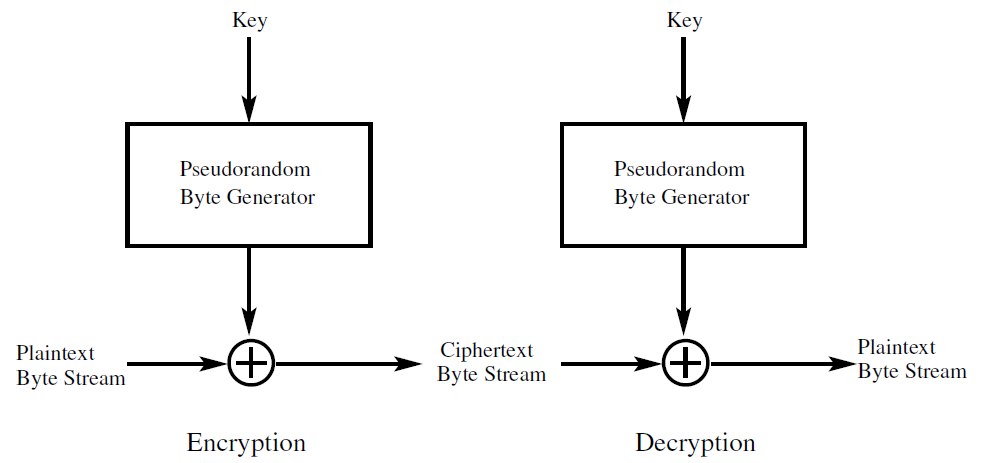
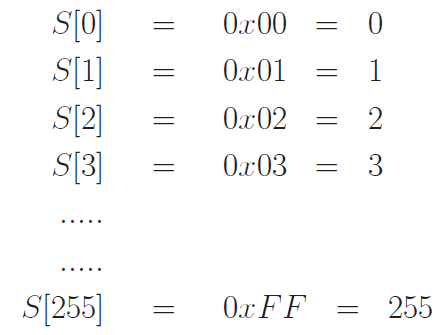
▓STREAM CIPHERS: a typical stream cipher encrypts plaintext one byte at a time.

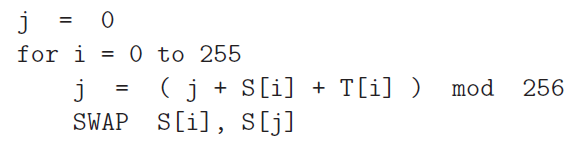
1. A simple stream cipher: as each byte of the plaintext becomes available, you XOR it with a byte of the pseudorandom byte stream. The output byte is what is transmitted to the destination.



1. The main processing step in a true stream cipher is the generation of a stream of pseudorandom bytes that depend on the encryption key.
2. 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.
3. 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.
4. 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

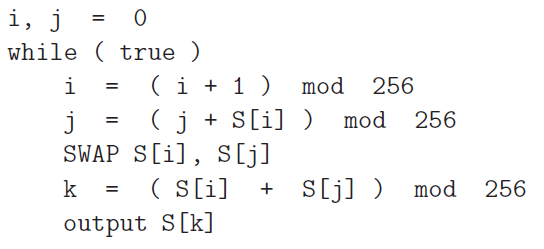
▓RC4 is a variable key length stream cipher with byte-oriented operations.

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.
   1. state vector S is initialized with entries from 0 to 255 in ascending order.
   2. 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.
   3. Key Scheduling Algorithm (KSA): use the 256-element vector T to produce the initial permutation of S.



* 1. There is no further use for the temporary vector T and encryption key after the state vector S is initialized as described above.
  2. 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.

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



* 1. the state of the pseudorandom number generator changes dynamically as the the numbers are being generated.
  2. 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.
  3. pseudorandom sequence of bytes generated by the above algorithm is also known as the keystream.

1. 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.
2. 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.
3. 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

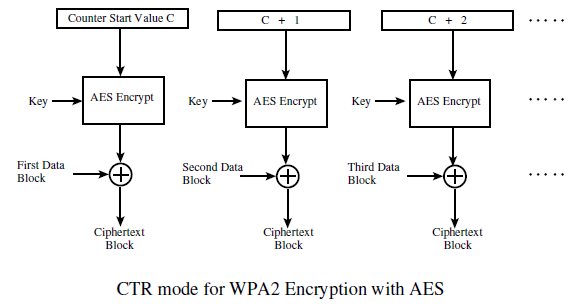


This can create a backdoor to extracting the plaintext stream from the ciphertext stream. All you have to do is to XOR the ciphertext in each packet with the ciphertext stream in a packet in which a reasonably large number of bytes are set to 0.

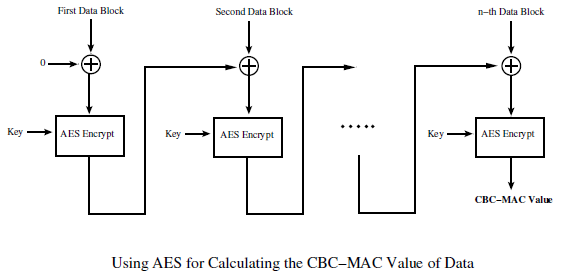
1. The RC4 key for each packet is a simple concatenation of a 24-bit Initialization Vector (IV) and the root key (APs security code). While the root key remains fixed over all the packets, you increment the value of IV from one packet to the next. Since the IV is sent in plaintext, anyone with a packet sniffer can directly see the first three bytes of the RC4 key used for a packet.
2. 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.
3. The RC4 key for a packet is then used to encrypt the data followed by ICV value
4. 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.
5. TKIP for Temporal Key Integrity Protocol:
   1. WPA uses a 48-bit Initialization Vector to enhance security
   2. 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
   3. 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
6. 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

