byte of the pseudorandom byte stream. The output byte is what is transmitted to the destination.



2. The main processing step in a true stream cipher is the generation of a stream of pseudorandom bytes that depend on the encryption key.
3. As a new byte of plaintext shows up for encryption, a new byte of the pseudorandom stream also becomes available at the same time and this happens on a continuous basis.
4. Every pseudorandom number generator produces a seemingly random sequence that eventually repeats. The longer the period, the more difficult it is to break the cipher.
5. a stream cipher is particularly appropriate for audio and video streaming. A stream cipher is also frequently used for browser web-server links. A block cipher, on the other hand, is more appropriate for file transfer

■RC4 is a variable key length stream cipher with byte-oriented operations.

1. 256 element array of 8-bit integers. It is called the state vector and denoted S. State vector is initialized with the encryption key.
   I. state vector S is initialized with entries from 0 to 255 in ascending order.
   II. another temporary 256-element vector T which is initialized by placing in it as many repetitions of the key as necessary until T is full.
   III. Key Scheduling Algorithm (KSA): use the 256-element vector T to produce the initial permutation of S.

$$
\begin{aligned}
S[0] &= 0x00 = 0 \\
S[1] &= 0x01 = 1 \\
S[2] &= 0x02 = 2 \\
S[3] &= 0x03 = 3 \\
&\cdots\cdots \\
&\cdots\cdots \\
S[255] &= 0xFF = 255
\end{aligned}
$$

```
j = 0
for i = 0 to 255
    j = ( j + S[i] + T[i] ) mod 256
    SWAP S[i], S[j]
```

IV. There is no further use for the temporary vector T and encryption key after the state vector S is initialized as described above.

V. initialization procedure for the state S is just a permutation of the integers from 0 through 255. Each integer in this range will be in one of the elements of S after initialization.

2. pseudorandom byte stream is generated from the state vector.

```
i, j = 0
while ( true )
    i = ( i + 1 ) mod 256
    j = ( j + S[i] ) mod 256
    SWAP S[i], S[j]
    k = ( S[i] + S[j] ) mod 256
    output S[k]
```

I. the state of the pseudorandom number generator changes dynamically as the the numbers are being generated.

II. The above procedure spits out S[k] for the pseudorandom byte stream. The plaintext byte is XORed with this byte to produce an encrypted byte.

III. pseudorandom sequence of bytes generated by the above algorithm is also known as the keystream.

3. Because all operations are at the byte level, the cipher possesses fast software implementation. For that reason, RC4 was the software stream cipher of choice for several years. More recently though, RC4 was shown to be vulnerable to attacks especially if the beginning portion of the output pseudorandom byte stream is not discarded. For that reason, the use of RC4 in the SSL/TLS protocol is now prohibited.

4. WiFi security started with RC4 in the WEP protocol. After it was discovered that the encryption key used in WEP could be acquired by an adversary in almost no time, WiFi security has now moved on to the WPA2 protocol that uses AES for encryption.

5. The WEP protocol requires each packet to be encrypted separately with its own RC4 key. If the same keystream S is used for two different plaintext byte streams P1 and P2, an XOR of the corresponding ciphertext streams becomes independent of the keystream because

$$C_1 \oplus C_2 \quad = \quad (P_1 \oplus S) \oplus (P_2 \oplus S) \quad = \quad P_1 \oplus P_2$$

This can create a backdoor to extracting the plaintext stream from the ciphertext

stream. All you have to do is to XOR the ciphertext in each packet with the ciphertext stream in a packet in which a reasonably large number of bytes are set to 0.

6. The RC4 key for each packet is a simple concatenation of a 24-bit Initialization Vector (IV) and the root key (APs security code). While the root key remains fixed over all the packets, you increment the value of IV from one packet to the next. Since the IV is sent in plaintext, anyone with a packet sniffer can directly see the first three bytes of the RC4 key used for a packet.
7. WEP then computes the CRC32 checksum of the data to be encrypted in the packet. CRC (Cyclic Redundancy Check) is a parity check to guard against data corruption during transmission. CRC32 gives us a 32-bit checksum. In WEP, this CRC32 signature is called Integrity Check Value (ICV). Finding CRC32 of a binary data stream amounts to dividing the data bit pattern (which could be the bits in an entire file) by a polynomial of degree 32. The bit pattern corresponding to the residue would therefore only be 32 bits long.
8. The RC4 key for a packet is then used to encrypt the data followed by ICV value
9. the root key remains fixed for long periods of time (in home use, people almost never change) and the IV has only 24 bits. This implies that distinct keystreams can be generated for only 2^24 different packets and the same keystream will be used for different packets in a long session.
10. TKIP for Temporal Key Integrity Protocol:
    I. WPA uses a 48-bit Initialization Vector to enhance security
    II. WPA uses a Message Integrity Check (MIC) for message authentication at the receiving endpoint to protect the packets against tampering caused by an adversary who had successfully broken the WEP encryption and who changed both the packet payload and its ICV value. MIC is an integrity check on both the packet header and the payload. MIC adds a sequence number field to the wireless frames. This allows the receiving endpoint to simply discard a frame that is received out of sequence. MIC consists of an 8-byte value that is placed between the data payload and the 4-byte ICV in an IEEE 802.11 frame. The MIC field is encrypted together with the payload and the ICV. All of these enhancements in WPA over WEP are a part of the
    III. Additional security services that determination of the unique starting encryption key for each user authentication (through, say, PSK); and synchronized changing of the encryption keys from packet to packet
11. TKIP is slightly-more-secure wrapper around WEP. With regard to the security of its encryption, TKIP suffers from the basic RC4-based weaknesses as WEP.

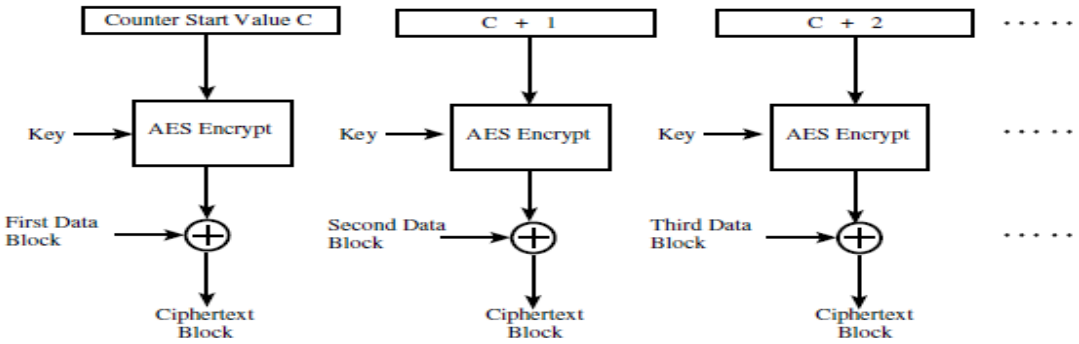
■WiFi communications are encrypted with WEP, WPA, and WPA2 protocols. RC4 is used for packet-based data encryption in both WEP and WPA. WPA2 uses the AES

block cipher

1. In addition to data encryption, the WiFi protocols also provide user authentication services. These services determine how a client (a laptop, smartphone, etc) would be allowed to join the WLAN.
2. All three WiFi security protocols allow for authentication to be carried out with a Pre-Shared Key (PSK). A PSK can be as simple as 10 manually specified hex digits for the case of WEP or, for the case of WPA and WPA2, derived with a key derivation function from a shared secret passphrase.
3. When a shared secret is used for client authentication, the WPA andWPA2 protocols are also referred to asWPA-PSK andWPA2-PSK.
4. WPA2-PSK / WPA2-Personal: it is meant for be used for SOHO (small office and home) applications where one may assume that it is safe to have a shared secret passphrase for the clients to connect with the WLAN.
5. WPA2-Enterprise: WPA2 can also be used in a more secure enterprise mode, each user in WPA2-Enterprise has a separate secret for connecting with the WLAN.
6. WPA2-Enterprise are based on the IEEE 802.1x standard. This standard involves three agents:
    I. a supplicant (a client) that wishes to join a WLAN
    II. an authenticator: an AP
    III. an authentication server: typically is based on the EAP (Extensible Authentication Protocol) for verifying the login credentials supplied by the supplicant to the authenticator. Extensible Authentication Protocol.
7. WPA2-PSK protocol is vulnerable to KRACK Key Reinstallation AttaCK: it causes of the vulnerability was NOT a bug in an implementation of the protocol, but in the WiFi standard itself. In the 4-way handshake that is used in WPA2-PSK to establish a randomly generated key for AES based encryption of the communications between the WiFi access point and a client digital device.
8. WPA2 uses CBC-MAC, Cipher Block Chaining mode-Message Authentication Code: generates a MAC value that the receiver can use to verify the data integrity of a received packet.
9. CCMP / Counter Mode Cipher Block Chaining Protocol: CTR mode of using AES for encryption and the CBC-MAC based message integrity checking, using a single encryption key for encryption and cryptographic message integrity checking
10. one of the main features of WPA2 is that it separates user authentication services from the services needed for encryption and message integrity. This allows WPA2 to be used for SOHO applications with a single shared passphrase, and in large enterprises applications where it is necessary to enforce per-user authentication with separate logon or certificate based credentials for each user. When WPA2 is

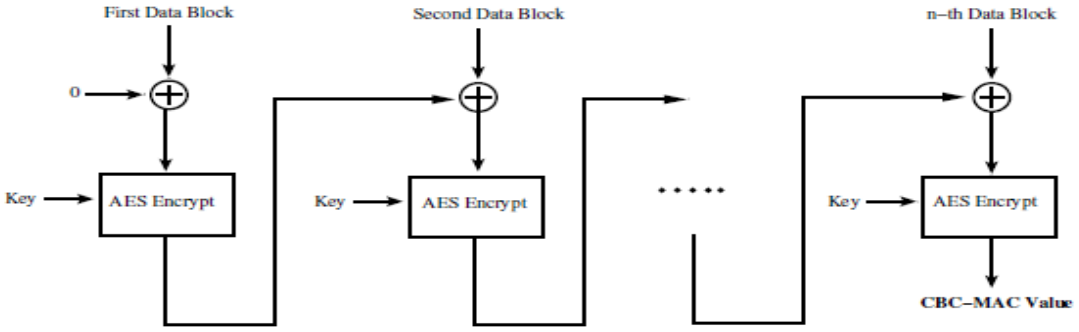
used with a single shared passphrase for WiFi access, it is referred to as WPA2-PSK. When WPA2 is used with per-user authentication, it is referred to as WPA2-Enterprise

11. For backward compatibility, WPA2 allows itself to be used with the WPAs RC4 based TKIP protocol.
12. CTR Mode for AES Encryption in WPA2.



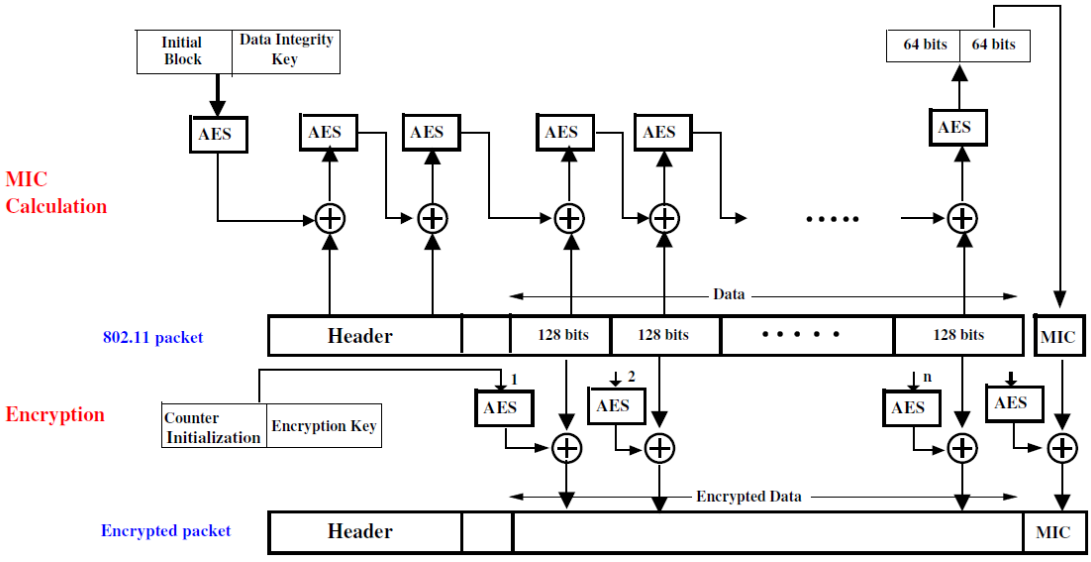
CTR mode for WPA2 Encryption with AES

13. Calculation of CBC-MAC in WPA2.



Using AES for Calculating the CBC–MAC Value of Data

14. CMP Protocol for WPA2 Encryption and for Calculating MIC for Data Integrity



■Some Highly Successful Attacks on WEP

1. Klein Attack for figuring out the WEP root key. This attack is based on Andreas Kleins combinatorial analysis of the pseudorandom sequence produced by the RC4 algorithm.

Algorithm 1 Algorithm 1: RC4 Key Scheduling

```
{initialization}
for i = 0 to n − 1 do
        S[i] ← i
end for
j ← 0
{generate a random permutation}
for i from 0 to n − 1 do
        j ← (j + S[i] + K[i mod length(K)]) mod n
        Swap S[i] and S[j]
end for
```

Algorithm 2 Algorithm 1: RC4 Pseudorandom Generator

```
{initialization}
i ← 0
j ← 0
{generate pseudorandom sequence}
loop
        i ← (i + 1) mod n
        j ← (j + S[i]) mod n
        Swap S[i] and S[j]
        k ← (S[i] + S[j]) mod n
        output S[k]
end loop
```

I. strong correlations exist in byte sequence produced by the pseudorandom byte generation algorithm. These correlations are expressed in the form of probabilities of the output pseudorandom sequence satisfying certain constraints vis-a-vis the the values of the state vector S.

II. attack proposed by Klein is a plaintext-ciphertext attack. an easy way to collect the needed plaintext-ciphertext pairs is for the attackers wireless interface to send repeated ARP requests to the wireless AP being attacked. Even though the attacker will only see the ciphertext for the encrypted portion of these 802.11 frames, he/she can make good guesses for the fields that come before the Data field. These plaintext bytes can be XORed with the ciphertext bytes to recover several initial bytes of the pseudorandom sequence that was generated by the RC4 algorithm.

III. There are two main theoretical results derived by Klein that play a critical role in the attack.

   i. For an i for a given output byte, the probability of the output byte plus the state vector byte S[j] being equal to i mod n is 2/n. For the first output pseudorandom byte, we can say that Prob(S[j] + S[k]) = 1 is 2/256 where S[k] is the value of the byte that is output and S[j] state vector byte that goes into the calculation of the output byte.

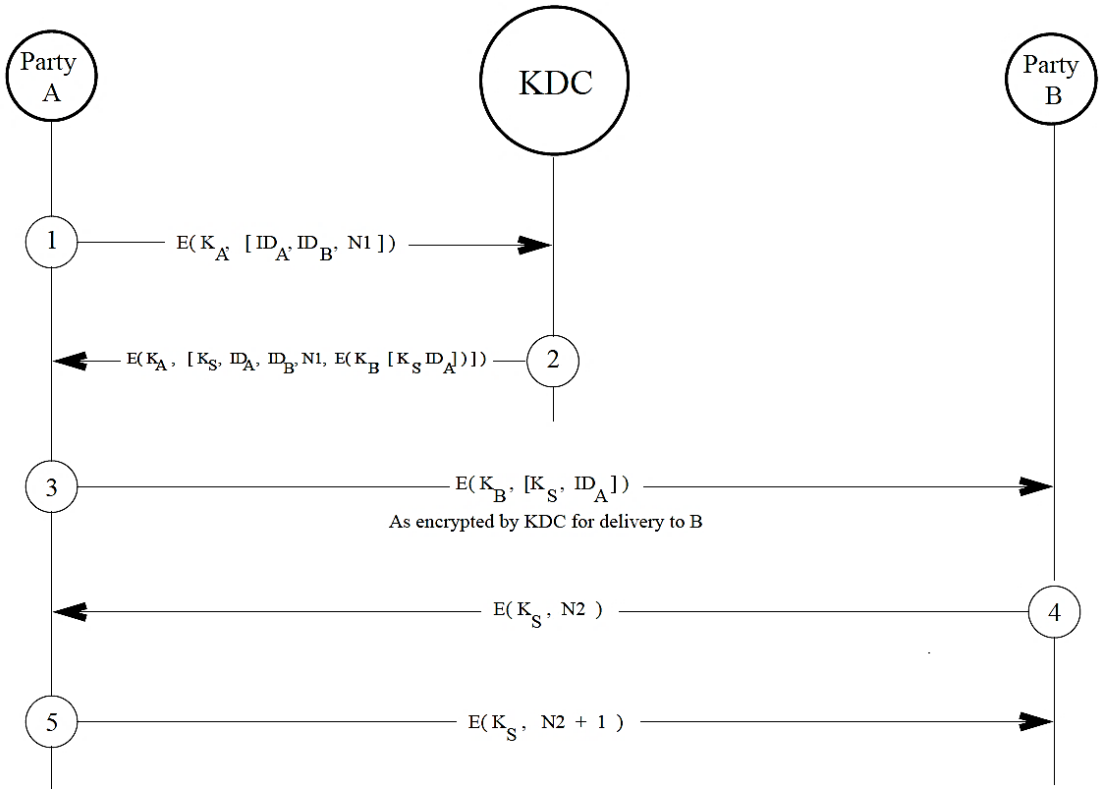
$$Prob\,(S[j] + S[k] \equiv i \bmod n) = \frac{2}{n}$$

$$Prob\,(S[j] + S[k] \equiv c \bmod n) = \frac{n-2}{n(n-1)}$$

   ii.

IV. The basic form of the attack consists of assuming that you know K[0] and you can guess a value for K[1] that will be the correct value with a high probability.

■THE NEED FOR KEY DISTRIBUTION CENTERS due to large groups of people/processes/systems, especially when group membership is ever changing.



