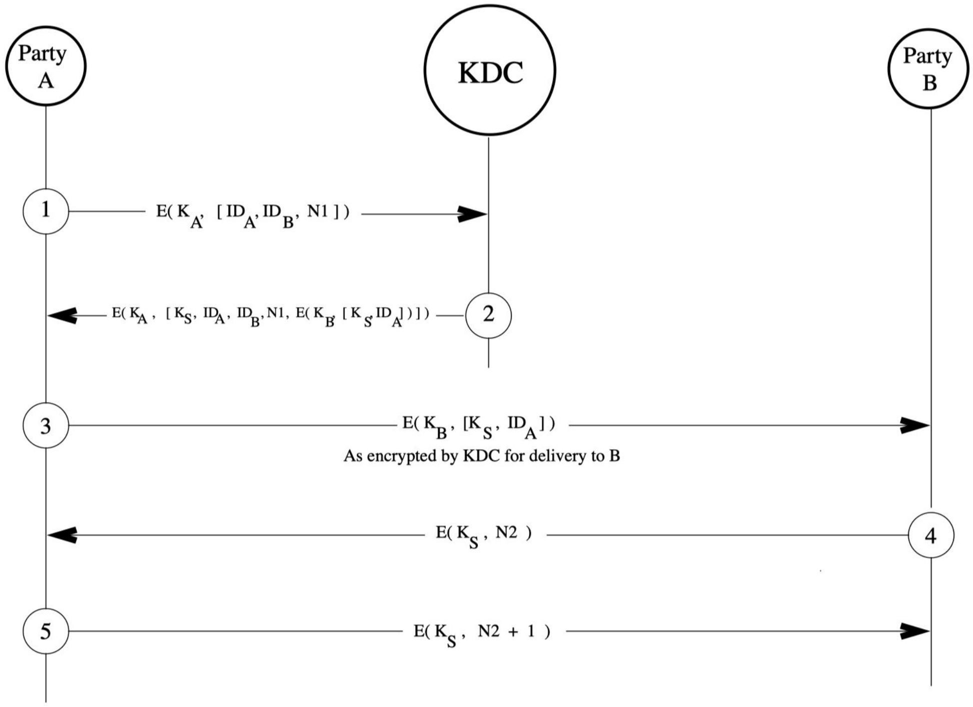
1. Stream Ciphers
2. WEP/WPA/WPA2 (differences and vulnerabilities)
3. Key Distribution Protocols (Needham-Schroeder and Kerberos)
4. Random Number Generation (including entropy sources and X9.31)
5. Miller-Rabin Primality Test (minor question)
6. Asymmetric Cryptography and RSA
7. Certificates and Digital Signatures
8. Diffie-Hellman Key Exchange Algorithm
9. ECC (minor question)
10. Hashing (including the birthday attack)
11. Cryptocurrency (minor question)
12. TCP/IP (just the OSI 7-layer Model and the 3-way Handshake)

Needham-Schroeder Symmetric Key Protocol - Checking Party ID

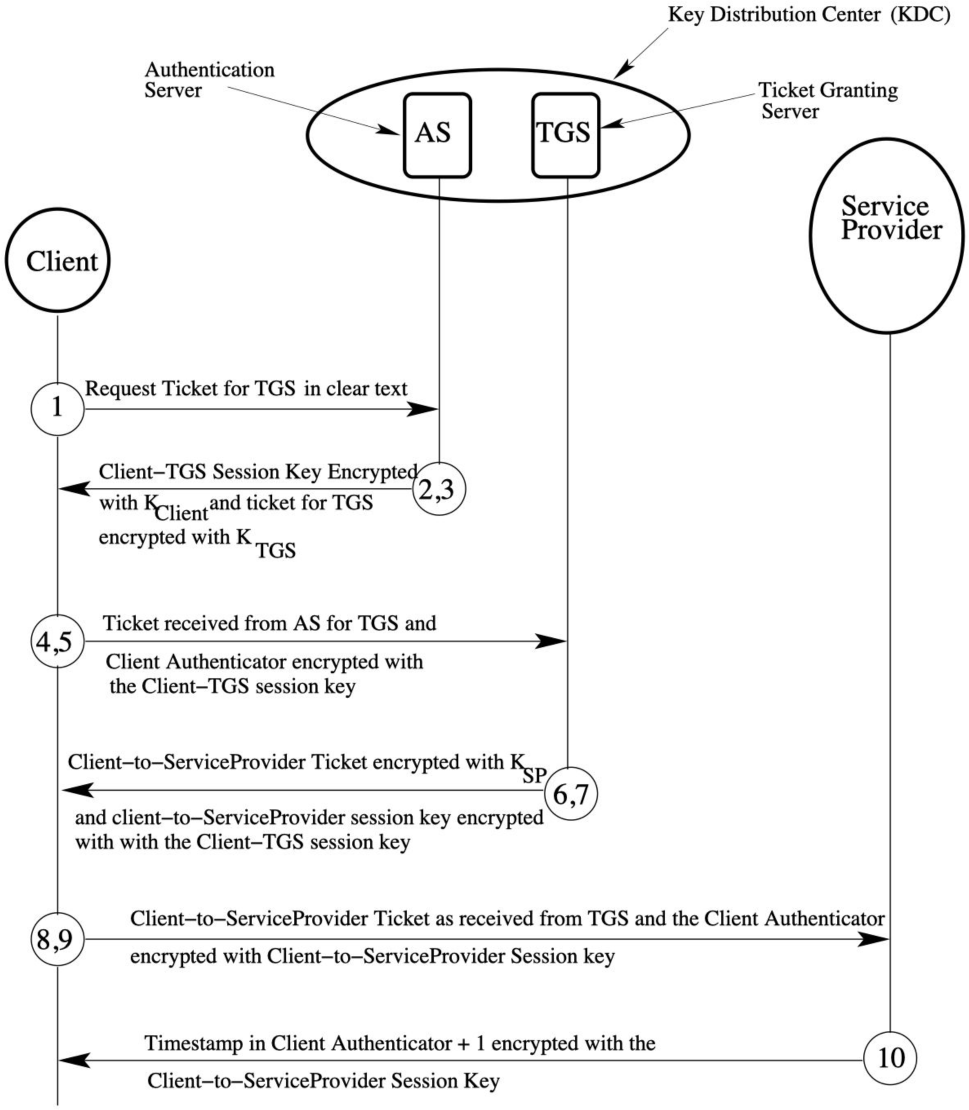
1. Provide confidentiality for communication
    a. Each party shares a master key with KDC (Key Distribution Center).
    b. KDC's function is to generate session keys for a connection between two parties.
    c. Session keys are protected by master keys during distribution.
2. What the KDC sends back to A for B can only be understood by B.