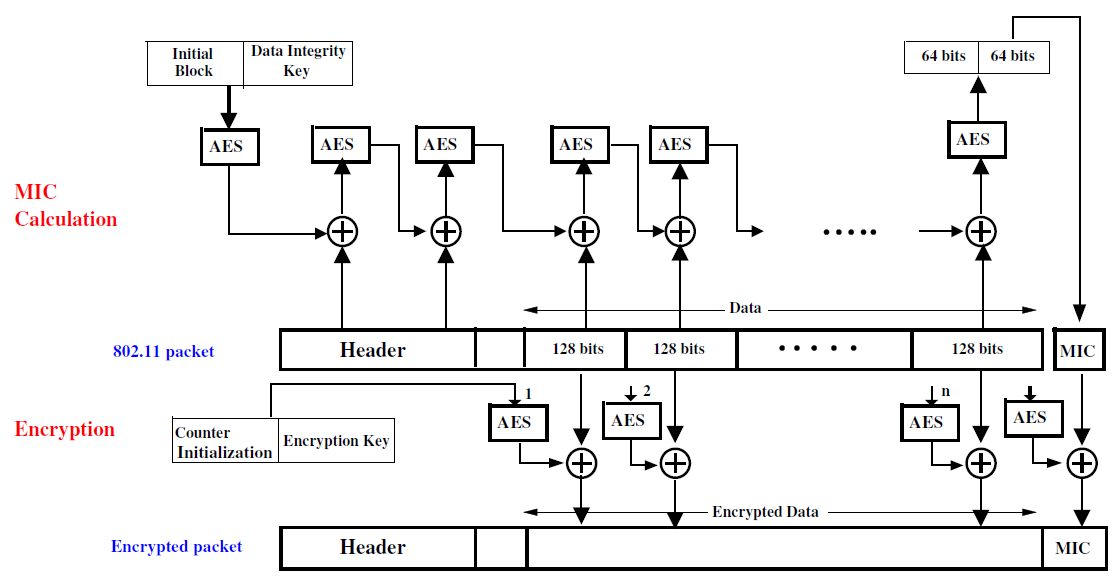
1. In addition to data encryption, the WiFi protocols also provide user authentication services. These services determine how a client (a laptop, smartphone, etc) would be allowed to join the WLAN.
2. All three WiFi security protocols allow for authentication to be carried out with a Pre-Shared Key (PSK). 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:
   1. a supplicant (a client) that wishes to join a WLAN
   2. an authenticator: an AP
   3. 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.



1. Calculation of CBC-MAC in WPA2.

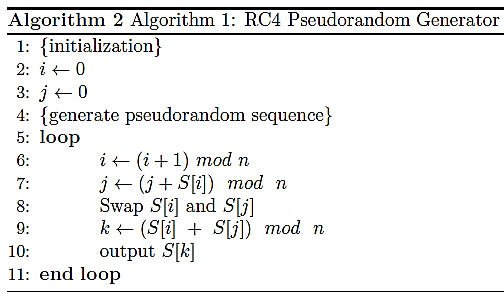
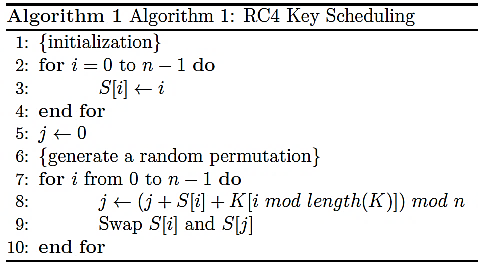


1. CMP Protocol for WPA2 Encryption and for Calculating MIC for Data Integrity

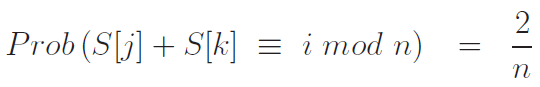


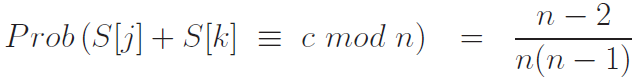
▓Some Highly Successful Attacks on WEP

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.

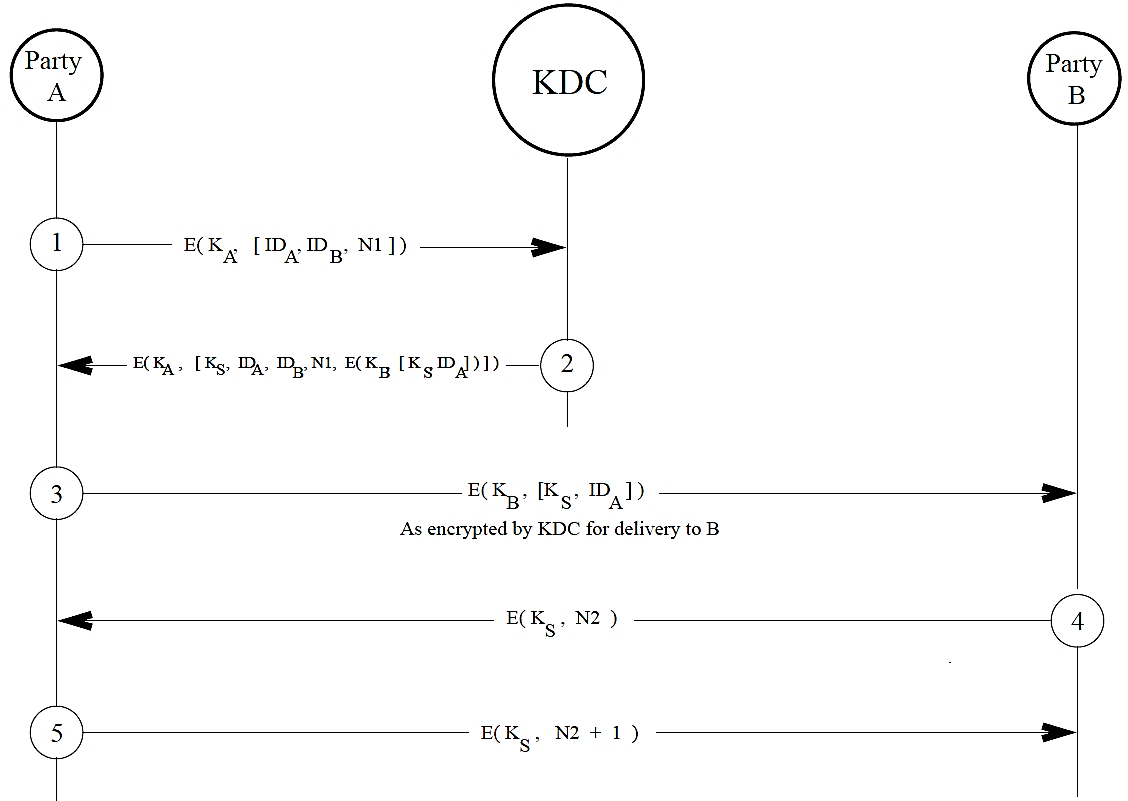


* 1. strong correlations exist in byte sequence produced by the pseudorandom byte generation algorithm. These correlations are expressed in the form of probabilities of the output pseudorandom sequence satisfying certain constraints vis-a-vis the the values of the state vector S.
  2. attack proposed 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.
  3. There are two main theoretical results derived by Klein that play a critical role in the attack.
     1. 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.