1. Needham-Schroder protocol
   - I. party A wants to establish a secure communication link with another party B. Both the parties A and B possess master key KA and KB, respectively, for communicating privately with a key distribution center (KDC).
   - II. Using the key KA for encryption, user A sends a request to KDC for a session key intended specifically for communicating with user B.
   - III. The message sent by A to KDC includes A's network address (IDA),B's network address (IDB), and a unique session identifier nonce (number used once, a random number)
   - IV. KDC responds to A with a message encrypted using the key KA:
     - i. session-key KS that A can use for communicating with B.
     - ii. original message received from A, including the nonce to allow A to match the response received from KDC with the request sent. A may be trying to establish multiple simultaneous sessions with B.
     - iii. ticket that A receives for sending to B: A packet of information meant for A to be sent to B encrypted using B's master key (A cannot look inside the packet)
       1. the session key
       2. A's identifier

V. Using the master key KA, A decrypts the message received from KDC. Because only A and KDC have access to the master key KA, A is certain that the message received is indeed from KDC.
VI. A keeps the session key KS and sends the packet intended for B to B. This message is sent to B unencrypted by A. But it was previously encrypted by KDC using Bs master key. Therefore, this first contact from A to B is protected from eavesdropping.
VII. B decrypts the message received from A using the master key KB. B compares the IDA in the decrypted message with the sender identifier associated with the message received from A. By matching the two, B makes certain that no one is masquerading as A.
VIII. Using the session key KS, B sends back to A a nonce N2. A responds back with N2 + 1, using the same session key KS.
    i. This serves as a confirmation that the session key KS works for the ongoing session this requires that what A encrypts with KS be different from what B encrypted with KS.
    ii. This part of the handshake also ensures that B knows that it did not receive a first contact from A that A is no longer interested in.
    iii. It provides some protection against a replay attack. (EX. if A was allowed to send back to B the same nonce that it received from the latter, then B could suspect that some other party C posing as A was merely replaying back Bs message that it had obtained by, say, eavesdropping.)

2. One can think of KDCs organized hierarchically, with each local network serviced by its own KDC, and a group of networks serviced by a more global KDC, and so on. A local KDC would distribute the session keys for secure communications between users/processes/systems in the local network. But when a user/process/system desires a secure communication link with another user/process/system in another network, the local KDC would communicate with a higher level KDC and request a session key for the desired communication link. Such a hierarchy of KDCs simplifies the distribution of master keys. A KDC hierarchy also limits the damage caused by a faulty or subverted KDC.

■Kerberos protocol provides security for client-server interactions in a network.

1. The main difference between the Needham-Schroeder protocol and the Kerberos protocol is that the latter makes a distinction between the clients, on the one hand, and the service providers, on the other.
2. In Kerberos protocol, the Key Distribution Center (KDC) is divided into two parts
    I. Authentication Server (AS): devoted to client authentication. A client must

first authtenticate himself/herself/itself to AS and obtain from AS a session key for accessing TGS.

II. Ticket Granting Server (TGS):in charge of providing security to the service providers.

III. Keys

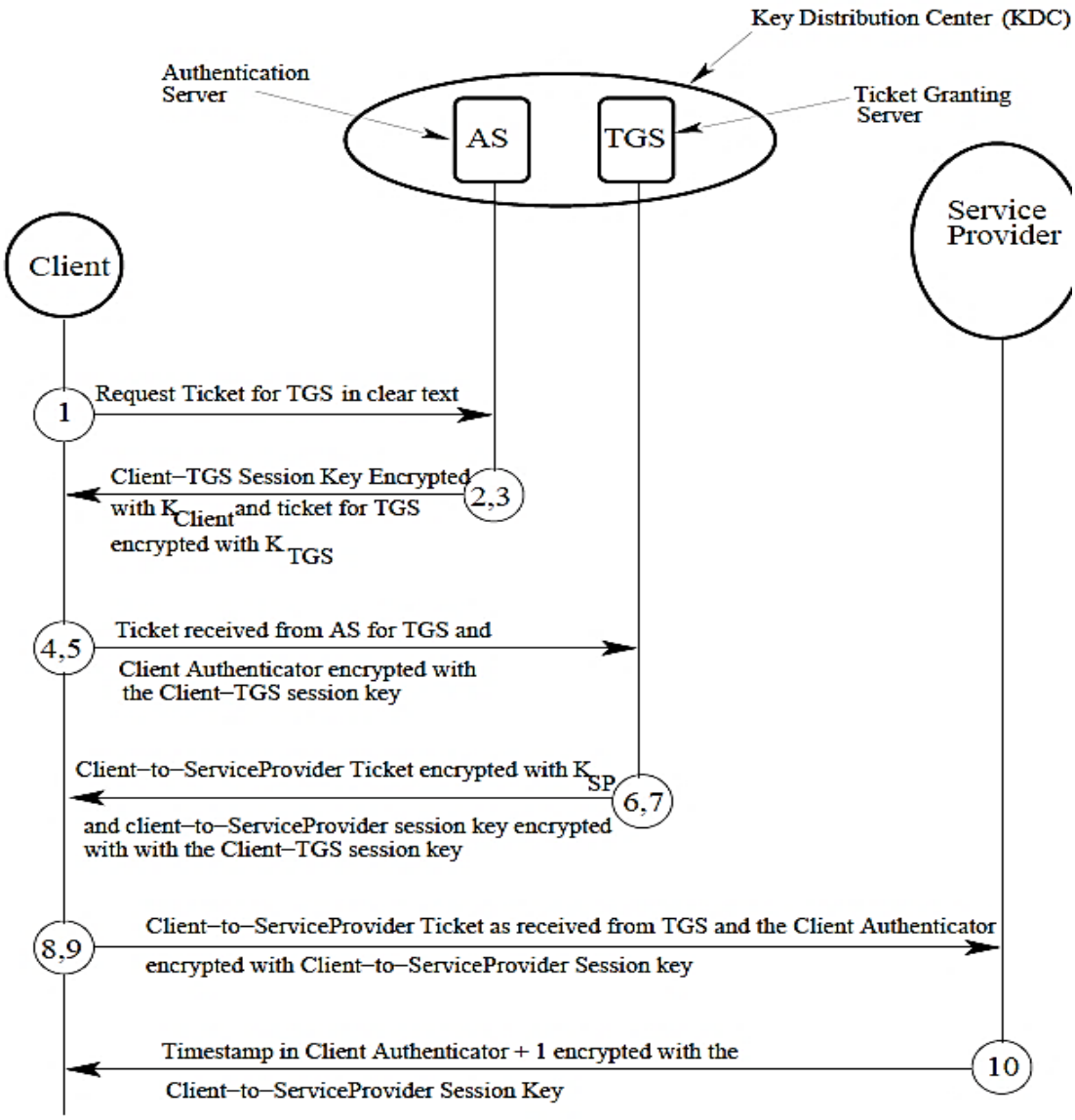
i. KClient: secret key held by AS for the Client. this encryption key is not directly known to the Client.

ii. KTGS: secret key held by AS for TGS. TGS also has this key.

iii. KServiceProvider: secret key held by AS for the Service Provider. The Service Provider also has access to this key.

iv. KClient-TCG: session key that AS will send to Client for communicating with TGS.

v. KClient-ServiceProvider: session key that TGS will send to the Client for communicating with the Service Provider.



IV. Process

i. Each Client registers with the Authentication Server and is granted a user identity and a secret password.

ii. Client sends a request in plain text to the AS.

iii. AS sends back to the Client the following two messages encrypted with the KClient key.

1. A session key KClient-TGS that the client can use to communicate with TGS
2. Ticket-Granting Ticket (TGT) / initial ticket: encrypted with KTGS secret key that the AS server maintains for TGS, meant for delivery to TGS. Also called This ticket includes
    I. clients user ID
    II. clients network address
    III. validation time
    IV. session key KClient-TGS

iv. client receives the above messages and enters his/her password into a dialog box. An algorithm converts this password into what would be the KClient encryption key if the password is correct. The password is immediately destroyed and the generated key used to decrypt the messages received from AS. The decryption allows the Client to extract the session key KClient-TGS and the ticket meant for TGS from the information received from AS.

v. client sends the following two messages to TGS:

1. The encrypted ticket meant for TGS followed by the ID of the requested service. If the client wants to access an FTP server, this would be the ID of the FTP server.
2. A Client Authenticator that is composed of the client ID and the timestamp, the two encrypted with the KClient-TGS session key.

vi. TGS recovers the ticket from the first of the two messages listed above. From the ticket, it recovers the KClient-TGS session key and uses the session key to decrypt the second message listed above that allows it to authenticate the Client.

vii. TGS now sends back to the Client the following two messages:

1. A Client-to-ServiceProvider ticket that consists of:
    I. The Client ID
    II. the Client network address
    III. the validation period
    IV. a session key for the Client and the Service Provider,

KClient.ServiceProvider. This session key is encrypted with the KServiceProvider key that is known to TGS.

2. KServiceProvider key session key encrypted with KClient-TGS session key

viii. The client recovers the ticket meant for the service provider with KClient-TGS session key.

ix. client sends the following two messages to the service provider:

1. The Client-to-ServiceProvider ticket that was encrypted by TGS with the KServiceProvider key.
2. An authenticator that consists of the Client ID and the timestamp. This authenticator is encrypted with the KClien-ServiceProvider session key.

x. The Service Provider decrypts the ticket with its own KServiceProvider key. It extracts the KClient-ServiceProvider session key from the ticket, and uses the session key to decrypt the second message received from the client.

xi. If the client is authenticated, the ServiceProvider sends to the Client a message that consists of the timestamp in the authenticator received from the Client plus one. This message is encrypted using the KClient-ServiceProvider session key

xii. client decrypts the message received from the Service Provider using KClient-ServiceProvider session key and makes sure that the message contains the correct value for the timestamp. If that is the case, the client can start interacting with the Service Provider.

3. An advantage of separating AS from TGS (although they may reside in the same machine) is that the Client needs to contact AS only once for a Client-to-TGS ticket and the Client-to-TGS session key. These can subsequently be used for multiple requests to the different service providers in a network.
4. GSS-API (Generic Security Services Application Programming Interface.) is an official standard and Kerberos is the most common implementation of this API.

■Secure communications in computer networks is impossible without high quality random and pseudorandom number generation. Reasons:

1. session keys that a KDC must generate on the fly are bytes. For a sequence of randomly generated purpose of transmission over character-oriented channels (as is the case with all internet communications), each byte could be represented by two hex digits. So a 128-bit session key would be a string of 32 hex
2. nonces that are exchanged during handshaking between a host and a KDC, and amongst hosts are also random numbers.

3. random numbers are needed for the RSA public-key encryption algorithm. RSA needs prime numbers. Since there do not exist methods that can generate prime numbers directly, we generate random numbers and testing them for primality.
4. random numbers are served as salts in password hashing schemes. Combine randomly generated bits with the string of characters entered by user as his/her password, and then hash the whole thing to create a password hash. Salts make it much more challenging to crack passwords by table lookup.
5. You need true random numbers to serve as one-time keys.

■To be considered truly random, a sequence of numbers must exhibit two properties:

1. Uniform Distribution: all numbers in a designated range must occur equally often.
2. Independence: if we know some or all the number up to a certain point in a random sequence, we should not be able to predict the next one (or any of the future ones).

■Algorithmically generated random numbers are called pseudorandom numbers.

■pseudorandom numbers generated by Linear Congruential Generator algorithm do not pass muster when security is involved

1. sequence of pseudorandom numbers X0, X1,., Xi, . is generated using recursion:

$$X_{n+1} = (a \cdot X_n + c) \bmod m$$

m	*the modulus*	$m > 0$
a	*the multiplier*	$0 < a < m$
c	*the increment*	$0 \leq c < m$
X_0	*the seed*	$0 < X_0 < m$

2. how random the produced sequence of numbers depends on values chosen for m, a, and c. For a given choice of m, c, a, the next number depends only on the current number. For example, if a=c= 1 results in a predictable sequence.
3. criteria on how to select values for these parameters m, a, c:
   I. To the maximum extent possible, the selected parameters should yield a full-period sequence of numbers. The period of a full-period sequence is equal to the size of the modulus. In a full-period sequence, each number between 0 and m-1 will appear only once in a sequence of m numbers.
   II. When m is a prime and c is zero, then for certain value of a, the recursion formula shown above is guaranteed to produce a sequence of period m-1. Such a sequence will have the number 0 missing. But every number n, 0 < n < m, will make exactly one appearance in such a sequence.
   III. The sequence produced must pass a suite of statistical tests to evaluate its randomness. how uniform the distribution of the sequence of numbers is

and how statistically independent the numbers are.

4. for a 4-byte signed integer representation, m would commonly be set to 2^31-1. With c= 0, A commonly used value for a is 7^5 = 16807.our recursion for generating a pseudorandom sequence then becomes

$$X_{n+1} = (a \cdot X_n) \bmod (2^{31} - 1)$$

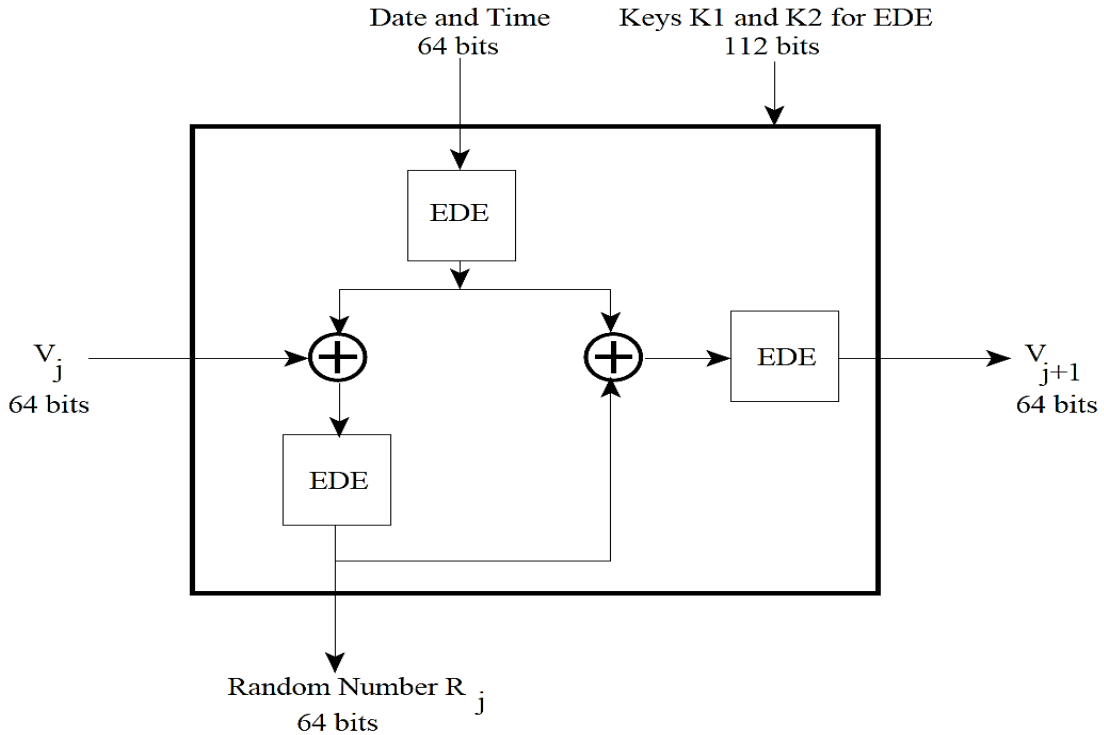
5. A pseudorandom sequence of numbers is cryptographically secure if it is difficult for an attacker to predict the next number from the numbers already in his/her possession.
6. When linear congruential generators are used for producing random numbers, the attacker only needs three pieces of information to predict the next number from the current number: m, a, c

$$X_1 = (a \cdot X_0 + c) \bmod m$$
$$X_2 = (a \cdot X_1 + c) \bmod m$$
$$X_3 = (a \cdot X_2 + c) \bmod m$$

7. A pseudorandom sequence produced by a PRNG (pseudorandom number generator) can be made more secure from a cryptographic standpoint by restarting the sequence with a different seed after every N numbers. One way to do this would be to take the current clock time modulo m as a new seed after every so many numbers of the sequence have been produced.

■cryptographically secure pseudorandom number generator (CSPRNG)

1. ANSI X9.17/X9.31 Pseudorandom Number Generator. is driven by two encryption keys and two special inputs that change for each output number in a sequence.



I. Each of the three EDE boxes stands for the two-key 3DES algorithm. All three EDE boxes use the same two 56-bitencryption keys K1 and K2.

II. two inputs are:

i. A 64-bit representation of the current date and time (DTj); and

ii. A 64-bit number generated when the previous random number was output (Vj).

III. The PRNG is initialized with a seed value for V0 for the very first random number that is output.

IV. The output of the PRNG consists of the sequence of pairs (Rj, Vj+1), j=0,1,2…, Rj is the jth random number produced and Vj+1 is the input for the (j+1)th iteration of the algorithm

$$R_j = EDE([K_1, K_2], [V_j \otimes EDE([K_1, K_2], DT_j)])$$
$$V_{j+1} = EDE([K_1, K_2], [R_j \otimes EDE([K_1, K_2], DT_j)])$$

V. reasons contribute to the cryptographic security of this approach to PRNG:

i. difficult-to-predict pseudorandom seed for each random number We can think of Vj+1 as a new seed for the next random number to be generated. This seed cannot be predicted from current random number Rj.

ii. the scheme uses independently specified pseudorandom input an encryption of the current date and time.

iii. Each random number is related to the previous random number through multiple stages of DES encryption. There exist nine DES encryptions between two consecutive random numbers, making it virtually impossible to predict the next random number from the current random number.

iv. Even if the attacker were to somehow get hold of the current Vj, it would still be practically impossible to predict Vj+1 because there stand at least two EDE encryptions between the two.

VI. it is a much slower way to generate pseudorandom numbers. That makes this approach unsuitable for many applications that require randomized inputs.

2. BLUM BLUM SHUB GENERATOR (BBS) / cryptographically secure pseudorandom bit generator (CSPRBG)

I. Algorithm

i. first choosing two large prime numbers p and q that both yield a remainder of 3 when divided by 4. Let n = p*q

ii. choose a number s that is relatively prime to n.

iii. BBS generator produces a pseudorandom sequence of bits Bj: (Bi is

the least significant bit of Xi at each iteration.)

$$X_0 = s^2 \mod n$$

$$\begin{aligned} for\ i &= 1\ to\ \inf \\ X_i &= (X_{i-1})^2 \mod n \\ B_i &= X_i \mod 2 \end{aligned}$$

II. CSPRBG must pass the next-bit test: not exist a polynomial-time algorithm that can predict the kth bit given the first k-1 bits with a probability significantly greater than 0.5.

■A fundamental difference between a PRNG and TRNG (True Random Number Generator) is that whereas the former must have a seed for initialization, the latter works without seeds. This fundamental difference between a PRNG and TRNG also applies to the difference between a CSPRNG and TRNG.

1. only the analog phenomena can be trusted to produce truly random numbers. We will consider an to be any source entropy source that is capable of yielding a TRULY random stream of 1s and 0s.
2. entropy sources, in general, are not capable of providing random bits at the rate needed by high-performance applications. For such applications, the best they can do is to serve as the seeds needed by CSPRNGs
3. bit stream produced by an entropy source into bytes, pi =1/256

$$H = -\sum_i p_i \log_2 p_i$$

4. if a network device were to use a poor quality random number generator (one whose random numbers are predictable) it would be much too vulnerable to security exploits. The more nonuniform the probabilities of the values taken by the random numbers, the more predictable they become.
5. A one-time random number means that there is very little chance that the same random number will be used again in the foreseeable future.
6. There are two types of entropy sources to consider
   I. the on-chip hardware based entropy sources: uses two inverters with the output of one connected to the input of the other. When the output of one inverter is 1, the output of the other must be 0. As to which inverter would output a 1 and which would output a 0 depends on the thermal noise that accompanies the 1-to-0 and 0-to-1 transitions of the circuit elements. This bit stream must subsequently be conditioned to compensate for any biases in randomness caused by the two inverters not being truly identical. Finally, the conditioned bits are used to initialize a hardware implementation of a CSPRNG for higher production rates of the random bytes. Intel also

provides a machine-code instruction, RDRAND, for 64-bit processors for fetching random numbers from the DRNG.