* + 1. 
  1. 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.

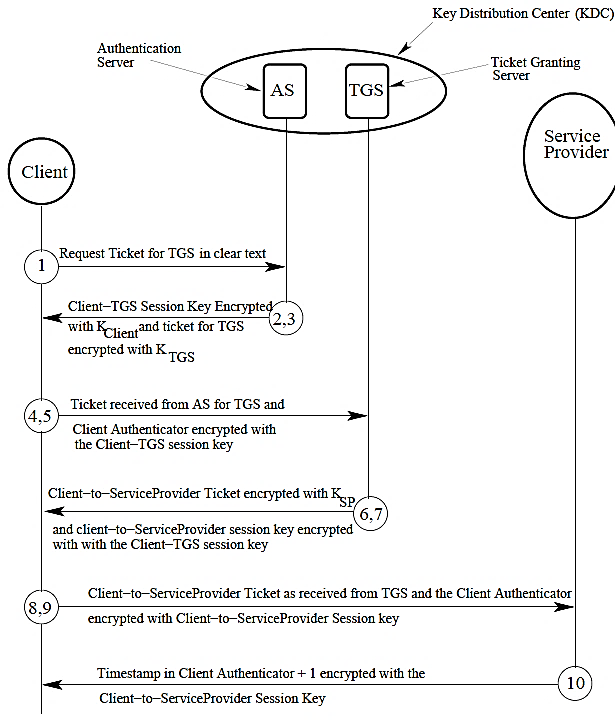
▓THE NEED FOR KEY DISTRIBUTION CENTERS due to large groups of people/processes/systems, especially when group membership is ever changing.



1. Needham-Schroder protocol
   1. 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).
   2. Using the key KA for encryption, user A sends a request to KDC for a session key intended specifically for communicating with user B.
   3. 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)
   4. KDC responds to A with a message encrypted using the key KA:
      1. session-key KS that A can use for communicating with B.
      2. 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.
      3. 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
   5. 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.
   6. 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.
   7. 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.
   8. 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.
      1. 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.
      2. 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.
      3. 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
   1. 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.
   2. Ticket Granting Server (TGS):in charge of providing security to the service providers.
   3. Keys
      1. KClient: secret key held by AS for the Client. this encryption key is not directly known to the Client.
      2. KTGS: secret key held by AS for TGS. TGS also has this key.
      3. KServiceProvider: secret key held by AS for the Service Provider. The Service Provider also has access to this key.
      4. KClient-TCG: session key that AS will send to Client for communicating with TGS.
      5. KClient-ServiceProvider: session key that TGS will send to the Client for communicating with the Service Provider.



* 1. Process
     1. Each Client registers with the Authentication Server and is granted a user identity and a secret password.
     2. Client sends a request in plain text to the AS.
     3. 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
           1. clients user ID
           2. clients network address
           3. validation time
           4. session key KClient-TGS
     4. 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.

* + 1. 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.
    2. 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.
    3. TGS now sends back to the Client the following two messages:
       1. A Client-to-ServiceProvider ticket that consists of:
          1. The Client ID
          2. the Client network address
          3. the validation period
          4. 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
    4. The client recovers the ticket meant for the service provider with KClient-TGS session key.
    5. 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.
    6. 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.
    7. 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
    8. 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.

1. 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.
2. 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.

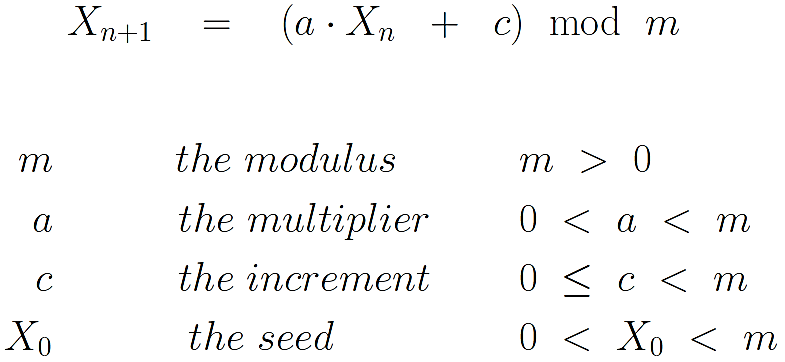
▓To be considered truly random, a sequence of numbers must exhibit two properties:

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).

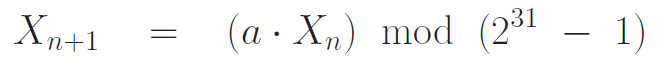
▓Algorithmically generated random numbers are called pseudorandom numbers.

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

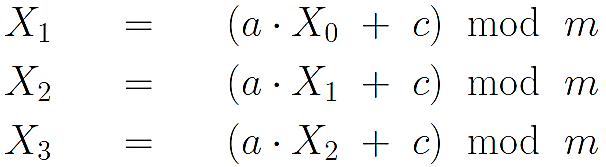
1. sequence of pseudorandom numbers X0, X1,., Xi, . is generated using recursion:



1. 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.
2. criteria on how to select values for these parameters m, a, c:
   1. 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.
   2. 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.
   3. 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.
3. 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