II. software based entropy sources.
   i. every computer there are constantly occurring phenomena that are consequences of some human interaction with that computer or some other networked computer.
   ii. software sources of entropy can be divided into two categories:
      1. kernel space: those that can only be accessed with root privileges. /dev/random gathers entropy in the kernel space. It is based on the randomness associated with keystrokes, mouse movements, disk I/O, device driver I/O, etc.
      2. user space: those that are accessible with ordinary user privileges. The random bits made available by user space entropy sources can be obtained either through EGD (Entropy Gathering Daemon) or through PRNGD (Pseudo Random Number Generator Daemon).
   iii. For a non-blocking kernel space source of entropy, you can use /dev/urandom that uses the random bits supplied by /dev/random to initialize a CSPRNG to produce a very high-quality stream of pseudorandom bytes. Being pseudorandom, the byte stream produced by /dev/urandom will have less entropy than the byte stream coming from /dev/random.
   iv. PRNGD uses the random bits collected from its entropy sources to seed a CSPRNG

■Two integers m and n are coprimes if and only if their Greatest Common Divisor is equal to 1. number 1 is coprime to every integer.

■Fermats Little Theorem: when p is a prime, then for any integer a that is coprime to p, the following relationship must hold $a^{p-1} \equiv 1 \pmod{p}$
(a = 0 and as that are multiples of p are excluded specifically.)

1. Proof:
   I. assuming that p is prime and a is a non-zero integer that is coprime to p: a, 2a, 3a, 4a, ..., (p-1)a
   II. if we reduce these numbers modulo p, we will simply obtain a rearrangement of the sequence: 1, 2, 3, 4, ..., (p-1)
   III. Therefore, we can say {a, 2a, 3a, ..., (p-1)a} mod p = some permutation of {1, 2, 3, ..., (p-1)} for every prime p and every a that is coprime to p.
   IV. multiplying all of the terms 2a, 3a, 4a, ..., (p-1)a yield $a^{p-1} \times 1 \times 2 \ldots \times p-1 \equiv 1 \times 2 \ldots \times p-1 \ (mod\ p)$

V. Canceling out the common factors on both sides then gives the Fermats Little Theorem

2. the relationship of Fermats Little Theorem is also satisfied by numbers that are composite. For example, consider the case n = 25 and a = 7. So if Fermats Little Theorem is satisfied for a given number n for a random choice for a, try another choice for a. [Fermats Little Theorem must be satisfied by every a that is coprime to n.] The larger the number of probes, as, you use for a given n, with all the as satisfying Fermats Little Theorem, the greater the probability that n is a prime. You stop testing as soon you see the theorem not being satisfied for some value of a, since that is an iron-clad guarantee that n is NOT a prime.

■Euler's Totient Function ϕ(n).

1. For a given positive integer n, ϕ(n) is the number of positive integers less than or equal to n that are coprime to n. ϕ(n) is known as the totient of n. [0 cannot be a coprime to any integer n]

ints:	1	2	3	4	5	6	7	8	9	10	11	12 ..
totients:	1	1	2	2	4	2	6	4	6	4	10	4 ..

2. If p is prime, its totient is given by ϕ(p) = p-1.
3. Suppose a number n is a product of two primes p and q, n=pq then

$$\phi(n) = \phi(p) \cdot \phi(q) = (p-1)(q-1)$$

■EULERS THEOREM: for every positive integer n and every a that is coprime to n, the following must be true $a^{\phi(n)} \equiv 1 \pmod{n}$

1. when n is a prime, ϕ(n) = n-1. Eulers Theorem reduces to the Fermats Little Theorem. However, Eulers Theorem holds for all positive integers n as long as a and n are coprime.
2. Proof
   I. R = {x1, x2,…,$x_{\phi(n)}$} the set of all integer less than n that are relatively prime to n.
   II. S be the set obtained when we multiply modulo n each element of R by some integer a co-prime to n.

$$S = \{a \times x_1 \bmod n,\ a \times x_2 \bmod n, \ldots,\ a \times x_{\phi(n)} \bmod n\}$$

   III. S is simply a permutation of R implies that multiplying all of the elements of S should equal the product of all of the elements of R.
      i. (a× $x_i$mod n) cannot be zero: because a and xi are coprimes to n, so a× x_i cannot contain n as a factor.
      ii. (a× $x_i$mod n) = (a× $x_j$mod n) is impossible: if (a× x_i − a × x_j mod n) = 0, either a is 0 mod n or $x_i \equiv x_j$ mod n

$$\prod_i s_i \in S \bmod n \quad = \quad \prod_i r_i \in R \bmod n$$

$$a^{\phi(n)} \times \prod_i r_i \in R \quad \equiv \quad \prod_i r_i \in R \pmod{n}$$

■given an odd integer, its least significant bit will be 1. if an odd integer is multiplied k times by 2, you would be shifting the bit pattern for q to the left by k positions.

■Miller-Rabin algorithm. Based on an Intuitive Decomposition of an Even Number into Odd and Even Parts

1. it only makes a probabilistic assessment of primality: If the algorithm says that the number is composite, then the number is definitely not a prime. If the algorithm says that the number is a prime, then with a very small probability the number may not actually be a prime.
2. all the Miller-Rabin test does is to check whether or not the equality $a^{p-1} \equiv 1(mod\ p)$ is satisfied for a prime p and for a set of values for the probe a.
3. Miller-Rabin test is carried out in a computationally efficient manner by exploiting a factorization of the even number p-1.
4. observation
   I. Given any odd positive integer n, we can express n-1 as a product of a power of 2 and a smaller odd number:

   $$n - 1 = 2^k \cdot q \quad for\ some\ k > 0,\ and\ \mathbf{odd}\ q$$

   for any prime p, it being an odd number, the following must hold

   $$p - 1 = 2^k \cdot q \quad for\ some\ k > 0,\ and\ odd\ q$$

   So

   $$a^{p-1} \equiv 1\ (mod\ p)$$

   $$a^{2^k \cdot q} \equiv 1 \pmod{p}$$

   II. x^2= 1 has only trivial roots in Zp for any prime p: mean that only x = 1 and x =-1 can satisfy the equation x^2= 1.

   $$-1 \equiv (p - 1) \pmod{p}$$

   there exist only two numbers,1 and 1, in the field that when squared give us 1. (1*1 mod p = 1, -1*-1 mod p = 1). Besides 1 and -1, there do not exist any other integers x in Zp that when squared will return 1 mod p.

   $$x^2 \equiv 1 \pmod{p}$$

   $$x^2 - 1 \equiv 0 \pmod{p}$$

   $$x^2 - x + x - 1 \equiv 0 \pmod{p}$$

   $$(x - 1) \cdot (x + 1) \equiv 0 \pmod{p}$$

3. algorithm: For any integer a in the range 1 < a < p-1, one of the following conditions must be true when p is a prime:

I. Either it must be the case that

$$a^q \equiv 1 \pmod{p}$$

II. Or, it must be the case that (1<=j<=k)

$$a^{2^{j-1}q} \equiv -1 \pmod{p}$$

4. if either of the two Conditions is true for a probe a, then p may be either a composite or a prime.
5. When n is to be composite, then known

$$a^q \not\equiv 1 \quad a^{2^i \cdot q} \not\equiv -1 \bmod n \quad for\ all\ 0 < i < k-1$$

   I. All such as are called witnesses for the compositeness of n. at least 3/4th of the numbers a < n will be witnesses for its compositeness.
   II. When a randomly chosen a for a known composite n does not satisfy the dual test above, it is called a liar for the compositeness of n.
6. probability of a composite number being declared prime by the Miller-Rabin algorithm is significantly less than 4^-t, t is the number of probes used.
7. running time of this algorithm is O(tlog^3n) or O(tlog^2n) where n is the integer being tested for its primality and t the number of probes used for testing.
8. In the theory of algorithms, the Miller-Rabin algorithm would be called a randomized algorithm belongs to the class co-RP. (randomized polynomial)
   I. randomized algorithm: algorithm that can make random choices during its execution.
   II. co-RP: problems that can be solved in polynomial time but when the answer is known to be no, the algorithm occasionally says yes
   III. co-RP is a subset of the class BPP.(bounded probabilistic polynomial-time). These are randomized polynomial-time algorithms that yield the correct answer with an exponentially small probability of error.

```
def MillerRabin_primality_test(p):
    if p == 1: return 0
    #Researchers have shown that using these for probes suffices
    #for primality testing for integers smaller than 341,550,071,728,321.
    probes = [2,3,5,7,11,13,17]
    if p in probes: return 1
    if any([p % a == 0 for a in probes]): return 0
    k, q = 0, p-1          # need to represent p-1 as  q * 2^k
    while not q&1:
        q >>= 1
        k += 1
    for a in probes:
        a_raised_to_q = pow(a, q, p)
        if a_raised_to_q == 1: continue
        if (a_raised_to_q == p-1) and (k > 0): continue
        a_raised_to_jq = a_raised_to_q
        primeflag = 0
        for j in range(k-1):
            a_raised_to_jq = pow(a_raised_to_jq, 2, p)
            if a_raised_to_jq == p-1:
                primeflag = 1
                break
        if not primeflag: return 0
    probability_of_prime = 1 - 1.0/(4 ** len(probes))
    return probability_of_prime
```

■Agrawal-Kayal-Saxena (AKS) algorithm: if a number a is coprime to another number p, p > 1, then p is prime if and only if the (x+a)^p defined over the finite field Zp obeys:

$$(x + a)^p \equiv x^p + a \pmod{p}$$

$$(x + a)^p \bmod (x^r - 1) = (x^p + a) \bmod (x^r - 1)$$

1. two main challenges in creating an efficient implementation
   I. For large candidate numbers, the number of iterations of the while loop for finding an appropriate value for r may be large enough to require that you use the binary GCD algorithm as opposed to the regular Euclids algorithm
   II. in line (B) where you are supposed to figure out whether, for the given value for a, the polynomial (x+a)^p is congruent to the polynomial x^p-a modulo the polynomial x^r-1.

```
p  =  integer to be tested for primality
if ( p == a^b for some integer a and for some integer b > 1 ) :
        then return ‘‘p is COMPOSITE’’
r = 2

### This loop is to find the appropriate value for the number r:
while r < p:
    if ( gcd(p,r) is not 1 ) :                                  # (A)
        return "p is COMPOSITE"

    if ( r is a prime greater than 2 ):
        let q be the largest factor of r-1
        if ( q > (4 . sqrt(r) . log p) )   and
           ( p^{(r-1)/q} is not 1 mod r ) :
             break
    r = r+1

### Now that r is known, apply the following test:
for a = 1 to (2 . sqrt(r) . log p) :
    if ( (x-a)^p is not (x^p - a) mod (x^r - 1):                #(B)
        return "p is COMPOSITE"

return "p is PRIME"
```

2. computational complexity of the AKS algorithm $O\left((\log p)^{12} \cdot f(\log\log p)\right)$ where p is the integer whose primality is being tested and f is a polynomial. So the running time of the algorithm is proportional to the twelfth power of the number of bits required to represent the candidate integer times a polynomial function of the logarithm of the number of bits.

■CHINESE REMAINDER THEOREM (CRT): Particularly useful for modulo arithmetic operations on very large numbers with respect to large moduli.

1. Algorithm
   I. in modulo M arithmetic, if M can be expressed as a product of n integers that are pairwise coprime, then every integer in the set ZM={0, 1, 2, ...,M-1} can be reconstructed from residues with respect to those n numbers.
   II. EX: 10=2*5 => according to CRT, 9 can be represented by the tuple (1, 4)
   III. CRT allows us to represent any integer A in ZM by the k-tuple and makes two assertions about the k-tuple representations for integers::
      i. The mapping between the integers A ∈ ZM and the k-tuples is a bijection, meaning that the mapping is one-to-one and onto.
      ii. Arithmetic operations on the numbers in ZM can be carried out equivalently on the k-tuples. When operating on the k-tuples, the operations can be carried out independently on each of coordinates of the tuples

$$
\begin{aligned}
(A + B) \bmod M &\Leftrightarrow ((a_1 + b_1) \bmod m_i, \ldots, (a_k + b_k) \bmod m_k) \\
(A - B) \bmod M &\Leftrightarrow ((a_1 - b_1) \bmod m_i, \ldots, (a_k - b_k) \bmod m_k) \\
(A \times B) \bmod M &\Leftrightarrow ((a_1 \times b_1) \bmod m_i, \ldots, (a_k \times b_k) \bmod m_k)
\end{aligned}
$$

where $A \Leftrightarrow (a_1, a_2, \ldots, a_k)$ and $B \Leftrightarrow (b_1, b_2, \ldots, b_k)$ are two arbitrary numbers in Z_M.

   IV. To compute the number A for a given tuple (a1, a2,…,ak)
      i. we first calculate Mi= M/mi, 1<= mi <= k

$$M_i \equiv 0 \pmod{m_j} \quad for \ all \ j \neq i$$

      ii.

$$c_i = M_i \times (M_i^{-1} \bmod m_i) \quad for \ all \ 1 \leq i \leq k$$

      iii.

$$A = \left(\sum_{i=1}^{k} a_i \times c_i \right) \bmod M$$

2. CRT is useful for manipulating very large numbers in modulo arithmetic
   I. Modulus M = 8633 = 89*97 => m1 =89, m2 =97
   II. M1= M/m1= 97 and M2= M/m2= 89.
   III. $M_1^{-1} \ mod \ m_1 = 78, \ M_2^{-1} \ mod \ m_1 = 12$
   IV. we want to add two integers 2345 and 6789 modulo 8633.
   V. express the operand 2345 by its CRT representation, which is (31, 17) since 2345 mod 89 = 31 and 2345 mod 97 = 17.
   VI. express the operand 6789 by its CRT representation,which is (25, 96) since 6789 mod 89 = 25 and 6789 mod 97 = 96.
   VII. To add the two large integers, we simply add the two corresponding CRT tuples modulo the respective moduli. This gives us (56, 16).

VIII. To recover the result as a single number, that returns the result 501.

$$a_1 \times M_1 \times M_1^{-1} \quad + \quad a_2 \times M_2 \times M_2^{-1} \qquad \text{mod } M$$

which for our example becomes

$$56 \times 97 \times 78 \quad + \quad 16 \times 89 \times 12 \qquad \text{mod } 8633$$

■DISCRETE LOGARITHMS

1. primitive root modulo a positive number N: for any positive integer N, the set of all integers i < N that are coprime to N form a group with modulo N multiplication as the group operator. For example, when N = 8, the set of coprimes is {1, 3, 5, 7} This set forms a group with modulo N multiplication as the group operator. Denoted $(Z/8Z)^x$ = {1, 3, 5,7}. Choosing a prime for N, Z_{17}^8 = {1,2,3,…,16}
2. For some values of N, the set $(Z/NZ)^x$ contains an element whose various powers, when computed modulo N, are all distinct and span the entire set (Z/NZ). Such an element is the primitive element of the set $(Z/NZ)^x$ or primitive root modulo N. for example, N = 9, 2 is a primitive element of the group $(Z/9Z)^x$, which is the same as primitive root mod 9. A primitive root can serve as the base of discrete logarithm.

$$\begin{aligned} Z_9 &= \{0,\ 1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7,\ 8\} \\ (Z/9Z)^\times &= \{1,\ 2,\ 4,\ 5,\ 7,\ 8\} \end{aligned}$$

$$x^y \equiv z \pmod N \qquad dlog_{x,N}\ z = y$$

$$\begin{aligned} 2^0 &= 1 \\ 2^1 &= 2 \\ 2^2 &= 4 \\ 2^3 &= 8 \\ 2^4 &\equiv 7 \pmod 9 \\ 2^5 &\equiv 5 \pmod 9 \\ \cdots &\quad \cdots \quad \cdots \\ 2^6 &\equiv 1 \pmod 9 \\ 2^7 &\equiv 2 \pmod 9 \\ 2^8 &\equiv 4 \pmod 9 \\ &\vdots \end{aligned} \qquad \begin{aligned} dlog_{2,9}\ 1 &= 0 \\ dlog_{2,9}\ 2 &= 1 \\ dlog_{2,9}\ 4 &= 2 \\ dlog_{2,9}\ 8 &= 3 \\ dlog_{2,9}\ 7 &= 4 \\ dlog_{2,9}\ 5 &= 5 \\ \cdots &\quad \cdots \quad \cdots \\ dlog_{2,9}\ 1 &= 6 \\ dlog_{2,9}\ 2 &= 7 \\ dlog_{2,9}\ 4 &= 8 \\ &\cdots \end{aligned}$$

3. unique discrete logarithm mod N to some base a exists only if a is a primitive root modulo N.

■Public-key cryptography / asymmetric-key cryptography: Encryption and decryption are carried out using public key and the private key.

1. With public key cryptography, all parties interested in secure communications

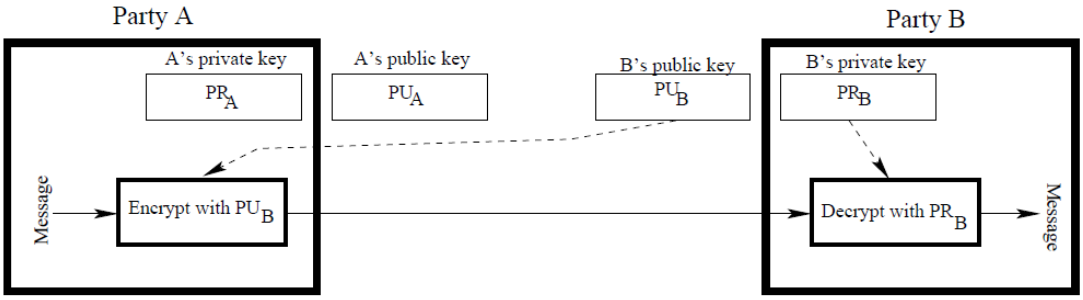
publish their public keys. How that is done depends on the protocol.

I. SSH protocol: each server makes its public key stored available through port 22 for your login id on the server. When a client wants to connect with an SSHD server, it sends a connection request to port 22 of the server machine and the server makes its host key available automatically.

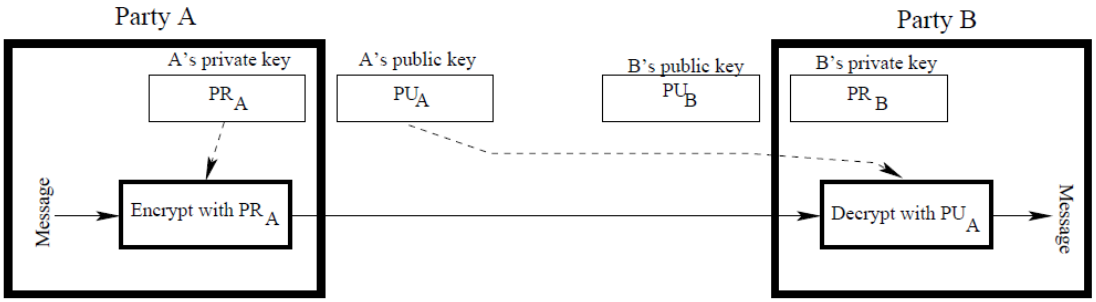
II. SSL/TLS protocol: an HTTPS web server makes its public key available through a certificate

2. confidentiality: protect a message from eavesdroppers. Party A can encrypt a message using B's publicly key to communicate confidentially with party B, communication would only be decipherable by B's private key.
3. authentication: recipient needs a guarantee as to the identity of the sender. Party A would encrypt the message with As private key (A putting his digital signature on message) to send an authenticated message to party B, to show A was indeed the source of the message.

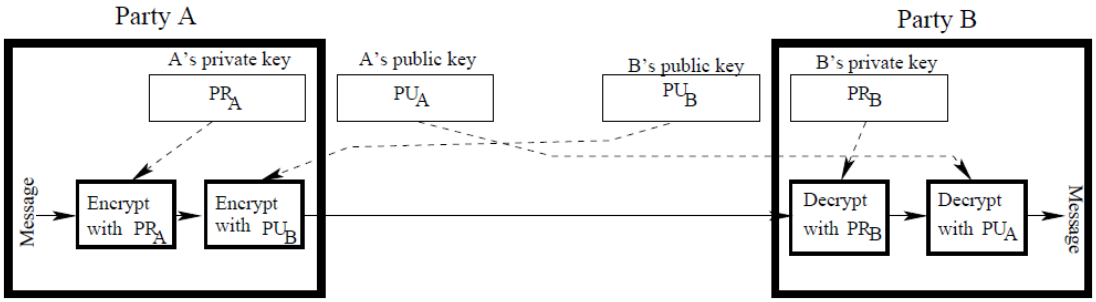
When only confidentiality is needed:



When only authentication is needed:



When both confidentiality and authentication are needed:



4. send a message M to B with both authentication and confidentiality: The message goes through two encryptions at the senders place and two decryptions at the receivers place. Each of these four steps involves separately the computationally complex public-key algorithm.
   I. The processing steps undertaken by A to convert M into C
   $$C = E(PU_B, E(PR_A, M))$$
   II. The processing steps undertaken by B to recover M from C
   $$M = D(PU_A, D(PR_B, C))$$
   III. Because of the greater computational overhead associated with public-key crypto systems, symmetric-key systems continue to be widely used for content encryption.

■RIVEST-SHAMIR-ADLEMAN (RSA) ALGORITHM: public-key cryptography was made possible by this algorithm

1. Eulers Theorem: for every positive integer n and every a that is coprime to n
   $$a^{\phi(n)} \equiv 1 \pmod{n}$$
2. when a and n are relatively prime, the exponents will behave modulo the totient -ϕ(n) in exponentiated forms like a^k mod n
3. if a and n are relatively prime, the following must be true
   $$a^k \equiv a^{k_1 \cdot \phi(n) + k_2} \equiv a^{k_1 \cdot \phi(n)} a^{k_2} \equiv a^{k_2} \pmod{n}$$
4. For example, a = 4 in arithmetic modulo 15. The totient of 15 is 8. (15 = 3*5, so - ϕ(15) = 2*4 = 8.)
   $$4^7 \cdot 4^4 \bmod 15 = 4^{(7+4) \bmod 8} \bmod 15 = 4^3 \bmod 15 = 64 \bmod 15 = 4$$
   $$(4^3)^5 \bmod 15 = 4^{(3\times 5) \bmod 8} \bmod 15 = 4^7 \bmod 15 = 4$$
5. M is an integer that represents a message, conjure up two integers e and d that are each others multiplicative inverses modulo the totient ϕ(n).
   $$M^{e\times d} \equiv M^{e\times d \pmod{\phi(n)}} \equiv M \pmod{n}$$
   I. N = a modulus for modular arithmetic
   II. ϕ(n) = the totient of n
   III. e = an integer that is relatively prime to ϕ(n) [This guarantees that e will possess a multiplicative inverse modulo - ϕ(n)]
   IV. d = an integer that is the multiplicative inverse of e modulo ϕ(n)
   $$C = M^e \bmod n \qquad M = C^d \bmod n$$
   $$(M^e)^d \pmod{n} = M^{ed \pmod{\phi(n)}} \equiv M \pmod{n}$$
   I. An individual A who wishes to receive messages confidentially will use the pair of integers {e,n} as his/her public key and {d, n} as private key.
   II. Party B wishing to send a message M to A confidentially will encrypt M using As public key {e, n} to create ciphertext C. Subsequently, A will

decrypt C using his private key {d,n}

III. If the plaintext message M is too long, B may choose to use RSA as a block cipher for encrypting the message meant for A. When RSA is used as a block cipher, the block size is likely to be half the number of bits required to represent the modulus n. If the modulus required 1024 bits for its representation, message encryption would be based on 512-bit blocks.

6. Eulers theorem, requires M and n be coprime. However, when n is a product of two primes p and q, this result applies to all M, 0<M < n.
   I. If two integers p and q are coprimes
   $$\{a \equiv b \pmod{p} \quad and \quad a \equiv b \pmod{q}\} \quad \Leftrightarrow \quad \{a \equiv b \pmod{pq}\}$$
   II. Only when p and q are individually prime that
   $$\phi(n) \quad = \quad \phi(p) \times \phi(q) \quad = \quad (p-1) \times (q-1)$$
   III. it is important that both p and q be very large primes, so it is computationally harder to determine its primality.
   IV. We also need to ensure that n is not factorizable by one of the modern integer factorization algorithms.
7. Proof of the RSA Algorithm: prove that when n is a product of two primes p and q, then, in arithmetic modulo n, the exponents behave modulo the totient of n.
   I. since the integer d is the multiplicative inverse of the integer e modulo the totient - ϕ(n)
   $$e \times d \equiv 1 \pmod{\phi(n)}$$
   $$e \times d - 1 \equiv 0 \pmod{\phi(n)} = k \times \phi(n)$$
   II. since - ϕ(n) = - ϕ(p)* ϕ(q), ϕ(p) and - ϕ(q) must also individually be divisors of e*d-1
   $$e \times d - 1 = k_1\phi(p) = k_1(p-1).$$
   $$M^{e \times d} \bmod p = M^{e \times d - 1 + 1} \bmod p = M^{k_1(p-1)} \times M \bmod p$$
   III. if M and p are coprimes. By Fermats Little Theorem
   $$M^{p-1} \equiv 1 \pmod{p}$$
   $$M^{k_1(p-1)} \equiv 1 \pmod{p}$$
   $$M^{e \times d} \bmod p = M \bmod p$$
   IV. if M is a multiple of the prime p, M mod p = 0 => M^k mod p = 0
   $$M^{e \times d} \bmod p = M \bmod p$$
   IV. we can draw identical conclusion regarding q
   $$M^{e \times d} \bmod q = M \bmod q$$
   V. when p and q are coprimes, for any integers a and b if we have a≡b (mod p) and a ≡ b (mod q), then it must also be the case that a ≡b (mod pq).
   $$M^{e \times d} \bmod n = M \bmod n$$
8. computational steps for key generation

I. Generate two different primes p and q
   i. generate a random number of size B/2 bits.
   ii. set the lowest bit of the integer generated to ensure it is odd number
   iii. set the two highest bits to ensure the highest bits of n is set.
   iv. Using the Miller-Rabin algorithm, check to see if the resulting integer is prime. If not,you increment the integer by 2 and check again.

II. Calculate the modulus n = p*q

III. Calculate the totient -ϕ(n) = (p-1)*(q-1)

IV. Select for public exponent an integer e such that 1 < e < -ϕ(n) and gcd(ϕ(n), e) = 1 [equivalent to gcd(p-1, e) = 1 and gcd(q-1, e) = 1]
   i. For computational ease, one typically chooses a value for e that is prime, has as few bits as possible equal to 1 for fast multiplication. Typical values for e are 3, 17, and 65537
   ii. Small values for e, such as 3, are considered cryptographically insecure. Sender A sends the same message M to three different receivers using their respective public keys that have the same e = 3 but different values of n (n1, n2, and n3). Attacker can intercept all three transmissions and see three ciphertext messages: C1 = M^3 mod n1, C2 = M^3 mod n2, C3 = M^3 mod n3. Assume n1, n2, n3 are relatively prime, the attacker can use the Chinese Remainder Theorem (CRT) to reconstruct M^3 mod modulo N = n1*n2*n3. All the attacker has to do is to figure out the cube-root of M^3 to recover M.

V. Calculate for the private exponent a value for d such that d = e^(-1) mod -ϕ(n) [modular inversion]
   i. use the Extended Euclids Algorithm: e=17, ϕ(n)=41160, d=26633

```
gcd(17, 41160)        |
  = gcd(41160, 17)    |  residue 17 =  0 x 41160  +  1 x 17
  = gcd(17, 3)        |  residue  3 =  1 x 41160  -  2421 x 17
  = gcd(3,2)          |  residue  2 =  -5 x 3     +  1 x 17
                      |             =  -5x(1 x 41160 - 2421 x 17) + 1 x 17
                      |             =  12106 x 17  - 5 x 41160
  = gcd(2,1)          |  residue  1 =  1x3  -  1 x 2
                      |             =  1x(41160 - 2421x17)
                      |                    - 1x(12106x17 -5x41160)
                      |             =  6 x 41160   - 14527 x 17
                      |             =  6 x 41160   + 26633 x 17
```

VI. Public Key= [e, n]
   i. modular exponentiation: raising the message integer M to the power of the public exponent e modulo n.
   ii. Algorithm for Modular Exponentiation
      1. A^B calculation can be speeded up by expressing B as a sum of

smaller parts, then result is a product of smaller exponentiations.

```
result = 1
while B > 0:
    if B & 1:                              # check the lowest bit of B
        result = ( result * A ) % n
    B = B >> 1                             # shift B by one bit to right
    A = ( A * A ) % n
return  result
```

VII. Private Key = [d, n]

i. speeded up by using the Chinese Remainder Theorem (CRT)

$$V_p = C^d \bmod p \quad X_p = q \times (q^{-1} \bmod p)$$
$$V_q = C^d \bmod q \quad X_q = p \times (p^{-1} \bmod q)$$
$$C^d \bmod n = (V_p X_p + V_q X_q) \bmod n$$

ii. Fermats Little Theorem (FLT) can speed up the calculation of Vp & Vq:

$$V_p = C^d \bmod p = C^{u \times (p-1) + v} \bmod p = C^v \bmod p$$

iii. Using CRT to speed up makes the calculation of C^d (mod n) vulnerable to different types of Side Channel Attacks, such as the Fault Injection Attack and the Timing Attack. In the Fault Injection attack, you can get a processor to reveal the values of the prime factors p and q just by deliberately causing the processor to miscalculate the value of either Vp or Vq (but not both).

5. RSA lacks forward secrecy

I. A communication link possesses forward secrecy (Perfect Forward Secrecy) if the content encryption keys used in a session cannot be inferred from a future compromise of one or both ends of the communication link.

II. attacker, who has managed to install a packet sniffer in the LAN to which the client is connected, patiently records all encrypted communications between the client and the server and someday he will be able to get hold of the servers private keys (Private keys may be leaked out anonymously by disloyal employees or through bugs in software.). If that were to happen, the attacker would be able to decrypt the session key that was sent encrypted by client to server.

III. solution to this problem with RSA lies in creating a session key without either party transmitting the key to the other party.

6. Chosen Ciphertext Attacks (CCA)

I. you use my public key (n, e) to encrypt a plaintext message M into the ciphertext C. C is picked up by attacker

II. attacker randomly chooses an integer s and constructs a new message by forming the product C”= s^e * C mod n.

III. attacker somehow lures me into decrypting C” and I send back the attacker

M’ = C’^d = (s^e * C)^d mod n = s^(e*d) * C^d mod n = s*M mod n

IV. attacker will now be able to recover the original message M by M = M’ *s^(-1) mod n

7. once attacker acquired both factors of a modulus, attacker can quickly calculate the private key that goes with the public key associated with the modulus.
8. mathematical attack: Trying to break RSA by developing an integer factorization solution for the moduli (figuring out the prime factors p and q of the modulus n)
   I. semiprime/ biprimes/ pq-numbers/ 2-almost primes: a number that is a product of two (not necessarily distinct) primes
   II. the security of the RSA algorithm is so critically dependent on the difficulty of finding the prime factors of a large number (mathematical techniques for solving the integer factorization problem)
9. The size of the key in the RSA algorithm typically refers to the size of the modulus integer in bits.
   I. The exponential relationship between what it takes to represent an integer in the memory of a computer and the value of that integer.
   II. RSA Laboratories recommends that the two primes that compose the modulus should be roughly of equal length. If use 1024-bit RSA encryption, modulus integer will have a 1024 bit presentation, need to generate two primes that are roughly 512 bits each.
   III. Doubling the size of the key (size of the modulus) will, increase the time required for public key operations (encryption or signature verification) by a factor of four and increase the time taken by private key operations (decryption and signing) by a factor of eight.
   IV. The public and the private keys are stored in particular formats specified by various protocols. Typically, the formats call for the keys to be stored using Base64 encoding so that they can be displayed using printable characters.
10. Using the best possible random number generators to create candidates for the primes that are needed and use a version of the RSA scheme that is resistant to the chosen ciphertext attacks, the security of RSA encryption depends critically on the difficulty of factoring large integers. As integer factorization algorithms have become more and more powerful over the years, RSA cryptography has had to rely on increasingly larger values for the integer modulus and, therefore, increasingly longer encryption keys. Computational overhead of RSA encryption/decryption goes up as the size of the modulus integer increases. This makes RSA inappropriate for encryption/decryption of actual message content for high data-rate communication links.

■If each of the two parties A and B has full confidence that a message received from the

other party is indeed authentic, the exchange of the secret session key for a symmetric-key based secure communication link can be carried out with a simple protocol as below:

1. A generates a public/private key pair {PUA,PRA} and transmits an unencrypted message to B consisting of PUA, A's identifier, IDA(which can be As IP address).
2. Upon receiving the message from A, B generates and stores a secret session key KS and responds to A with KS encrypted with PUA.
3. A decrypts the message received from B with PRA and retrieves KS.
4. A discards both PUA and PRA, and B discards PUA. A and B communicate confidentially KS.
5. this protocol is vulnerable to the man-in-the-middle attack by an adversary V who is able to intercept messages between A and B.
    I. When A sends the first unencrypted message consisting of PUA and IDA, V intercepts the message. (B never sees this initial message.)
    II. V generates its own public/private key pair {PUV,PRV} and transmits {PUV, IDA} to B.
    III. B received message and generates the secret key KS, encodes it with PUV, and sends it back to IDA
    IV. This transmission from B is intercepted by V and decoded. V encodes the KS with PUA and sends to A.
    V. A retrieves the secret key and starts communicating with B using the secret key.. V can now successfully eavesdrop on all communications between A and B.

■protocols for security that provides either one-sided or mutual authentication between two communicating parties usually involves Certificate Authorities

1. A certificate issued by a certificate authority (CA) is your public key signed by the CA's private key (CA having signed the certificate).
    I. There are three kinds of certificates, depending on the level of identity assurance and authentication
        i. Extended Validation (EV) certificates: At the highest level
        ii. Organization Validation (OV) certificates: At the next lower level
        iii. Domain Validation (DV) certificates: At the lowest level.
    II. certificate consists of the users public key, the identifier of the key owner, expiration date/time for the As public key

$$C_A = E(PR_{CA}, [T, ID_A, PU_A])$$

2. Usage
    I. When party A presents his/her certificate to party B, the latter can verify the legitimacy of the certificate by decrypting it with the CAs public key. Successful decryption authenticates both the certificate supplied by A and

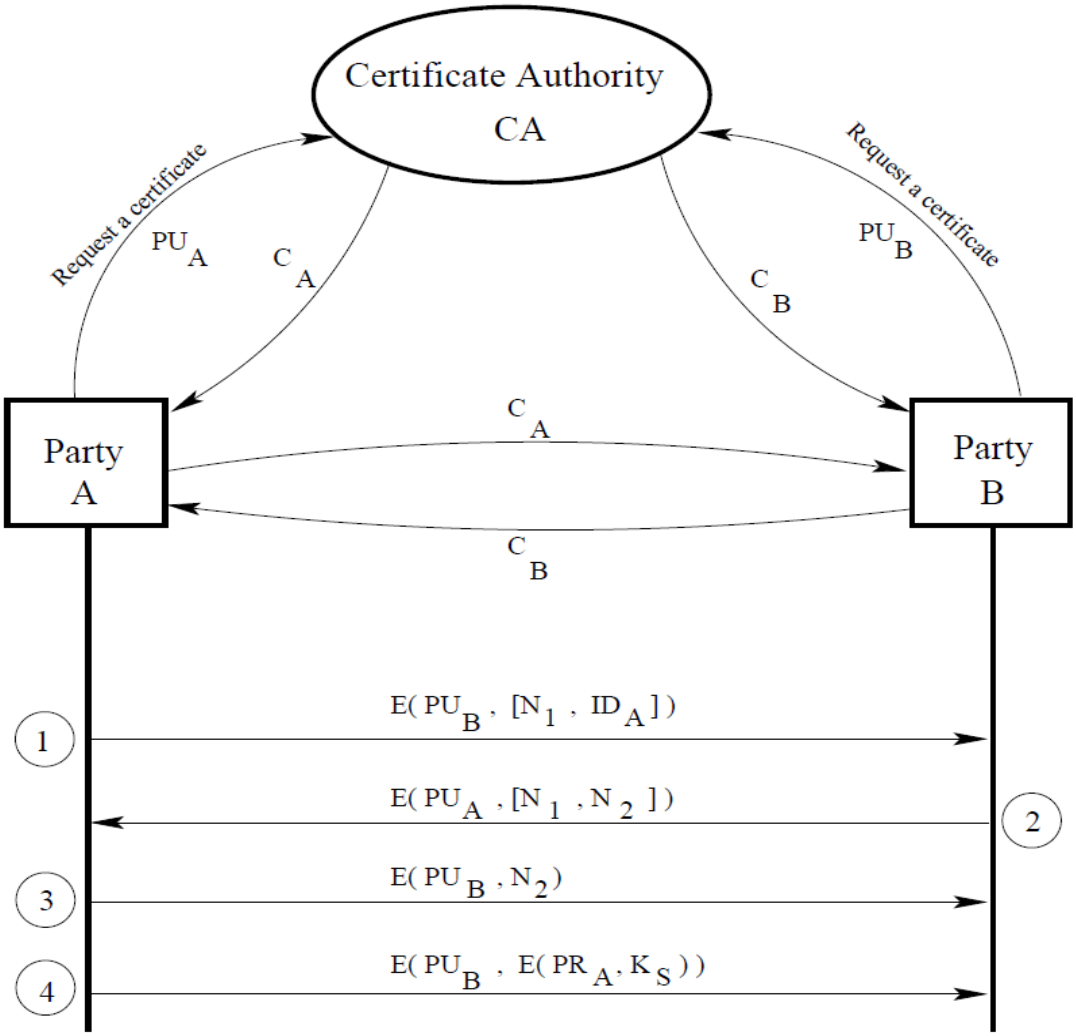
As public key.