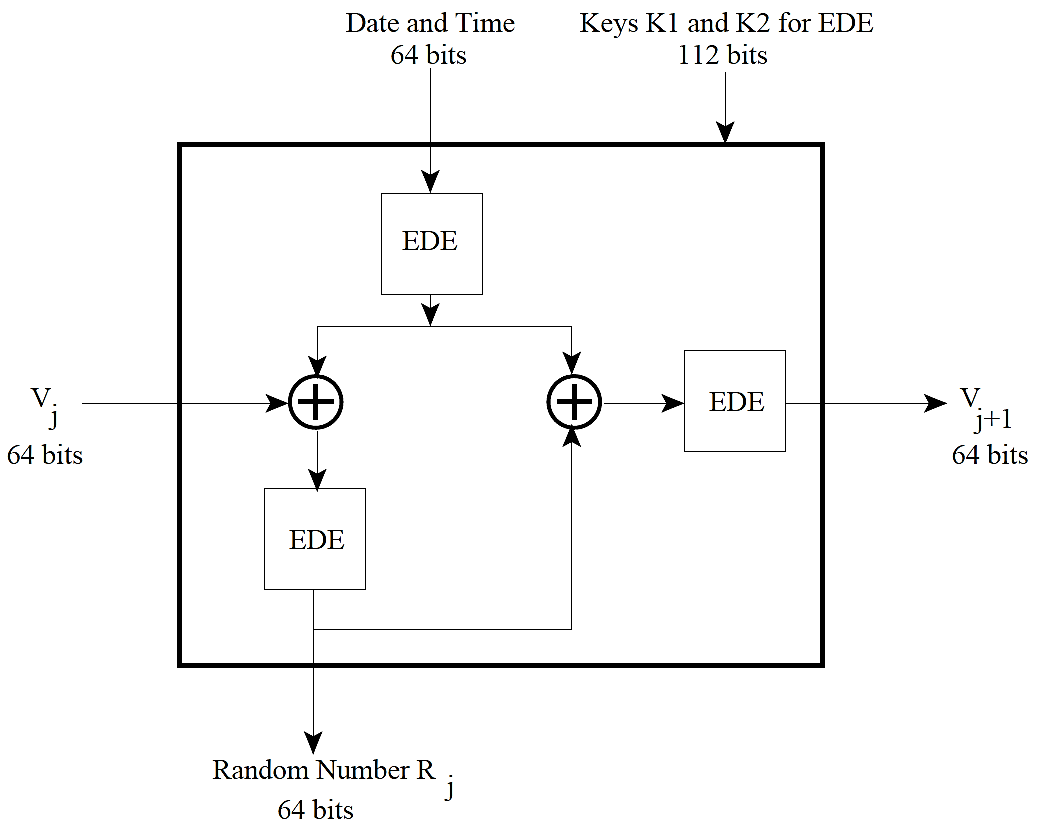
1. A pseudorandom sequence of numbers is cryptographically secure if it is difficult for an attacker to predict the next number from the numbers already in his/her possession.
2. When linear congruential generators are used for producing random numbers, the attacker only needs three pieces of information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



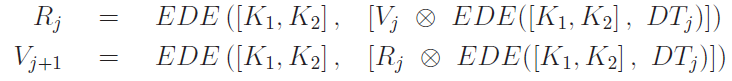
1. A pseudorandom sequence produced by a PRNG (pseudorandom number generator) can be made more secure from a cryptographic standpoint by restarting the sequence with a different seed after every N numbers. One way to do this would be to take the current clock time modulo m as a new seed after every so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



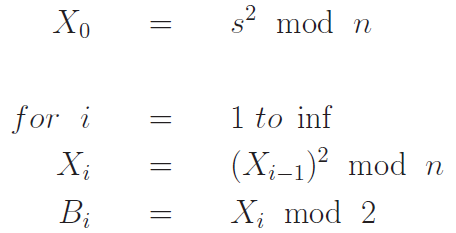
* 1. Each of the three EDE boxes stands for the two-key 3DES algorithm. All three EDE boxes use the same two 56-bitencryption keys K1 and K2.
  2. two inputs are:
     1. A 64-bit representation of the current date and time (DTj); and
     2. A 64-bit number generated when the previous random number was output (Vj).
  3. The PRNG is initialized with a seed value for V0 for the very first random number that is output.
  4. 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



* 1. reasons contribute to the cryptographic security of this approach to PRNG:
     1. 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.
     2. the scheme uses independently specified pseudorandom input an encryption of the current date and time.
     3. 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.
     4. 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.
  2. it is a much slower way to generate pseudorandom numbers. That makes this approach unsuitable for many applications that require randomized inputs.

1. BLUM BLUM SHUB GENERATOR (BBS) / cryptographically secure pseudorandom bit generator (CSPRBG)
   1. Algorithm
      1. first choosing two large prime numbers p and q that both yield a remainder of 3 when divided by 4. Let n = p\*q
      2. choose a number s that is relatively prime to n.
      3. BBS generator produces a pseudorandom sequence of bits Bj: (Bi is

the least significant bit of Xi at each iteration.)



* 1. 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



1. 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.
2. 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.
3. There are two types of entropy sources to consider
   1. 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.
   2. software based entropy sources.
      1. every computer there are constantly occurring phenomena that are consequences of some human interaction with that computer or some other networked computer.
      2. 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).
      3. 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.
      4. 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 = 0 and as that are multiples of p are excluded specifically.)

1. Proof:
   1. assuming that p is prime and a is a non-zero integer that is coprime to p: a, 2a, 3a, 4a, ......, (p-1)a
   2. if we reduce these numbers modulo p, we will simply obtain a rearrangement of the sequence: 1, 2, 3, 4, ......, (p-1)
   3. Therefore, we can say {a, 2a, 3a, ......, (p-1)a} mod p = some permutation of

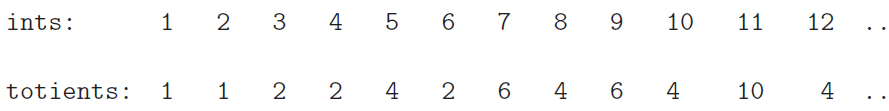
{1, 2, 3, ......, (p-1)} for every prime p and every a that is coprime to p.

* 1. multiplying all of the terms 2a, 3a, 4a, ......, (p-1)a yield
  2. Canceling out the common factors on both sides then gives the Fermats Little Theorem

1. 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 (n).

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

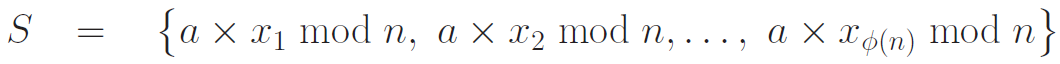


1. If p is prime, its totient is given by (p) = p-1.
2. Suppose a number n is a product of two primes p and q, n=pq then



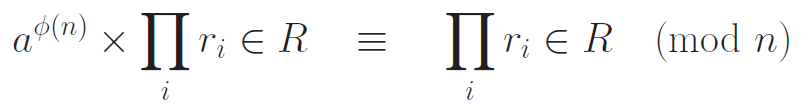
▓EULERS THEOREM: for every positive integer n and every a that is coprime to n, the following must be true

1. when n is a prime, (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
   1. R = {x1, x2,…,} the set of all integer less than n that are relatively prime to n.
   2. S be the set obtained when we multiply modulo n each element of R by some integer a co-prime to n.