II. Having established the certificates legitimacy, having authenticated A, and having acquired As public key, B responds back to A with its own certificate. A processes Bs certificate in the same manner as B processed As certificate.

III. Most of the business transactions in e-commerce utilize only one-way authentication. Before you upload your credit-card info to Amazon.com, your laptop must make certain that the website at the other end is truly Amazon.com. There is no need for Amazon.com to authenticate you or your laptop directly.

i. It is highly likely that a client will not possess a certificate

ii. while it is important for your laptop to authenticate Amazon.com, the company does not really care as to who you are as long as your credit-card information proves to be valid.



IV. Having acquired the public keys (and having cached them for future use), the two parties A and B then proceed to exchange a secret session key.

i. A uses PUB to encrypt a message that contains IDA and a nonce N1 as

a transaction identifier. A sends this encrypted message to B.

ii. B responds back with a message encrypted using PUA, the message containing A's nonce N1 and new nonce N2 from B to A. Since only B could decrypt the first message from A to B, the presence of the N1 in this response from B further assures A that the responding party is B

iii. A selects a secret session key KS and sends B. A encrypts KS with PRA (for B to authenticate the sender of the secret key) before further encrypting it PUB

iv. B decrypts the message first with PRB and recovers the secret key by applying another round of decryption using PUA.

3. CAs operate through a strict hierarchical organization in which the trust can only flow downwards.
   I. Root Cas: CAs at the top of the hierarchy
   II. Intermediate-Level CAs: CAs below the root
   III. There is a very practical reason for why Intermediate-Level CAs are needed: public keys for the Root CAs come pre-loaded with your computer (and also with the browsers). Situation that would arise should the private key of a Root CA become compromised for some reason. The only fix for that problem would be for you to update your software in order to replace the now defunct public key for the Root CA. But there must exist hundreds of millions of devices for which the software is rarely updated. You don't run into this problem when the private key of an Intermediate-Level CA is compromised. The affected certificates can simply be added to a Certificate Revocation List maintained by a higher-level CA. The affected CA can then proceed to issue fresh certificates to the affected parties.
   IV. consider a certificate issued by a CA that is not just below the root in the tree of CAs. Before your browser trusts such a certificate, it will verify the public key of the next higher level CA that validated the certificate your browser has received. This process is recursive until the root certificate that is pre-loaded in your computer is invoked. In order to save your browser from having to make repeated requests for the certificates as it goes up the tree of CAs, the webserver that sent you the certificate you are specifically interested in may send the whole bundle of higher level certificates also.
4. Public Key Infrastructure (PKI): The set of standards related to the creation, distribution, use, and revocation of digital certificates
   I. X.509 standard is based on a strict hierarchical organization of the CAs in which the trust can only flow downwards.
   II. The public keys of the root CAs are incorporated in your browser software

and other applications that require networking so that the root-level verification is not subject to network-based man-in-the-middle attacks. This also enables quick local authentication at the root level.

III. format of an X.509 certificate

i. Version Number: version of the X.509 standard to which the certificate corresponds.

ii. Serial Number: serial number assigned to a certificate by the CA.

iii. Signature Algorithm ID: name of digital signature algorithm used to sign certificate. signature itself is placed in the last field of certificate.

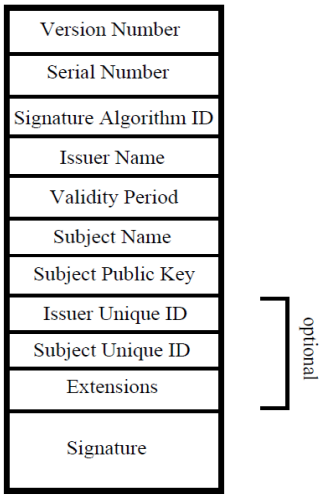
iv. Issuer Name: name of the Certificate Authority that issued this certificate.

v. Validity Period: time period during which the certificate is valid.

vi. Subject Name: individual/organization to which the certificate was issued.

vii. Subject Public Key: public key that is meant to be authenticated by certificate.

X.509 Certificate Format



viii. Issuer Unique Identifier: (optional) With this identifier, two or more different CAs can operate as logically a single CA. The Issuer Name field will be distinct for each such CA but they will share the same value for the Issuer Unique Identifier.

ix. Subject Unique Identifier: (optional) With this identifier, 2+ different certificate holders can act as a single logical entity. Each holder have a different value for the Subject Name field but they will share the same value for the Subject Unique Identifier field.

x. Extensions: (optional) This field allows a CA to add additional private information to a certificate.

xi. Signature: contains the digital signature by the issuing CA for the certificate. This signature is obtained by first computing a message digest of the rest of the fields with a hashing algorithm like SHA-1 and then encrypting it with the CAs private key. Authenticity of the contents of the certificate can be verified by using CAs public key to retrieve the message digest and then by comparing this digest with one computed from the rest of the fields.

xii. The digital representation of an X.509 certificate, described in RFC 5280, is created by first using ASN.1 representation to generate a byte

stream for the certificate and converting the bytestream into a printable form with Base64 encoding.

■Diffie-Hellman Key Exchange algorithm (ephemeral secret key agreement protocol)

1. Two parties A and B using this algorithm for creating a shared secret key first agree on a large prime number p and an element g of Z_p^* that generates a large-order cyclic subgroup of the multiplicative group Z_p^*. We want to choose for the DH protocol a g so that the order M is a large prime factor of p-1
   I. the order of a group: the cardinality of the group, meaning the number of elements in the group.
   II. The order of element (a) in a group is the smallest value t, such that $a^t \equiv a \circ a \circ \ldots (t\ times) \ldots \circ a = group\ identity\ element$, $\circ$ is group operator
   III. multiplicative group Z_p^* : the set {1, 2, …, p-1} constitutes a group with the group operator being modulo p multiplication, group order is p-1, identity element is 1.
   IV. A subset of Z_p^*forms a cyclic subgroup if the group operator continues to be modulo p multiplication and if all of the elements of the subgroup can be generated through the powers of one of the elements of the subgroup. All of the elements of the subset can be generated by $g^i\ mod\ p$ for i=0,1,2, … for some element g in subset
   V. Z_{17}^* is a cyclic group with g= 3. That is, if you compute $3^i\ mod\ 17\ for\ all\ i = 0,1,2 \ldots$, you will get the 16 numbers in the multiplicative group Z_{17}^*. If use 2 as a generator element, we get the cyclic subgroup {1,2,4,8,16,15,13,9} whose order is 8. All of the elements in this subgroup are given by 2^i mod 17 for all i=0,1,2…
   VI. Lagranges Theorem in Group Theory: if M is the order of a cyclic subgroup of Z_p^*, M will be a divisor of p-1.
   VII. within each order-M cyclic subgroup of Z_p^*, we have $g^M = 1$, if g is the generator for that subgroup. g is the primitive element of the cyclic subgroup generated by it. g is called the generator of the multiplicative subgroup that is generated by raising g to all possible power.
2. With n being the order of the generator g, the triple of numbers (p, g, n) is made public. The security of the DH protocol depends on the size of the cyclic subgroup. The triple of numbers (p, g, n) may be used for several runs of the protocol. These two numbers may even stay the same for a large number of users for a long period of time.
3. Algorithm: denote A's and B's private keys by XA and XB, public keys by YA and YB.
   I. A selects a random number XA, 1 <XA < n, for its private key. Chosen g can

generate the entire multiplicative group Z_p^*, the private key XA is chosen from the set {2, . . . , p-2}. Note that p-1 is excluded from the set since $g^{p-1} = 1\ by\ FLT$

II. A then calculates a public key integer YA and makes the public key YA available to B.

$$Y_A = g^{X_A} \bmod p$$

III. B selects a random number XB, 1 <XB < n, for its private key.

IV. B then calculates a public key integer YB and makes the public key YB available to A

$$Y_B = g^{X_B} \bmod p .$$

V. A calculates the secret key K from its private key XA and Bs public key YB

$$K = (Y_B)^{X_A} \bmod p$$

VI. B also calculates the shared secret key K from his/her private key XB and As public key YA

VII. secret key K is used only once.

$$K = (Y_A)^{X_B} \bmod p$$

$$\begin{aligned} K\ as\ calculated\ by\ A &= (Y_B)^{X_A} \bmod p \\ &= (g^{X_B} \bmod p)^{X_A} \bmod p \\ &= (g^{X_B})^{X_A} \bmod p \\ &= g^{X_B X_A} \bmod p \\ &= (g^{X_A})^{X_B} \bmod p \\ &= (g^{X_A} \bmod p)^{X_B} \bmod p \\ &= (Y_A)^{X_B} \bmod p \\ &= K\ as\ calculated\ by\ B \end{aligned}$$

4. Example: p=17, g=2, $Z_{17}^* = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16\}$

I. By calculating 2^I mod 17 for all i = 0, 1, 2, 3,…, the cyclic subgroup is {1, 2, 4, 8, 16, 15, 13, 9}

II. Since the size of this subgroup is 8, we have n = 8

III. party A chooses XA = 5 as a number between 1 and 8 as its private key, A's public key YA = 2^XA mod 17 = 2^5 mod 17 =15

IV. B chooses XB = 7 as a number between 1 and 8 as its private key. B's public key YB = 2^XB mod 17 = 2^7 mod 17 =9

V. secret session key as calculated by A: KA = YB^XA mod 17 = 9^5 mod 17=8

VI. secret session key as calculated by B: KB = YA^XB mod 17 = 15^7 mod 17=8

5. having access to the public keys for both A and B would still not be able to figure out the secret key K. It allows two parties A and B to create a shared secret K without either party having to send it directly to the other.

6. The security of the Diffie-Hellman algorithm is that whereas it is easy to compute

the powers of an integer in a finite field,

$$Y_A = g^{X_A} \bmod p$$

it is extremely hard to compute the discrete logarithms. To figure out the private keys XA or XB from a knowledge of all of the publicly available information {p, g, n,YA,YB}, the adversary have to carry out the discrete logarithm calculation

$$X_A = d\log_{g,p} Y_A$$

7. Computational Diffie-Hellman Assumption: The difficulty of determining the secret shared key K from the publicly available p, g, n,YA,andYB
8. ElGamal protocol: variant of the Diffie-Hellman protocol, in which A's public key remains fixed (and publicly available) over a long period of time. Party B encrypts his/her message M by calculating M*K mod p where K is the same as defined earlier. Party B can directly encrypt the message M without having to resort to a block cipher for content encryption. For reasons of computational efficiency, this works well only when M is small. The decryption by A consists of dividing the received ciphertext by K modulo p. This mechanism is useful in some implementations of anonymous client connections.
9. One of the most serious vulnerabilities of DH is to the man-in-the-middle attack, so use of the DH protocol should be preceded by sender authentication (use of the DH protocol should be preceded by sender Authentication). In authenticated DH, each party acquires a certificate for the other party. The DH public key that each party sends to the other party is digitally signed by the sender using the private key that corresponds to the public key on the senders certificate.
10. The basic weakness of DH lies in fact that a large number of servers use the same set of DH parameters. This dramatically reduces the cost of large-scale attacks,

■digitally signed a document: first calculated a hash of the document, then encrypted the hash with private key, and made this encrypted block available (as your signature) along with the document. When a party wants to verify that the document is authentic, they use your public key to extract the hash of the document from the encrypted block, and compare this hash with the hash their computer calculates directly from the document.

■Digital Signature Standard (DSS): based on ElGamal algorithm

1. Select a large prime p and then randomly select two numbers, g and X, less than p. You will make the numbers p and g publicly available and treat X as private key.
2. calculate your public key Y:

$$Y = g^X \ mod\ p$$

3. generate a one-time random number K such that 0 < K < p . 1 and gcd(K, p . 1) = 1. That is, each digital signature you create will be with a different K.
4. Let M be the integer that represents what you want to sign. Typically, M will be

the output of a hashing function applied to the document.

5. digital signature you construct for M will consist of two parts sig1 and sig2.
   I. $$sig_1 = g^K \mod p$$
   II. $$sig_2 = K^{-1} \times (M - X \times sig_1) \mod (p-1)$$
6. you have sent the message M along with your signature (sig1,sig2) to some recipient. Recipient can verify the authenticity of M by checking:
$$Y^{sig_1} \times sig_1^{sig_2} \equiv g^M \pmod p$$

■DISCRETE LOGARITHM PROBLEM

1. if an adversary can solve $g^s = k \mod p$ for s for given values of g and k, the Diffie-Hellman encryption will be broken.
2. One obvious way to solve the discrete logarithm problem is brute force by g^i for i = 0, 1, 2, ... If p requires an n bit representation, then the complexity is propotional to 2^n, grows exponentially with the size of p in bits.
3. slightly more efficient way to solve the discrete logarithm problem is by the baby-step giant-step method:
   I. baby steps: Compute, sort, and store the m elements g^0, g^1, g^2,…g^m in a table.
   II. Giant steps: Compute k/g^m and check to see if it is in the above table. If not, compute k/g^2m and check to see if it is in the table. If not, repeat until you find a j so that k/g^jm is in the table. k/g^jm = g^I implies s =jm + i. time complexity of this algorithm is O(p/m) and the memory requirement O(m). The product of the two is O(p) = O(2^n), which is still exponential in n, the size of p.
4. Pollard-p method
   I. based on the assumption that g can serve as the generator of a subgroup of prime order q within Zp. That means that the set {g0, g1, . . .} would form a subgroup within the set Zp.
   II. Let f be a random mapping function from a finite set A to itself. randomly selected ai∈A, sequence a0, a1, a2, . . ., will eventually cycle because A was assumed to be finite.
   III. $$f(x,u,v) = \begin{cases} (x^2, 2u, 2v), & if \ x \equiv 0 \pmod 3, \\ (kx, u, v+1), & if \ x \equiv 1 \pmod 3, \\ (gx, u+1, v), & if \ x \equiv 2 \pmod 3 \end{cases}$$
   IV. $$(x_0, u_0, v_0) = (1, 0, 0),$$
   $$(x_{i+1}, u_{i+1}, v_{i+1}) = f(x_i, u_i, v_i)$$
   V. $$x_i = g^{u_i} k^{v_i} \text{ for all } i \geq 0$$
   VI. Assume that we can find an x_i such that x_2i=x_i. When that happens,

g^u_2i * k^v_2i = g^u_i * k^v_i. Substituting this in k=g^s, we have g^u_2i * g^sv_2i = g^u_i * g^sv_i

$$s = \frac{u_{2i} - u_i}{v_i - v_{2i}} \bmod (q - 1)$$

VII. time complexity is O(2^(n/2)) if it takes n bits to represent the prime integer p.

■CERTIFICATES ISSUED BY A CA can BE FORGED

1. acquired some real certificates from a root CA and then proceeded to attach the CAs signature to a different public-key embedded in a digital document whose MD5 signature was the same as that in one of the legal certificates.
2. attacker break into the resellers account and created for himself a new user account with authorization to issue Comodo certificates. these certificates were forged because the attacker held the private keys corresponding to the public keys signed by the Comodos private key. This would have allowed the attacker to act like Google, Yahoo, Skype, etc.

■Elliptic curve cryptography (ECC):

1. computation of the RSA-based approach to public-key cryptography increases with the size of the keys. As algorithms for integer factorization have become more and more efficient, the RSA based methods have had to resort to longer and longer keys. Elliptic curve cryptography (ECC) can provide the same level and type of security as RSA (or Diffie-Hellman) but with much shorter keys
   I. with ECC, it takes one-sixth the computational effort to provide the same level of cryptographic security that you get with 1024-bit RSA.

Symmetric Encryption Key Size in bits	*RSA and Diffie-Hellman "Key" size in bits*	*ECC "Key" Size in bits*
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	512

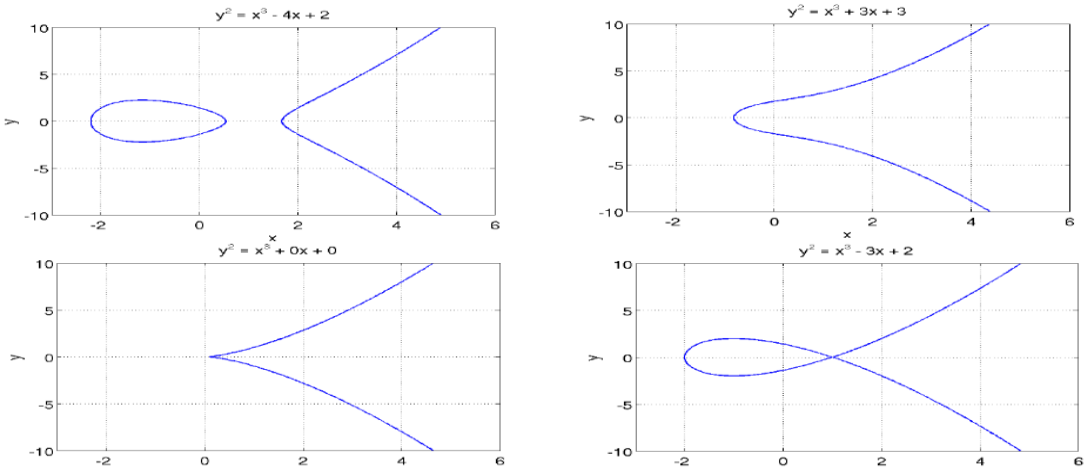
   II. computational overhead of both RSA and ECC grows as O(N^3) where N is the key length in bits, but it takes far less computational overhead to use ECC on account of the fact that you can get away with much shorter keys.
2. In ECDHE-RSA, RSA is used for certificate based authentication using the TLS/SSL protocol and ECDHE (Elliptic Curve Diffie-Hellman Ephemera) used for creating a one-time session key (ephemeral: when there is no authentication between two parties and they just want a session key on a one-time basis.)
3. ECC with AES, is used in game consoles to keep others from gaining direct access to binaries and for ensuring that the hardware only executes code. Authenticated
4. In PS3, the SELF files are signed with ECDSA (Elliptic Curve Digital Signature

Algorithm) algorithm so that the hardware only executes authenticated code.

5. Algorithm
   I. have a set E of points (xi, yi) in a plane. The set is very, very large but finite
   II. given two points P and Q in the set E, the group operator will allow us to calculate a third point R, also in the set E, P + Q = R.
   III. Given a point G∈E, an ordinary integer k, k×G represents the repeated addition G + G + . . . + G in which G makes k appearances, with the operator + being invoked k-1 times. Assume that the only way to recover k from k G is to try every possible repeated summation like G + G, G + G + G, G + G + G + . . . + G until the result equals what we have for k×G.
   IV. All of the assumptions we have made above are satisfied when the set E of points (xi,yi) is drawn from an elliptic curve.
   V. points on an elliptic curve can be shown to constitute a group
   VI. the amount of computational effort that it takes to add a point G to itself XA number of times is logarithmic in the size of XA.
      i. You add G to itself once and you get 2G. Next you add 2G to itself and you get 4G, followed by adding 4G to itself to get 8G, and so on. Since the attacker would not know the value of XA, he would not be able to take advantage of such exponentially jump
      ii. all these calculations are carried out modulo a prime p (in the most commonly used form of ECC). So, as you keep on adding G to itself, the size of what you get cannot serve as a guide to how many more times you must repeat that addition in order to get to the final value.
6. elliptic curves (Weierstrass Equation of characteristic 0): $y^2 = x^3 + ax + b$
   I. The characteristic of such a ring is the number of times you must add the multiplicative identity element in order to get the additive identity element.
   II. If adding multiplicative identity element to itself, no matter how many times, never gives us the additive identity element, we say the characteristic is 0.
   III. When a set is not of characteristic 0, there will exist an integer p such that p*n = 0 for all n. The value of p is then the characteristic of the integral domain. EX. Z9 is a ring of characteristic 9, 9*n are 0 for every value of the integer n.
   IV. When we say that the equation is of characteristic 0, we mean that the set of numbers that satisfy the equation. constitutes a ring of characteristic 0
   V. discriminant of a polynomial is the product of the squares of the differences of the polynomial roots. When the polynomial is $y^2 = x^3 + ax + b$, the discriminant reduces to $-16(4a^3 + 27b^2)$. This discriminant must not become zero for an elliptic curve polynomial possess three distinct roots.
   VI. elliptic curves in their standard form will be symmetric about the x-axis.

$$y \quad = \quad \pm \sqrt{x^3 \quad + \quad ax \quad + \quad b}$$

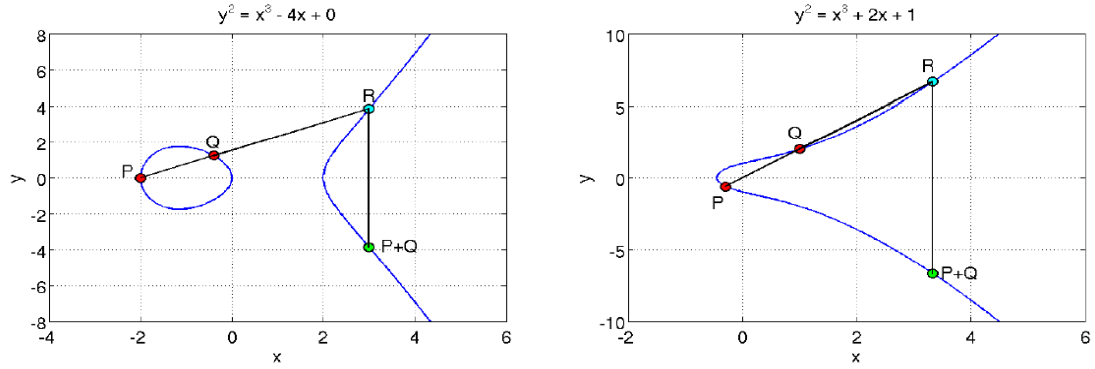
VII. Non-smooth curves are called singular: discriminant is zero, it is not safe to use singular curves for cryptography



VIII. To add a point P on an elliptic curve to another point Q on the same curve:

i. join P with Q with a straight line. The third point of the intersection of this straight line with the curve is R (If the third point of intersection does not exist, we say it is at infinity O, this can serve as the additive identity element for the group operator.)

ii. The mirror image of this point with respect to the x-coordinate is the point P + Q. (P+Q=-R => P+Q+R=O)



iii. P + O = P for any point on the curve: draw a line through P that is parallel to the y-axis, and that we then find the other point where this line intersects the curve. It follows from the next bullet that this other point will be the mirror reflection of P about the x-axis. That is, this other point will be at .P. When we reflect it with respect to the x-axis, we get back P.

iv. the additive inverse of a point P as its mirror reflection with respect to the x coordinate. So if Q on the curve is the mirror reflection of P on the curve, then Q = -P. The point of intersection of a point and its additive inverse will be the distinguished point O.

v. O + O = O, implying that O = -O.

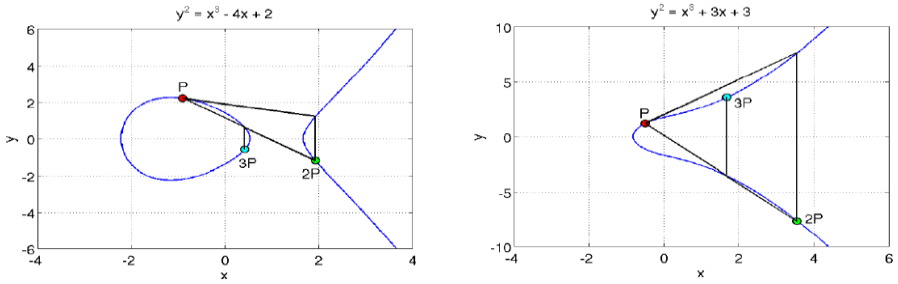
vi. when P and Q are each others mirror reflections with regard to the x-axis. The intersection of P and Q is at the distinguished point O, whose mirror reflection is also at O. Therefore, for such points, P + Q = O and Q = -P.

vii. What is the additive inverse of a point where the tangent is parallel to the y-axis? The additive inverse of such a point is the point itself. That is, if the tangent at P is parallel to the y-axis, then P + P = O.

viii. the operation P + P means that we must draw a tangent at P, find the intersection of the tangent with the curve, and then take the mirror reflection of the intersection.

ix. For an elliptic curve, we define the set of all points on the curve along with the distinguished point O by E(a, b). E(a, b) is an abelian group.

x. Calculating 2P and 3P for a given P



IX. The examples of the elliptic curves shown so far were for the field of real numbers. the coefficients a and b and the values taken on by the variables x and y all belong to the field of real numbers. These fields are of characteristic zero because no matter how many times you add the multiplicative identity element to itself, you' ll never get the additive identity element.

X. when we consider real numbers modulo 2, we have an underlying field of characteristic 2. When we consider real numbers modulo 3, we have an underlying field of characteristic 3. group law can also be defined when the underlying field is of characteristic 2 or 3. But now the elliptic curve becomes singular, points on the curve become isomorphic to either the multiplicative or the additive group over the underlying field itself, makes singular elliptic curves unsuitable for cryptography because they are easy to crack.

XI. A point on the curve is singular if $\frac{dy}{dx}=\frac{3x^2+a}{2y}$ is not properly defined and curve that contains a singular point is a singular curve. This would be the point where both the numerator and the denominator are zero. $(3x^2 + a = 0, 2y = 0)$.

i. When the underlying field is of characteristic 2, the equation 2y = 0 will always be satisfied.

ii. When the underlying field is of characteristic 3, dy/dx = a/2y, This curve becomes singular if we should choose a = 0

XII. equation of the straight line that runs through the points P and Q is

$$y = \alpha x + \beta$$

For a point (x, y) to lie at the intersection of the straight line and the elliptic curve E(a, b)

$$(\alpha x + \beta)^2 = x^3 + ax + b$$

cubic equation has three roots xP, xQ, and xR. Monic polynomial (coefficient of the highest power of x is 1) has property that the sum of their roots is equal to the negative of the coefficient of the second highest power. And P+Q=R

$$x^3 - \alpha^2 x^2 + (a - 2\alpha\beta)x + (b - \beta^2) = 0$$

$$x_P + x_Q + x_R = \alpha^2$$

$$x_R = \alpha^2 - x_P - x_Q$$

$$\begin{aligned} y_R &= \alpha x_R + \beta \\ &= \alpha x_R + (y_P - \alpha x_P) \\ &= \alpha(x_R - x_P) + y_P \end{aligned}$$

$$x_{P+Q} = \alpha^2 - x_P - x_Q$$

$$y_{P+Q} = \alpha(x_P - x_R) - y_P$$

XIII. ALGEBRAIC EXPRESSION FOR CALCULATING 2P FROM P

i. the slope of the tangent at point P

$$\alpha = \frac{3x_P^2 + a}{2y_P}$$

ii. if we draw a tangent at point P to an elliptic curve, the tangent will intersect the curve at a point R

$$\begin{aligned} x_R &= \alpha^2 - 2x_P \\ y_R &= \alpha(x_R - x_P) + y_P \end{aligned}$$

iii. Since the value of 2P is the reflection of the point R about the x-axis, the value of 2P is obtained by taking the negative of the y-coordinate

$$\begin{aligned} x_{2P} &= \alpha^2 - 2x_P \\ y_{2P} &= \alpha(x_P - x_R) - y_P \end{aligned}$$

7. The elliptic curve arithmetic over real numbers cannot be used for cryptography because calculations with real numbers are prone to round-off error. Cryptography requires error-free arithmetic. By restricting the values of a, b, x, y to some prime finite field Zp we obtain elliptic curves that are more appropriate for cryptography.

$$y^2 \equiv (x^3 + ax + b) \pmod p$$

points in Ep(a, b) are the set of coordinates (x, y), with x,y∈Zp, such that $y^2 = x^3 + ax + b$ with a, b∈Zp is satisfied modulo p and $4a^3 + 27b^2 \neq 0 \bmod p$.. the set of points in Ep(a, b) is no longer a curve, but a collection of discrete points in the (x, y) plane (in the Cartesian product Zp×Zp).

8. for prime finite field Zp, the value of p is its characteristic. Elliptic curves over prime

finite fields with p≤3, while admitting group law, are not suitable for cryptography.

9. Elliptic Curves can also be defined over a Galois Field GF(2^n). However, for such curves to be cryptographically secure, the value of n must be prime. a finite field of GF(2n) is of characteristic 2.
   I. elliptic curve equation to use when the underlying field GF(2n). b cannot be the additive identity element

$$y^2 + xy = x^3 + ax^2 + b, \qquad b \neq 0$$

   II. if b = 0, discriminant will become 0. However, even when the b = 0 condition is not satisfied, certain values of a and b may cause the discriminant to go to 0.
   III. changes in the behavior of the group operator:
      i. Given P = (x, y), the negative of this point is at -P = (x, -(x + y)).
      ii. Given two distinct points P =(xP,yP) and Q =(xQ,yQ), the addition of the two points, represented by(xP+Q,yP+Q) is

$$x_{P+Q} = \alpha^2 + \alpha - x_P - x_Q - a$$
$$y_{P+Q} = -\alpha(x_{P+Q} - x_P) - x_{P+Q} - y_P$$
$$\alpha = \frac{y_Q - y_P}{x_Q - x_P}$$

      iii. to calculate 2P from P

$$x_{2P} = \alpha^2 + \alpha - a - 2x_P$$
$$y_{2P} = -\alpha^2 - \alpha + a + (2 + \alpha)x_P - \alpha x_{2P} - y_P$$
$$\alpha = \frac{3x_P{}^2 + 2ax_P - y_P}{2y_P + x_P}$$

      iv. Hasses Theorem: if N is the number of points on Eq(a, b) when the curve is defined on a finite field Zq with q elements, then

$$|N - (q + 1)| \leq 2\sqrt{q}$$

      v. Since Galois field GF(2^n) contains 2^n elements, the order of E_2n(a, b) is equal to 2^n +1-t, $|t| \leq \sqrt{2^n}$
      vi. An elliptic curve over a Galois Field GF(2^n) is supersingular if 2|t. Supersingular curves defined over fields of characteristic 2 always have an odd number of points, including the distinguished point O. Supersingular curves are to be avoided for cryptography because they are vulnerable to the MOV attack.
10. RSA uses multiplication as its basic arithmetic operation (exponentiation is merely repeated multiplication), ECC uses the addition group operator as its basic arithmetic operation (multiplication is merely repeated addition).
11. with a proper choice for G, C = M×G, it can be extremely difficult to recover M from C even when an adversary knows the curve Eq(a, b) and the G used.

■Elliptic-Curve Diffie-Hellman (ECDH) algorithm: establishing a secret session key

between two parties.

1. A community of users wishing to engage in secure communications with ECC chooses the parameters q, a, and b for an elliptic-curve based group Eq(a, b), and a base point G∈Eq(a, b).
   I. To increase the level of difficulty in solving the discrete logarithm problem, we select for G a base point whose order is very large. The order of a point on the elliptic curve is the least number of times G must be added to itself so that we get the identity element O of the group Eq(a, b).
   II. G is the generator of a subgroup of Eq(a, b) whose elements are all given by G, 2G,3G, . . ., and, of course, the identity element O. For the size of the subgroup to equal the degree of the generator G, the value of n must be a prime when the underlying field is a Galois field GF(2^n).
   III. for DH, you choose a generator g∈ Z_p^* that results in a large-order cyclic subgroup {g, g^2, $g^3 \cdots 1$} of the multiplicative group Z_p^*. For ECDH, you choose a generator point G that leads to a large-sized cyclic subgroup {G, 2G, 3G, ...,O} of the group Eq(a, b).
2. A selects an integer XA to serve as his/her private key. A then generates YA=XA×G to serve as his/her public key. A makes YA publicly available
3. B designates an integer XB to serve as his/her private key. As was done by A, B also calculates his/her public key by YB=XB×G.
4. to create a shared secret key
   I. A calculates the shared session key: K=XA×YB
   II. B calculates the shared session key: K=XB×YA

$$\begin{aligned} K \text{ as calculated by } A &= X_A \times Y_B &&= (X_B \times X_A) \times G \\ &= X_A \times (X_B \times G) &&= X_B \times (X_A \times G) \\ &= (X_A \times X_B) \times G &&= X_B \times Y_A \\ & &&= K \text{ as calculated by } B \end{aligned}$$

■ELLIPTIC-CURVE DIGITAL SIGNATURE ALGORITHM (ECDSA): its use for code authentication in PlayStation3 game consoles. Code authentication means digital signature of a binary file is checked and verified before it is allowed to be run on a processor.

1. For a digital signature based on an elliptic curve defined over a prime finite field Zp, select a large prime p, choose the parameters a and b for the curve, and a generator point G of high order n (meaning that n×G = O for a large n).
2. randomly select X, $1 \leq X \leq n-1$, to serve as your private key.
3. calculate your public key Y=X×G. public key consists of a pair of numbers that are the coordinates of the point Y on the elliptic curve.
4. make p, a, b, G, n and Y publicly available and you will treat X as your private key.
5. Generate a one-time random number K such that 0 < K < n-1.
6. Let H be the hash the document you want to sign. Digital signature you construct for

H will consist of two parts, sig1 and sig2.

I. Construct sig1 by first calculating the elliptic curve point K×G and retaining only its x-coordinate modulo p.

$$sig_1 = (K \times G)_x \mod n$$

II. $$sig_2 = K^{-1} \cdot (H + X \cdot sig_1) \mod n$$

7. you sent your document along with its signature (sig1,sig2) to some recipient. Recipient can verify the authenticity of the document by
   I. calculating its hash H of the document
   II. calculating the numbers $w = sig_2^{-1} mod\ n,\ u1 = H \cdot w\ mod\ n,\ u2 = sig_1 \cdot w\ mod\ n$
   III. compute the point (x, y) = $u1 \times G + u2 \times Y$ on the curve,
   IV. authenticating the signature by checking whether sig1≡x (mod n) holds.
8. danger of using the same K for two different documents

$$sig_1 = (K \times G)_x \mod n$$
$$sig_2 = K^{-1} \cdot (H - X \cdot sig_1) \mod n$$
$$sig_1' = (K \times G)_x \mod n$$
$$sig_2' = K^{-1} \cdot (H' - X \cdot sig_1') \mod n$$

$$sig_2 - sig_2' = K^{-1}(H - H') \qquad sig_2 = K^{-1} \cdot (H - X \cdot sig_1)$$

■SECURITY OF ECC: RSA depends on difficulty of large-number factorization for its security, ECC depends on difficulty of the large number discrete logarithm calculation. This is referred to as Elliptic Curve Discrete Logarithm Problem (ECDLP).

1. In order to not fall prey to the MOV attack, elliptic curve and the base point chosen must satisfy MOV Condition.
   I. The order m of the base point G is the value of m such that m×G = O where O is the additive identity element of the group Eq(a, b)
   II. the order m of the base-point q^B - 1 should not divide for small B (B < 20).
2. To not be vulnerable to Weil descent attack. When using GF(2^n) finite fields, n must be a prime.
3. Elliptic curves for which the total number of points on the curve equals the number of elements in the underlying finite field are also considered cryptographically weak.

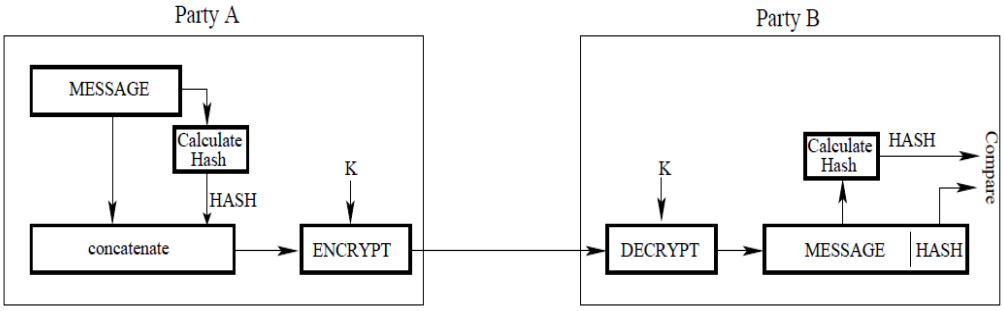
■ECC FOR DIGITAL RIGHTS MANAGEMENT(DRM)

1. DRM stands for technologies/algorithms that allow a content provider to impose limitations on the whos and hows of the usage of some media content made available by the provider.
2. DRM associated with the Windows Media (WM-DRM): WM-DRM Version 2 used elliptic curve cryptography for exchanging a secret session key between a users computer and the license server at the content providers location.

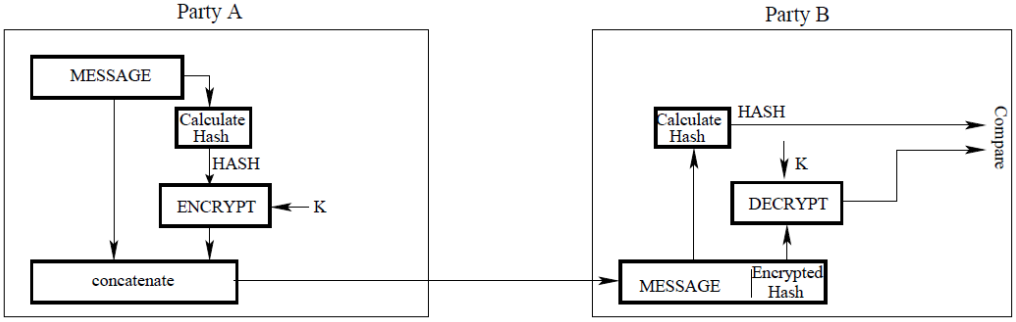
3. When you purchase media content from a Microsoft partner peddling their wares through the Window Media platform, you would need to download a license to be able play the content on your computer. Obtaining the license consists of your computer randomly generating a number n∈Zp for your computers private key. Your computer then multiplies the base point G with the private key to obtain the public key. Subsequently your computer can interact with the content providers license server
4. In order to ensure that only your computer can use the downloaded license, WM-DRM makes sure that you cannot access the private key that your computer generated for the ECC algorithm.