* 1. 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.
     1. (amod n) cannot be zero: because a and xi are coprimes to n, so acannot contain n as a factor.
     2. (amod n) = (amod n) is impossible: if (amod n) = 0, either a is 0 mod n or mod 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 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
   1. Given any odd positive integer n, we can express n-1 as a product of a power of 2 and a smaller odd number:



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



So



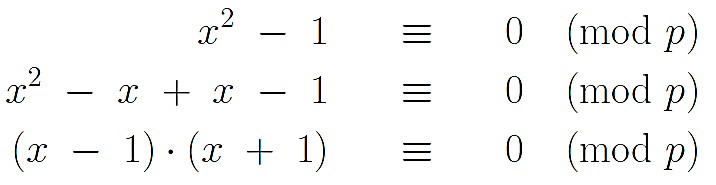


* 1. x^2= 1 has only trivial roots in Zp for any prime p: mean that only x = 1 and x =-1 can satisfy the equation x^2= 1.



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.

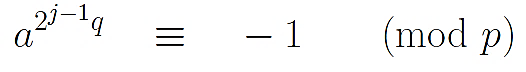




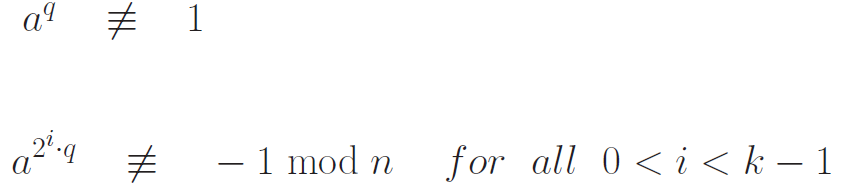
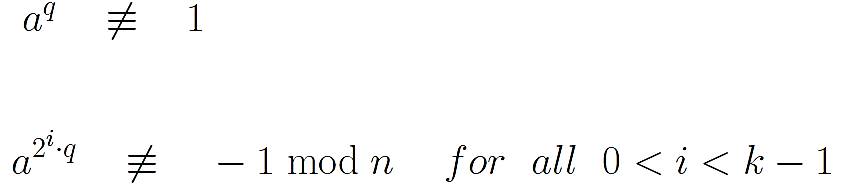
1. algorithm: For any integer a in the range 1 < a < p-1, one of the following conditions must be true when p is a prime:
   1. Either it must be the case that



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

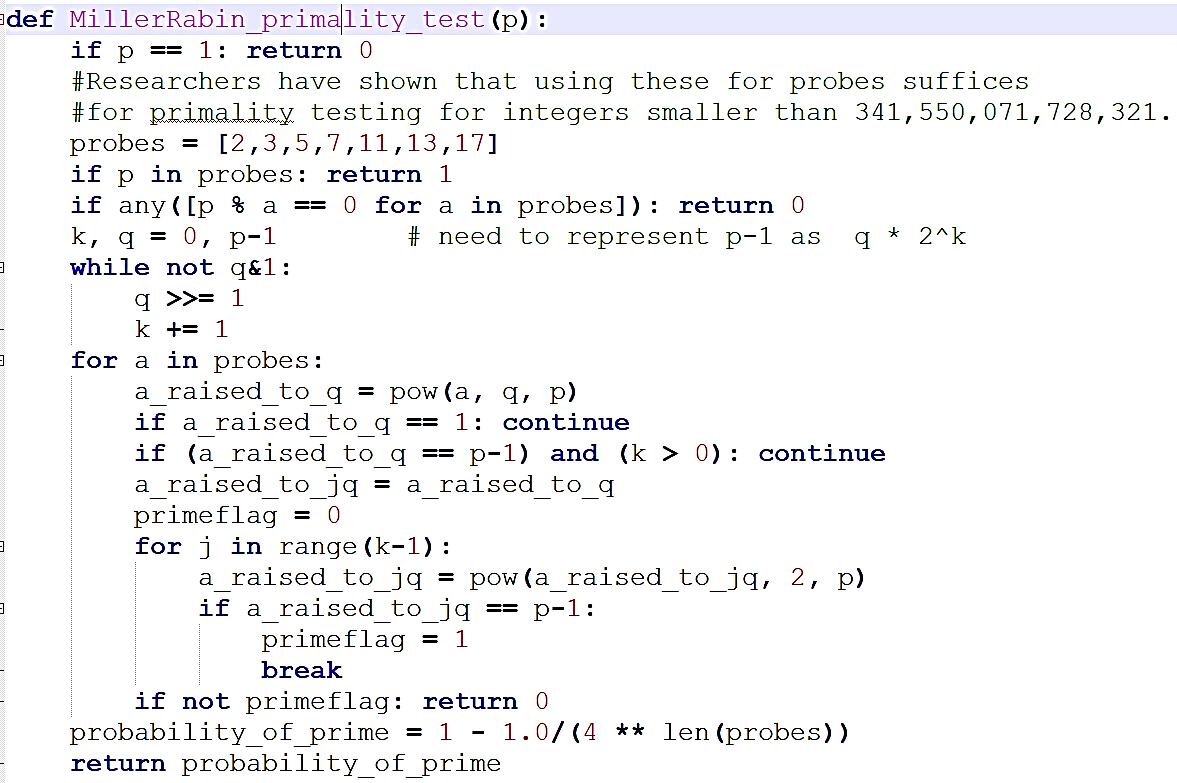


1. if either of the two Conditions is true for a probe a, then p may be either a composite or a prime.
2. When n is to be composite, then known



* 1. 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.
  2. 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.

1. 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.
2. running time of this algorithm is O(tn) or O(tn) where n is the integer being tested for its primality and t the number of probes used for testing.
3. In the theory of algorithms, the Miller-Rabin algorithm would be called a randomized algorithm belongs to the class co-RP. (randomized polynomial)
   1. randomized algorithm: algorithm that can make random choices during its execution.
   2. co-RP: problems that can be solved in polynomial time but when the answer is known to be no, the algorithm occasionally says yes
   3. 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.