■hash function: takes a variable sized input message and produces a fixed-sized output (hashcode, hash value, or message digest)

1. Secure Hash Algorithm (SHA)
2. a message digest depends on all the bits in the input message, any alteration of the input message during transmission would cause its message digest to not match with its original message digest.
3. six different ways incorporate message hashing in a communication network to protect the hash value of a message.
   I. In the symmetric-key encryption based scheme, the message and its hashcode are concatenated together to form a composite message that is then encrypted and placed on the wire. The hashcode provides authentication and the encryption provides confidentiality.

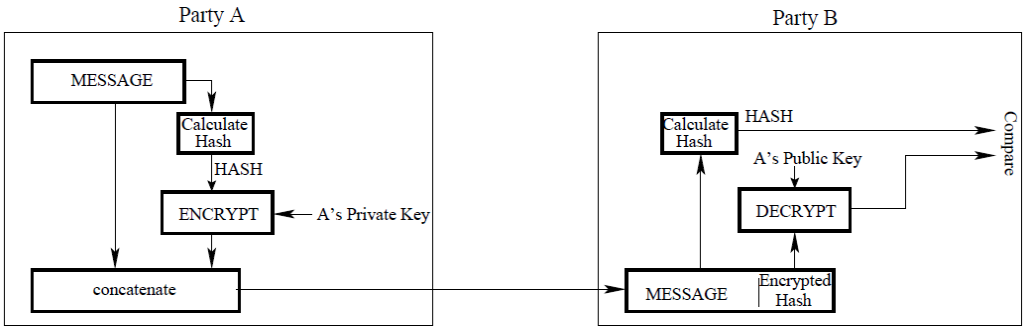


   II. Message Authentication Code (MAC): only the hashcode is encrypted. and the overall hash function as a keyed hash function. Confidentiality is not provided.

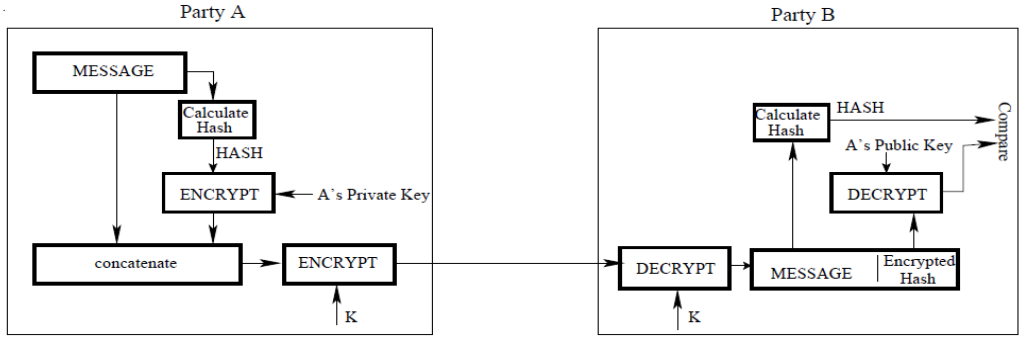


   III. In public-key encryption scheme: The hashcode of the message is

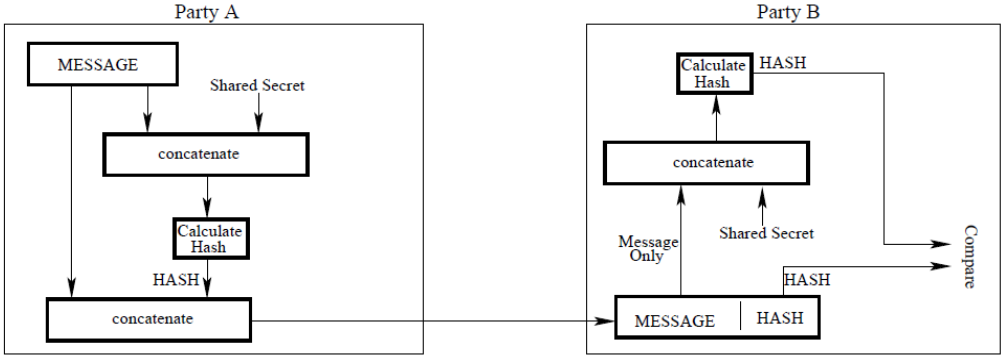
encrypted with the senders private key. The receiver can recover the hashcode with the senders public key. Confidentiality is not provided.



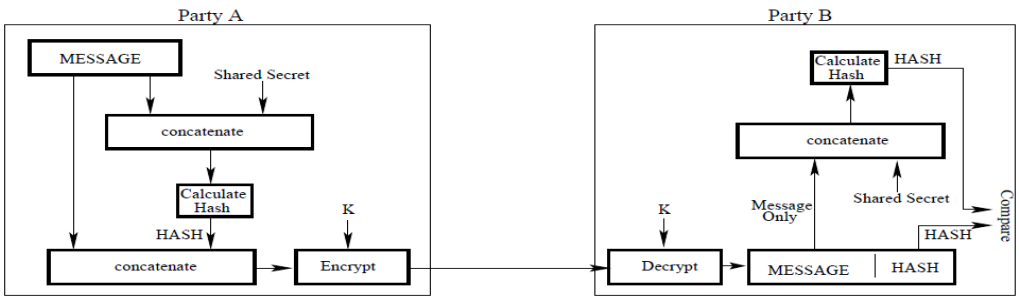
IV. add symmetric-key based confidentiality to the scheme



V. sender appends a secret string S (known to the receiver) to the message before computing its hashcode. Before checking the hashcode of the received message for its authentication, the receiver appends the same secret string S to the message. Nothing is encrypted. It would not be possible for anyone to alter such a message,



VI. add symmetric-key based confidentiality to the transmission between the sender and the receiver.



4. hash function is cryptographically secure if it satisfy two conditions :
    I. one-way property: It is computationally infeasible to find a message that corresponds to a given hashcode.
    II. strong collision resistance property: It is computationally infeasible to find two different messages that hash to the same hashcode value.
        i. weaker form of the strong collision resistance property: for a given message, there should not correspond another message with the same hashcode.
        ii. Collision resistance: likelihood that two different messages possessing certain basic structure so as to be meaningful will result in the same hashcode. Hash functions that are not collision resistant can fall prey to birthday attack.
        iii. If you use n bits to represent the hashcode, there are only 2^n distinct hashcode values.
5. Authentication: the message has not been altered, it is the authentic original message as produced by its author.
6. XOR hash algorithm:, XORing first n-bit block bit-by-bit with the second n-bit block, XORing the result with the next n-bit block, and so on.
    I. hashcode generated by the XOR algorithm can be useful as a data integrity check in the presence of completely random transmission errors: Every bit of the hashcode represents the parity at that bit position if we look across all of the n-bit blocks. Hashcode produced also known as longitudinal parity check.
    II. An adversary can modify the main message and add a suitable bit block before the hashcode so that the final hashcode remains unchanged.
    III. When you are hashing regular text and the character encoding is based on ASCII, the collision resistance property of the XOR algorithm suffers even more because the highest bit in every byte will be zero. This reduces the number of unique hashcode values available and increases the probability of collisions.
7. rotated-XOR algorithm (ROXR): To increase the space of distinct hashcode values available for the different messages, a variation on the basic XOR algorithm consists of performing a one-bit circular shift of the partial hashcode obtained after each n-bit block of the message is processed.
8. What is the value of k so that the pool contains at least one message whose hashcode is equal to h with probability 0.5?
    I. hashcode can take on N different but equiprobable values, message pool is size of k

II. pick two messages x and y randomly from the pool, probability that none of two messages has its hashcode equal to h is $(1-\frac{1}{N})^2$

III. probability that none of the messages in a pool of k messages has its hashcodes equal to h is $(1-\frac{1}{N})^k$

IV. the probability that at least one of the k messages has its hashcode equal to h is $1-(1-\frac{1}{N})^k \approx \frac{k}{N}$

V. when we use 64 bit hashcodes. N=2^64. K/2^64=0.5, k=2^63

9. Given a pool of k messages, each of which has a hashcode value from N possible such values

I. total number of ways,M1,in which we can construct a pool of k messages with no duplications in hashcode values

$$M_1 = N \times (N-1) \times \ldots \times (N-k+1) = \frac{N!}{(N-k)!}$$

II. total number of ways we can construct a pool of k messages without worrying about hashcode duplication

$$M_2 = N \times N \times \ldots \times N = N^k$$

III. the probability that this pool has no duplications in the hashcodes

$$\frac{M_1}{M_2} = \frac{N!}{(N-k)!N^k}$$

IV. probability that pool has at least one duplication in the hashcode values $1 - \frac{N!}{(N-k)!N^k}$ When k is small and N large, $\frac{k(k-1)}{2N}$

V. if the hashcode can take on a total N different values with equal probability, a pool of $\sqrt{}$N messages will contain at least one pair of messages with the same hashcode with a probability of 0.5..

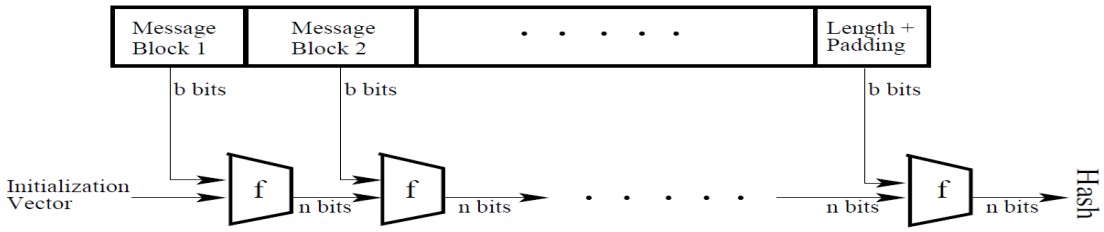
10. a hash function is cryptographically secure if it is computationally infeasible to find collisions

■BIRTHDAY ATTACK

1. Let the set of variations on the correct form of the contract be denoted {c1, c2, …ck} and the set of variations on the fraudulent contract by {f1, f2, …fk}. We need to figure out the probability that there exists at least one pair (ci, fj) so that h(ci) = h(fj).
2. probability that the hashcode h(c1) does not match the hashcode h(f1) is 1-1/N. Probability that h(c1) does not match h(f1) and h(f2) and . . .,h(fk) is $(1-\frac{1}{N})^k$

3. probability that there will NOT exist any hashcode matches between the two sets of contracts $(1-\frac{1}{N})^{k^2}$. probability that there will exist at least one match in hashcode values between the set of correct contracts and the set of fraudulent contracts 1-$(1-\frac{1}{N})^{k^2}$
4. What is the least value of k so that the above probability is 0.5? $\sqrt{N}$
5. For an n-bit hash coding algorithm that has no security flaws, the approximate value we obtained for k is the same in both cases. K=2^(n/2)

■Merkles structure



1. consists of L stages of processing: input message is partitioned into L number of bit blocks, each of size b bits.
2. final block also includes the total length of the message whose hash function is to be computed. This step enhances the security of the hash function since it places an additional constraint on the counterfeit messages.
3. Each stage of the structure takes two inputs, the b-bit block of the input message meant for that stage and the n-bit output of the previous stage.
4. For the n-bit input, the first stage is supplied with a special n-bit pattern called the Initialization Vector (IV).
5. compression function: processes the two inputs, one n bits long and the other b bits long, to produce an n bit output, usually,b > n

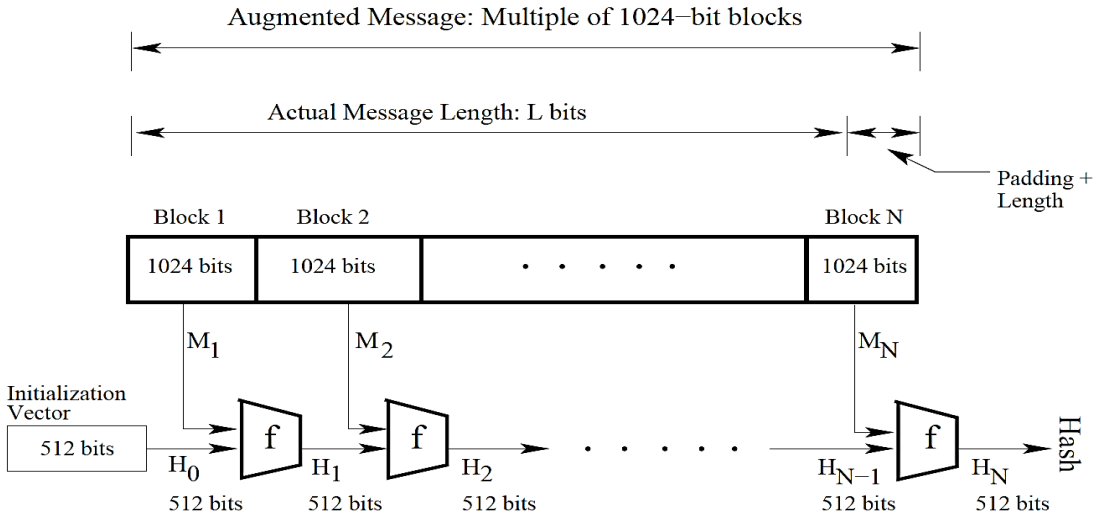
■SHA (Secure Hash Algorithm): a family of NIST-approved cryptographic hash functions

Algorithm	*Message Size* (bits)	*Block Size* (bits)	*Word Size* (bits)	*Message Digest Size* (bits)	*Security* (ideally) (bits)
SHA-1	$< 2^{64}$	512	32	160	80
SHA-256	$< 2^{64}$	512	32	256	128
SHA-384	$< 2^{128}$	1024	64	384	192
SHA-512	$< 2^{128}$	1024	64	512	256

1. Message Size: the upper bound on the size of the message that an algorithm can handle.
2. Message Digest Size: the size of the hashcode produced.
3. Security: how many messages have to be generated before two can be found

with the same hashcode with a probability of 0.5. For a secure hash algorithm that produces n-bit hashcodes, need to come up with 2^(n/2) messages to discover a collision with a probability of 0.5. So Security = Message Digest Size/2.

■SHA-512 Secure Hash Algorithm



1. Step

I. Pad the message so that its length is an multiple of 1024 bits(block size). The last 128 bits of the last block must be value of length of the message.

i. even if the original message were multiple of 1024, you still need to append another 1024-bit block at the end to make room for the 128-bit message length integer.

ii. padding consists of a 1-bit followed by the required number of 0-bits.

iii. length value in the trailing 128 bit positions is an unsigned integer with its most significant byte first.

II. Generate the message schedule: message schedule consists of 80 64-bit words {w0, w1,…w79}.

i. w0~w15, are the 1024-bit message block Mi.

ii. w16~w79 are obtained by applying permutation and mixing operations to the some of the previously generated words.

$$W_i = W_{i-16} +_{64} \sigma_0(W_{i-15}) +_{64} W_{i-7} +_{64} \sigma_1(W_{i-2})$$

$$\sigma_0(x) = ROTR^1(x) \oplus ROTR^8(x) \oplus SHR^7(x)$$

$$\sigma_1(x) = ROTR^{19}(x) \oplus ROTR^{61}(x) \oplus SHR^6(x)$$

$$ROTR^n(x) = circular\ right\ shift\ of\ the\ 64\ bit\ arg\ by\ n\ bits$$

$$SHR^n(x) = right\ shift\ of\ the\ 64\ bit\ arg\ by\ n\ bits\ with\ padding\ by\ zeros\ on\ the\ left$$

$$+_{64} = addition\ module\ 2^{64}$$

III. Apply round-based processing to each 1024-bit input message block: There

are 80 rounds to be carried out for each message block. The ith round is fed the 64-bit message schedule word w_i and a special constant K_i. For each round, first store the hash values calculated for PREVIOUS MESSAGE BLOCK in 64-bit variables denoted a, b, c, d, e, f, g, h. Permute the values stored in these eight variables and, with two of the variables, we mix in the message schedule word words[i] and a round constant K[i].

i. Initialize Hash Buffer with Initialization Vector: represent the hash buffer by 8 64-bit registers, a, b, c, d, e, f, g, h, initialized by the first 64 bits of the fractional parts of the square-roots of the first eight primes.

ii. constants Ki is the first 64 bits of the fractional parts of the cube roots of the first eighty prime numbers.

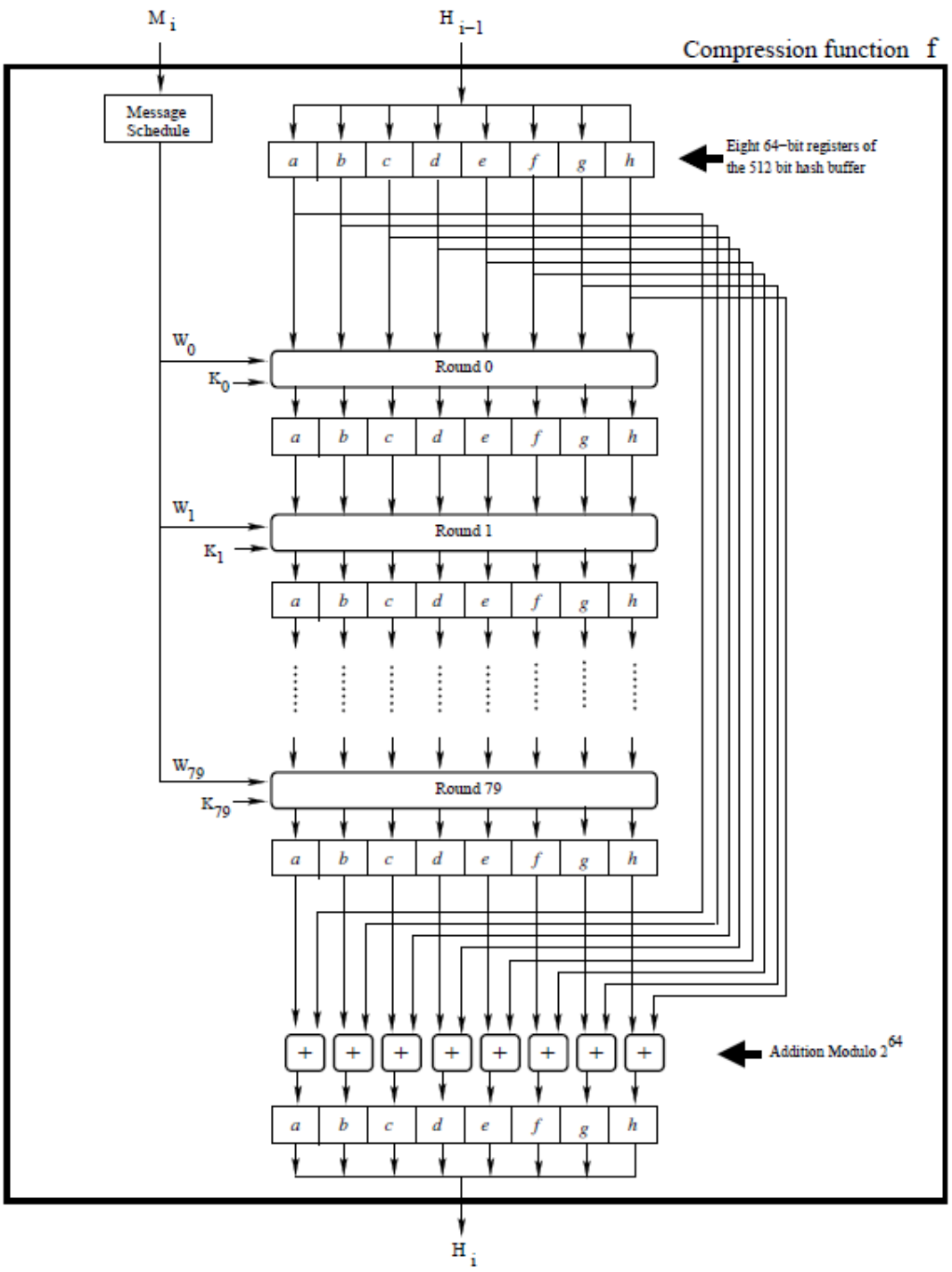
iii. Round function: consists of a sequence of transpositions and substitutions, all designed to diffuse to the maximum extent possible the content of the input message block.

1. the module f for processing the message block Mi has two inputs: the current contents of the 512-bit hash buffer and the 1024-bit message block.

$$
\begin{array}{lcl}
h = g & & T_1 = h +_{64} Ch(e,f,g) +_{64} \sum e +_{64} W_i +_{64} K_i \\
g = f & & T_2 = \sum a +_{64} Maj(a,b,c) \\
f = e & & \\
e = d +_{64} T_1 & & Ch(e,f,g) = (e\ AND\ f) \oplus (NOT\ e\ AND\ g) \\
d = c & & Maj(a,b,c) = (a\ AND\ b) \oplus (a\ AND\ c) \oplus (b\ AND\ c) \\
c = b & & \\
b = a & & \sum a = ROTR^{28}(a) \oplus ROTR^{34}(a) \oplus ROTR^{39}(a) \\
a = T_1 +_{64} T_2 & & \sum e = ROTR^{14}(e) \oplus ROTR^{18}(e) \oplus ROTR^{41}(e)
\end{array}
$$

2. Maj(): equal to the bit 1, when a majority of its arguments (meaning two out of three) are true.
3. Ch():at the bit level, if arg1 then arg2 else arg3
4. output of the 80th round is added to the content of the hash buffer at the beginning of the round-based processing. This addition is performed separately on each 64-bit word of the output of the 80th modulo 2^64. (addition is carried out separately for each of the eight registers of the hash buffer modulo 2^64)

IV. We update the hash values calculated for the PREVIOUS message block by adding to it the values in the temporary variables a, b, c, d, e, f, g, h.



■message authentication code (MAC): also known as a cryptographic checksum and as an authentication tag.

1. Sending a message along with an unencrypted hash provides zero security against forgery.
2. HMAC: A MAC produced by appending a secret key to the message and then hashing the composite message. The resulting hashcode is one way to generate the MAC of a message.

■Hashing finds extensive use in crypto currency algorithms.

1. Hashing algorithms are used
    I. proof-of-work calculations for mining a new coin: A node in crypto coin network establish ownership of a coin by taking a hash of all the relevant parameters related to the coin and encrypting the hash with its private key.
    II. authenticate ownership of coins;
    III. establish a protocol for transferring the ownership from one node to another in a cryptocoin network: Transferring the ownership of a coin from one node to another in a coin network established by
        i. the receiving node sending its public key to the selling node
        ii. the selling node append that public key to the coin, take a hash of the resulting structure, digitally encrypt the hash with its private key, and send the resulting object to the buyer.
    IV. create a protection against double spending (no node should be able to sell the same coin to multiple other nodes in the network.)
2. a crypto currency is based on the following basic notions
    I. Creating rare software objects through proof of work calculations. These objects becomes the coins of a virtual currency.
        i. For a software object to be rare, it must satisfy:
            1. difficult to produce
            2. easy to verify that it was difficult to produce the object.
        ii. use hash function to satisfy the requirements of rare:
            1. require rare software object to possess a hash value that has particular pattern, large computational effort is needed to discover a message whose hash correspond to that pattern.
            2. After you found such message, anyone can verify your discovery.
        iii. use an N-bit hash function, rareness is that the hash value be smaller than some integer t. The probability that you will discover a message whose hash is t / 2^N. POW_DIFFICULTY_LEVEL=252: a message hash would be considered acceptable for discovering a new coin whose value is less than 2^252
        iv. Another criterion for rareness could be requiring a certain designated number of the least significant bits of the hash value be all zeros.
    II. Public key cryptography for digital signatures
        i. A human users identity (ID) in the virtual world can be the hash of his/her public key. To verify an ID, hash the public key of the individual and see if it yields the same value as the ID under examination.
        ii. establish ownership of a coin: take hash of the coin, encrypt it with his/her private key, and append the signature to the coin.

III. transaction that comes with a verifiable proof of ownership transfer from the seller of a coin to its buyer.
    i. Seller ask buyer for his public key
    ii. Seller incorporate buyer's public key as an additional field in the coin and hash the augmented coin
    iii. Seller encrypt the coin the augmented coin with his private key
IV. a block as a set of transactions
    i. After a node has accumulated a certain designated number of outgoing transactions, it packages them into a block. a block is broadcast to the entire network.
    ii. How many transactions to pack into a block is a decision that each node must make for itself
        1. small number: its acceptance by the network would bring to an end for large number of ongoing mining runs at other nodes.
        2. Large number: receive a new block from some other node that increases the length of the blockchain that you are currently using in the coin mining algorithm
    iii. After a block is accepted by the network, the nodes in the network have no choice but to use that block as a genesis string for discovering the next coin.
    iv. authenticity of a block
        1. every new block has embedded in it a signed hash of the previous block whose hash was used as a genesis string for the new coin mined that are in the transactions in the new block
        2. every block has a unique identifier (BlockID) associated with it. That makes it a simple matter to verify that the new block is a valid construction from the previous block
V. blockchain: used to implicitly embed in each block a hash of all the previous transactions carried out so far. the length of the blockchain is equal to the number of transactions in the block
    i. This provides security against double spending and makes it practically possible to use crypto currencies in a manner similar to real currencies.

3. crypto currencies requires an implementation that must either be multithreaded or one that is based on multiprocessing.
    I. search for a new coin is a completely random process in which the time taken to discover the next coin cannot be predicted in advance.
    II. ongoing search for a new coin must be abandoned immediately when a

block of transactions is received from the network if the received block has a blockchain that is longer than what is being used in the current search, or, for a given blockchain length, if the received block has a POW difficultly that is greater than what is being used in the current search.

■TCP/IP protocol stack 7-layer model (OSI (Open Systems Interconnection) model)

1. Physical Layer (Ethernet 802.3, WiFi 802.11, USB, Bluetooth,
2. Data Link Layer (MAC,PPP, SLIP, ATM)
    I. Media Access Control (MAC) protocol: provides the addressing mechanism for data packets to be routed to a particular machine in a LAN (Local Area Network). Uses CSMA/CD (Carrier Sense Multiple Access with Collision Detection) protocol, to decide when the machines connected to the same communication medium, such as a LAN, should communicate.
3. Network Layer (IPv4, IPv6, ICMP, IPSec, IGMP)
    I. take care of network addressing: When a protocol in this layer receives a byte stream (a datagram or a packet) from an upper layer, it attaches a header with that byte stream that tells the protocols in the lower layers as to where the data is supposed to go in the internet.
    II. traffic control:
    III. IP (Internet Protocol): deals with routing packets of data from one computer to another or from one router to another. IP protocol is responsible for fragmenting a descending datagram at the sending end and reassembling the packets into what would become an ascending datagram at the receiving end.
    IV. ICMP (Internet Control Message Protocol): used for error/status messages in computer networks
        i. -Announce Network Errors: When a host or a portion of the net work becomes unreachable, an ICMP message is sent back to the sender.
        ii. Announce Network Congestion: If rate at which a router can transmit packets is slower than rate at which it receives, the routers buffers will begin to fill up. To slow down the incoming packets, the router may send the message ICMP Source Quench back to the sender.
        iii. Assist Troubleshooting: ICMP Echo messages are used by the popular ping utility to determine if a remote host is alive, for measuring round-trip propagation time to the remote host, and for determining the fraction of Echo packets lost en-route.
        iv. Announce Timeouts: When a packets TTL (Time To Live) drops to zero, the router discarding the packet sends an ICMP time exceeded message back to the sender announcing this fact.
    V. IGMP (Internet Group Management Protocol.): used for multicasting on the

internet. An IGMP header includes the IP addresses of the subscribers. So by examining an IGMP header, an enroute router can decide whether it is necessary to send copies of packet to multiple destinations, or whether just one packet can be sent to the next router.

VI. IPv4 32-bit, IPv6 128-bit

VII. IP Header: header added by the Network Layer,, contains information as to which higher level protocol the packet came from, the address of the source host, the address of the destination host, etc.

i. IP Header format for Version 4 of the IP protocol: field Identification, Flags, and fragment offset allow datagram descends from the upper TCP layer to be fragmented into IP packets and for the receiving endpoint to assemble packets back into a datagram for TCP layer.

1. Version field (4 bits wide): version of the IP protocol.
2. IHL field (4 bits wide) Internet Header Length: length of the IP header in 32-bit words. The minimum value for this field is 5 (shortest IP header consists of 20 bytes.)
3. DiffServ field (8 bits wide)
    I. Differentiated Service (DS): most significant 6 bits of DiffServ, expedited transmission of streaming data, such as video and voice, through the network routers and switches.
    II. Explicit Congestion Notification (ECN): The least significant 2 bits, receiving end-point of a communication link to notify sending endpoint about end-to-end traffic congestion
4. Total Length field (16 bits wide): size of the packet in bytes, including header and data. Minimum value for this field is 576.
5. Identification field (16 bits wide): assigned by sender to help receiver with the assembly of the received IP packets into larger datagrams expected by the upper TCP layer.
6. Flags field (3 bits wide):
    I. first bit is reserved and must be set to 0.
    II. Second bit is 0 means that the IP packet being pushed into the lower layer is a fragment of the datagram received from the TCP layer.
    III. third bit is 0 means the last fragment for the TCP datagram received from the upper layer; when set to 1 means more fragments are coming.
7. Fragment Offset field (13 bits wide): indicates where in the datagram this fragment belongs. The fragment offset is measured

in units of 8 bytes. This field is 0 for the first fragment.

8. Time To Live field (8 bits wide): determines how long the packet can live in the internet. Each time a packet passes through a router, its TTL is decremented by one.
9. Protocol field (8 bits wide): integer value that identifies higher-level protocol that generated the data portion of this packet.
10. Header Checksum field (16 bits wide): Since TTL varies each time a packet passes through a router, this field is recomputed at each routing point. Checksum is calculated by dividing the header into 16-bit words and then adding the words together. This provides a basic protection against corruption during transmission.
11. Source Address field (32 bits wide): IP address of the source.
12. Destination Address field (32 bits wide): IP address of destination
13. Options field: used to associate handling restrictions with a packet for enforcing security, record actual route taken from source to destination, to mark a packet with a timestamp, etc.
14. Padding field: ensure IP header ends on a 32-bit boundary.

4. Transport Layer (TCP, UDP)
   I. TCP (Transmission Control Protocol): ensuring that the data packets are delivered in a reliable manner from one computer to another.
      i. provide reliable exchange of data between two endpoints: sending endpoint receive an acknowledgment message from the receiving endpoint for each transmission to make sure that data actually arrived at receiving endpoint
      ii. provide mechanisms for congestion control: sending TCP to ramp up or ramp down the rate at which it sends out information in response to the ability of the receiving TCP to keep up with the traffic.
   II. UDP (User Datagram Protocol): used for quickly checking on the status of hosts and routers in the internet, for the transmission of error messages to the upstream hosts and routers in a communication link, fetching snippets of information from other hosts and routers, etc. Since UDP does not engage in elaborate handshaking and acknowledgments, it is a faster protocol to the overall efficiency with which the internet operates.
   III. on the transmit side, as packet descends down the protocol stack, starting with the transport layer, each layer adds its own header to the packet.
   IV. on the receive side, as each packet ascends up the protocol stack, each layer strips off the header corresponding to that layer and takes appropriate action vis-a-vis the packet before sending it up to the next higher layer.

5. Session Layer (TLS/SSL, NetBIOS, SOCKS, RPC, RMI)
    I. a session may consist of a single request from a client for some data from a server, or, multiple back-and-forth exchanges to data between two endpoints of a communication link.
    II. When security is an issue, these data transfers must be encrypted.
    III. TLS (Transport Layer Security), SSL (Secure Socket Layer)
6. Presentation Layer (MIME, XDR): translate, encode, compress, and apply other transformations to the data, condition it appropriately for processing by the protocols in the lower layers on the stack.
    I. MIME (Multipurpose Internet Mail Extensions.): all email is transmitted using the SMTP protocol in the Application Layer through the MIME protocol in the Presentation Layer.
    II. XDR (Extensible Data Representation): used for safe transfer of data between computers.
7. Application Layer (HTTP, HTTPS, FTP, SMTP, SSH, SMB, POP3, DNS, NFS):
    I. HTTP (HyperText Transport Protocol): protocols used for requesting and delivering web pages.
    II. SMTP (Simple Mail Transfer Protocol.)
    III. SMB (Samba) protocol: provide support for cross-platform (Microsoft Windows, Mac OS X, and other Unix systems) sharing of files and printers.

■TCP protocol is connection-oriented, IP protocol is a connectionless.

■Through handshaking and acknowledgments, TCP (Transmission Control Protocol) provides a reliable communication link between two hosts on the internet.

1. TCP connection is full-duplex: TCP connection simultaneously supports two independent byte streams, one for each direction of a communication link.
2. TCP provides flow control mechanism and congestion control mechanism.
    I. Flow control: receivers TCP is able to control the size of the segment dispatched by the senders TCP, accomplishes by putting to use the Window field of an acknowledgment packet
    II. Congestion control: sender TCP can measure traffic congestion through either the non-arrival of an expected ACK packet or by the arrival of three identical ACK packets consecutively, varies the rate at which it places the packets on the wire
3. header of a TCP segment
    I. Source Port field (16 bits wide): port that is the source of this TCP segment.
    II. Destination Port field (16 bits wide): port of the remote machine that is the final destination of this TCP segment.
    III. Sequence Number field (32-bit) and Acknowledgment Number field (32-

bit)

i. a TCP connection is in the process of being set up: When a host A first wants to establish a TCP connection with a remote host B, the two hosts A and B must engage 3-way handshake:
   1. A sends to B a SYN packet:
      i. initial sequence number (ISN): Sequence Number in this TCP packet is a randomly generated number M.
      ii. ISN generator: random number generator used for generating ISN
   2. remote host B send back to A a SYN/ACK packet containing
      iii. the number M + 1 in B's Acknowledgment Number field
      iv. another randomly generated number N in Sequence Number field
   3. A respond with an ACK packet containing N+1 in Acknowledgment Number field

ii. an already-established TCP connection is exchanging data: Sequence Number and the Acknowledgment Number fields are used to keep track of the byte count in the data streams that are exchanged between the two parties:
   1. Each endpoint in a TCP communication link associates a byte count with the first byte of outgoing bytes in each TCP segment.
   2. byte-count index is added to the initially sent ISN and placed in the Sequence Number field for an outgoing TCP packet. EX. A sends 10000 bytes to B by 100 segments, 1st segment contains 0 plus ISN in sequence number field. 2nd segment contains 100 plus ISN in sequence number field.
   3. When B receives these TCP segments, the Acknowledgment Number field of B's ACK packets contains the index it expects to see in the Sequence Number field of the next TCP segment it hopes to receive from A.

IV. Data Offset field (4 bits wide): number of 32-words in the TCP header.

V. Reserved field (6 bits wide): reserved for future Until its value must be 0.

VI. Control Bits field / flags (6 bits wide)

i. 1st flag bit URG: when set means URGENT data. when a TCP header has its URG bit set, that means that the receiving TCP engine should temporarily suspend accumulating the byte stream that it might be in the middle of and give higher priority to the urgent data.

ii. 2nd flag bit ACK: when set means acknowledgment. (ACK packet)

iii. 3rd flag bit PSH: when set means that we want the TCP segment to be put on the wire immediately. Ordinarily, TCP waits for its input buffer

to fill up before forming a TCP segment.

iv. 4th flag bit RST: when set means sender wants to reset connection.

v. 5th flag bit SYN: set means synchronization of sequence numbers.

vi. 6th flag bit FIN: set means sender wants to terminate connection.

VII. Window field (16 bits wide): maximum number of data bytes the receivers TCP is willing to accept from the senders TCP in a single TCP segment. Window field used by the sender TCP is set by the receiving TCP, the CWND field when used is set by the sender

VIII. Checksum field (16 bits wide): computed by adding all 16-bit words in a 12-byte pseudo header. If the data contains an odd number of bytes, a padding consisting of a zero byte is appended to the data.

i. pseudo-header and padding are not transmitted with TCP segment. While computing the checksum, the checksum field itself is replaced with zeros. The carry bits generated by the addition are added to the 16-bit sum. The checksum itself is the ones complement of the sum.

ii. At the receiving end, the pseudo-header is constructed from the overall length of the received TCP segment, the source IP address from the encapsulating IP header, and the destination IP address as assigned to the communications interface through which the segment was received.

iii. pseudo-header for the IPv4 protocol:

1. 4 bytes for the source IP address
2. 4 bytes for the destination IP address
3. 1 byte of zero bits,
4. 1 byte whose value represents the protocol for which the checksum is being carried out. EX 6 for TCP.
5. 2 bytes for the length of the TCP segment, including both the TCP header and the data

iv. checksum gives us an end-to-end verification from the sending TCP to the receiving TCP that the TCP segment was delivered to its intended destination.

IX. Urgent Pointer field (16 bits wide): value stored in the Urgent Pointer field is the offset from the value stored in the Sequence Number field where the urgent data ends. One can use urgent data TCP segments to abort an application at a remote site that may be in middle of a long data transfer from the sending end.

X. Options field is of variable size. If any optional header are included, their total length must be a multiple of a 32-bit word.