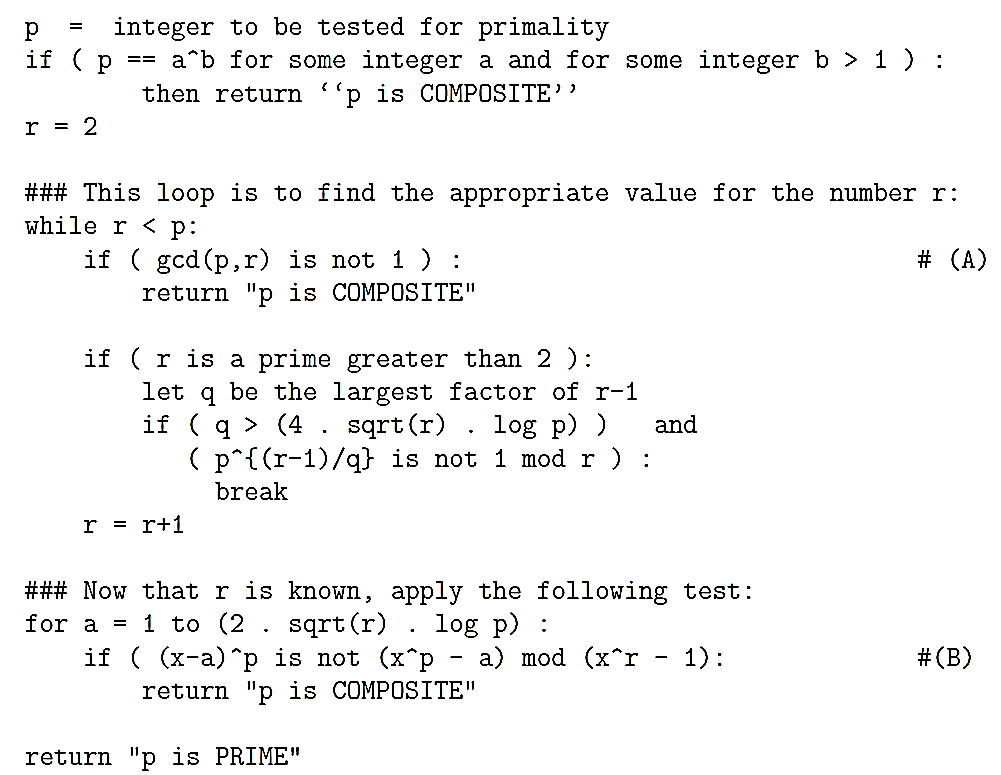


▓Agrawal-Kayal-Saxena (AKS) algorithm: if a number a is coprime to another number p, p > 1, then p is prime if and only if the (x+a)^p defined over the finite field Zp obeys:





1. two main challenges in creating an efficient implementation
   1. 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
   2. 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.



1. computational complexity of the AKS algorithm

where p is the integer whose primality is being tested and f is a polynomial. So the running time of the algorithm is

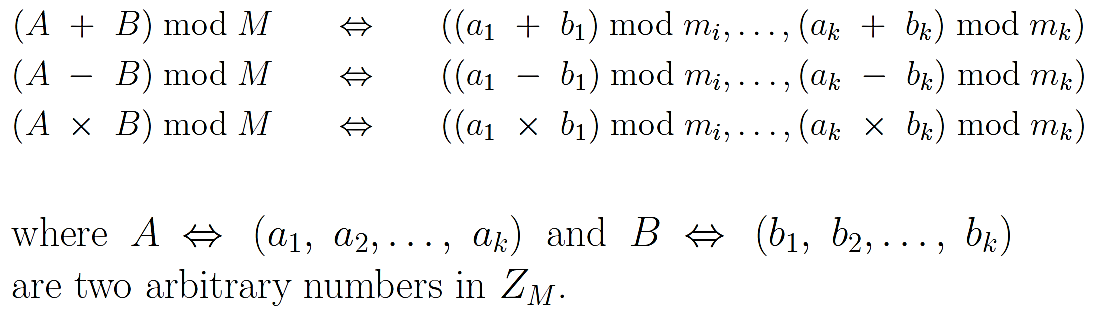
proportional to the twelfth power of the number of bits required to represent the candidate integer times a polynomial function of the logarithm of the number of bits.

▓CHINESE REMAINDER THEOREM (CRT): Particularly useful for modulo

arithmetic operations on very large numbers with respect to large moduli.

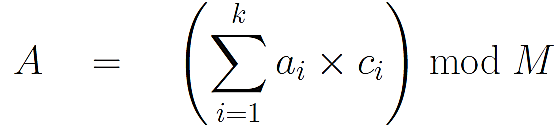
1. Algorithm
   1. in modulo M arithmetic, if M can be expressed as a product of n integers that are pairwise coprime, then every integer in the set ZM={0, 1, 2, ....,M-1} can be reconstructed from residues with respect to those n numbers.
   2. EX: 10=2\*5 => according to CRT, 9 can be represented by the tuple (1, 4)
   3. 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::
      1. The mapping between the integers A ZM and the k-tuples is a bijection, meaning that the mapping is one-to-one and onto.
      2. 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

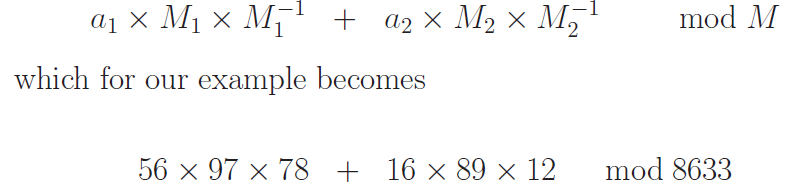


* 1. To compute the number A for a given tuple (a1, a2,…,ak)
     1. we first calculate Mi= M/mi, 1<= mi <= k



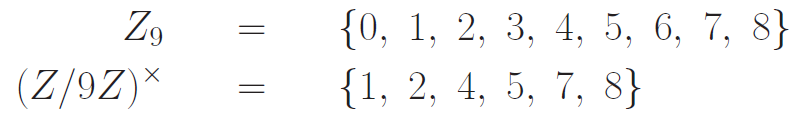
* + 1. 
    2. 

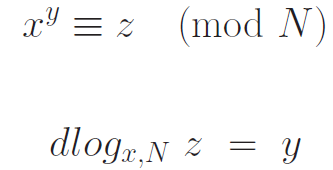
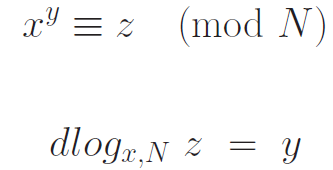
1. CRT is useful for manipulating very large numbers in modulo arithmetic
   1. Modulus M = 8633 = 89\*97 => m1 =89, m2 =97
   2. M1= M/m1= 97 and M2= M/m2= 89.
   3. ,
   4. we want to add two integers 2345 and 6789 modulo 8633.
   5. express the operand 2345 by its CRT representation, which is (31, 17) since 2345 mod 89 = 31 and 2345 mod 97 = 17.
   6. express the operand 6789 by its CRT representation,which is (25, 96) since 6789 mod 89 = 25 and 6789 mod 97 = 96.
   7. To add the two large integers, we simply add the two corresponding CRT tuples modulo the respective moduli. This gives us (56, 16).
   8. To recover the result as a single number, that returns the result 501.

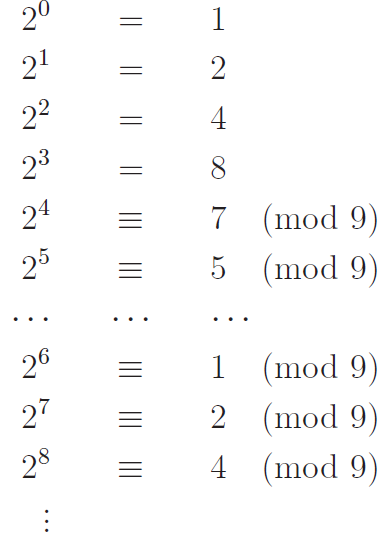
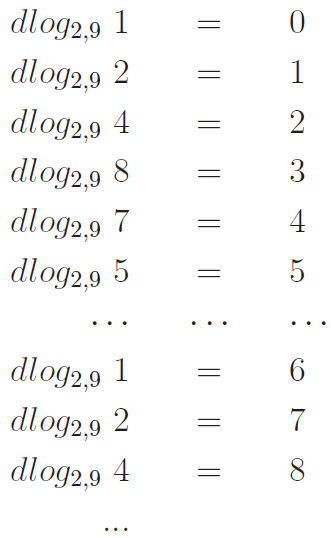


▓DISCRETE LOGARITHMS

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 = {1, 3, 5,7}. Choosing a prime for N, = {1,2,3,…,16}
2. For some values of N, the set 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 or primitive root modulo N. for example, N = 9, 2 is a primitive element of the group , which is the same as primitive root mod 9. A primitive root can serve as the base of discrete logarithm.



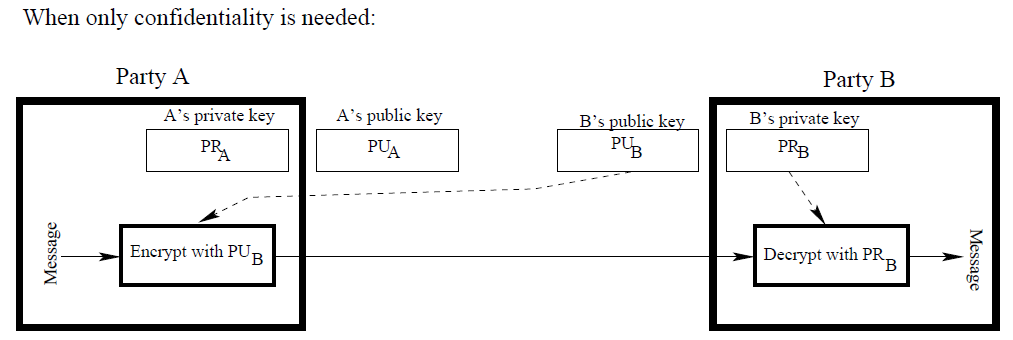
 

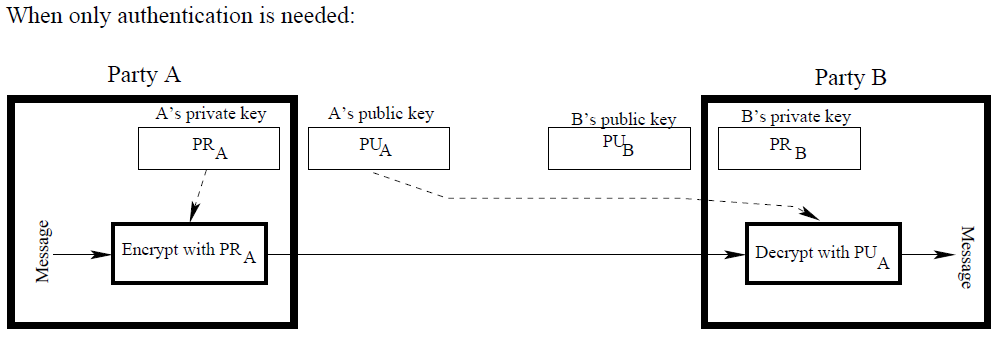
 

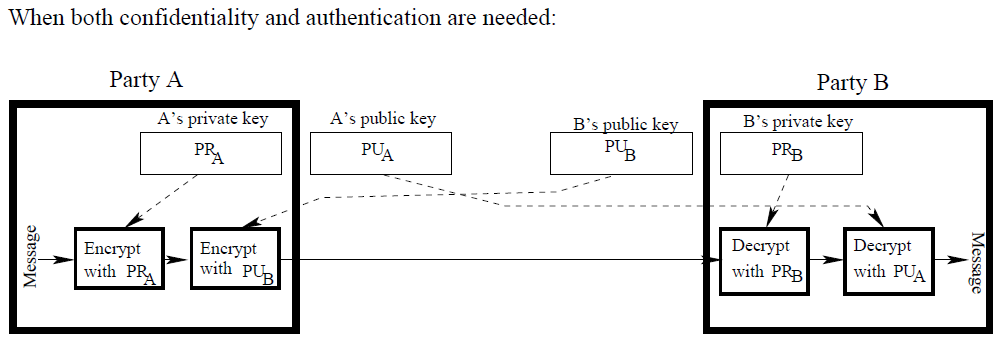
1. unique discrete logarithm mod N to some base a exists only if a is a primitive root modulo N.

▓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.
   1. 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.
   2. 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.







1. 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.

* 1. The processing steps undertaken by A to convert M into C



* 1. The processing steps undertaken by B to recover M from C



* 1. 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

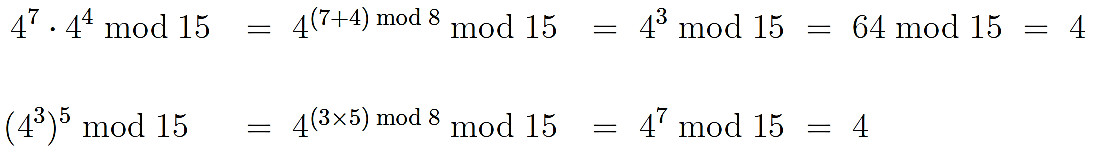
1. Eulers Theorem: for every positive integer n and every a that is coprime to n



1. when a and n are relatively prime, the exponents will behave modulo the totient ‑(n) in exponentiated forms like mod n
2. if a and n are relatively prime, the following must be true



1. For example, a = 4 in arithmetic modulo 15. The totient of 15 is 8. (15 = 3\*5, so ‑(15) = 2\*4 = 8.)

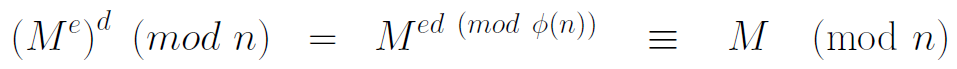


1. M is an integer that represents a message, conjure up two integers e and d that are each others multiplicative inverses modulo the totient (n).



* 1. N = a modulus for modular arithmetic
  2. (n) = the totient of n
  3. e = an integer that is relatively prime to(n) [This guarantees that e will possess a multiplicative inverse modulo ‑(n)]
  4. d = an integer that is the multiplicative inverse of e modulo (n)



1. 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.
2. 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}
3. 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.
4. 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.
   1. If two integers p and q are coprimes



* 1. Only when p and q are individually prime that



* 1. it is important that both p and q be very large primes, so it is computationally harder to determine its primality.
  2. We also need to ensure that n is not factorizable by one of the modern integer factorization algorithms.

1. 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.
   1. since the integer d is the multiplicative inverse of the integer e modulo the totient ‑(n)



* 1. since ‑(n) = ‑(p)\*(q), (p) and ‑(q) must also individually be divisors of e\*d-1

.



* 1. if M and p are coprimes. By Fermats Little Theorem







* 1. if M is a multiple of the prime p, M mod p = 0 => M^k mod p = 0



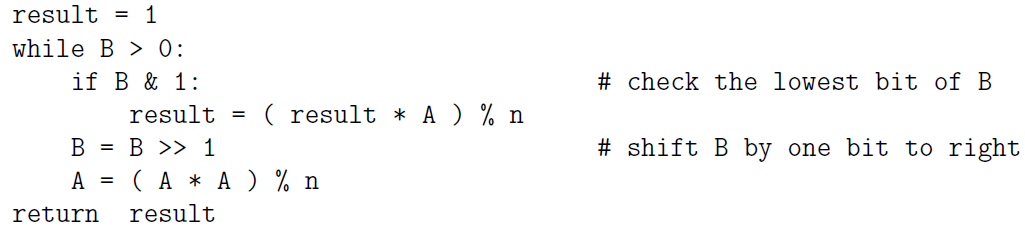
* 1. we can draw identical conclusion regarding q



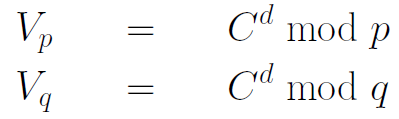
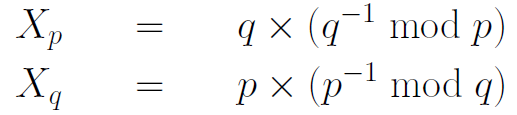
* 1. when p and q are coprimes, for any integers a and b if we have a≡b (mod p) and a ≡ b (mod q), then it must also be the case that a ≡b (mod pq).



1. computational steps for key generation
   1. Generate two different primes p and q
      1. generate a random number of size B/2 bits.
      2. set the lowest bit of the integer generated to ensure it is odd number
      3. set the two highest bits to ensure the highest bits of n is set.
      4. 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.
   2. Calculate the modulus n = p\*q
   3. Calculate the totient ‑(n) = (p-1)\*(q-1)
   4. Select for public exponent an integer e such that 1 < e < ‑(n) and gcd((n), e) = 1 [equivalent to gcd(p-1, e) = 1 and gcd(q-1, e) = 1]
      1. 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
      2. 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.
   5. Calculate for the private exponent a value for d such that d = e^(-1) mod ‑(n) [modular inversion]
      1. use the Extended Euclids Algorithm
   6. Public Key= [e, n]
      1. modular exponentiation: raising the message integer M to the power of the public exponent e modulo n.
      2. 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.



* 1. Private Key = [d, n]
     1. speeded up by using the Chinese Remainder Theorem (CRT)

* + 1. Fermats Little Theorem (FLT) can speed up the calculation of Vp & Vq:



* + 1. 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).

1. RSA lacks forward secrecy
   1. 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.
   2. 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.
   3. solution to this problem with RSA lies in creating a session key without either party transmitting the key to the other party.
2. Chosen Ciphertext Attacks (CCA)
   1. you use my public key (n, e) to encrypt a plaintext message M into the ciphertext C. C is picked up by attacker
   2. attacker randomly chooses an integer s and constructs a new message by forming the product C”= s^e \* C mod n.
   3. 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
   4. attacker will now be able to recover the original message M by M = M’ \*s^(-1) mod n
3. 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.
4. 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)
   1. semiprime/ biprimes/ pq-numbers/ 2-almost primes: a number that is a product of two (not necessarily distinct) primes
   2. 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)
5. The size of the key in the RSA algorithm typically refers to the size of the modulus integer in bits.
   1. The exponential relationship between what it takes to represent an integer in the memory of a computer and the value of that integer.
   2. 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.
   3. 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.
   4. 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.
6. 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.