Party A
KDC
Party B
1
$E(K_A, [ID_A, ID_B, N1])$
2
$E(K_A, [K_S, ID_A, ID_B, N1, E(K_B, [K_S, ID_A])])$
3
$E(K_B, [K_S, ID_A])$
As encrypted by KDC for delivery to B
4
$E(K_S, N2)$
5
$E(K_S, N2 + 1)$

    a. Party A cannot look inside the ticket/packet $[K_S, ID_A]$
    b. Message 3: This first contact from A to B is protected from eavesdropping.
    c. Message 5: Ensures that B knows that it did not receive a first contact from A that A is no longer interested in; It provides some protection against a replay attack.
3. Nonce N: a unique session identifier - prevent replay attack
    a. It is a random number.

Kerberos Symmetric Key Protocol

1. Differences between Needham-Schroeder and Kerberos:
    a. Kerberos protocol makes a distinctions between the clients and the service providers but Needham-Schroeder protocol does not.
    b. For Kerberos protocol, a client cannot gain direct access to TGS and only the TGS can provide a session key to communicate with a service provider.
2. Advantages of Kerberos:
    a. Makes a distinctions between the clients and the service providers
    b. Separating AS from TGS allows the Client needs to contact AS only once for a Client-to-TGS ticket and the Client-to-TGS session key. These can be used for multiple requests to the different service providers.
3. The KDC in Kerberos has 2 parts, AS, which provides client authentication, and TGS, which provides security to service providers. A client must first authenticate himself/herself/itself to AS and obtain from AS a session key for accessing TGS.


Key Distribution Center (KDC)
Authentication Server
AS
TGS
Ticket Granting Server
Service Provider
Client
1
Request Ticket for TGS in clear text
2,3
Client−TGS Session Key Encrypted with K_{Client} and ticket for TGS encrypted with K_{TGS}
4,5
Ticket received from AS for TGS and Client Authenticator encrypted with the Client−TGS session key
6,7
Client−to−ServiceProvider Ticket encrypted with K_{SP} and client−to−ServiceProvider session key encrypted with with the Client−TGS session key
8,9
Client−to−ServiceProvider Ticket as received from TGS and the Client Authenticator encrypted with Client−to−ServiceProvider Session key
10
Timestamp in Client Authenticator + 1 encrypted with the Client−to−ServiceProvider Session Key

a. Message 1: The Client sends a request in plain text to the AS.

b. The AS sends back 2 messages, “Here is a TGT (the initial ticket) that you can decrypt this using your password.”

   i. Message 2: $E(K_{Client}, [K_{Client-TGS}])$

      K_{Client} is NOT directly known to the Client, but the Client can decrypt it and get $K_{Client\text{-}TGS}$ using his/her/its password. The password is immediately destroyed after it is used.

   ii. Message 3:

      may be expressed by

      $$E(K_{Client}, [K_{Client-TGS}, E(K_{TGS}, \overbrace{\left([ClientID, ClientIP, ValidityPeriod, K_{Client-TGS}]\right)}^{TGT})])$$

      A TGT (Ticket-Granting Ticket) is meant for delivery to TGS.

c. After decryption, the Client sends TGS the TGT (still encrypted by K_{TGS}) and the Client Authenticator, “Here is the TGT. Give me the service ticket.”

   i. Message 5: A Client Authenticator is composed of the client ID and the timestamp together encrypted by $K_{Client\text{-}TGS}$.

d. The TGS decrypts the ticket in Message 5 using session key $K_{Client\text{-}TGS}$ and sends back a Client-to-ServiceProvider ticket and a Client-to-ServiceProvider session key.

   i. Message 6: A Client-to-ServiceProvider ticket consists of ClientID, ClientIP, ValidityPeriod, and a session key for the Client and the Service Provider $K_{Client-ServiceProvider}$.

      The ticket is encrypted by $K_{ServiceProvider}$ (K_{SP}).

   ii. Message 7: $K_{Client-ServiceProvider}$ encrypted by $K_{Client\text{-}TGS}$.

e. The client recovers the ticket and sends the Service Provider the Client-to-ServiceProvider ticket and an authenticator.

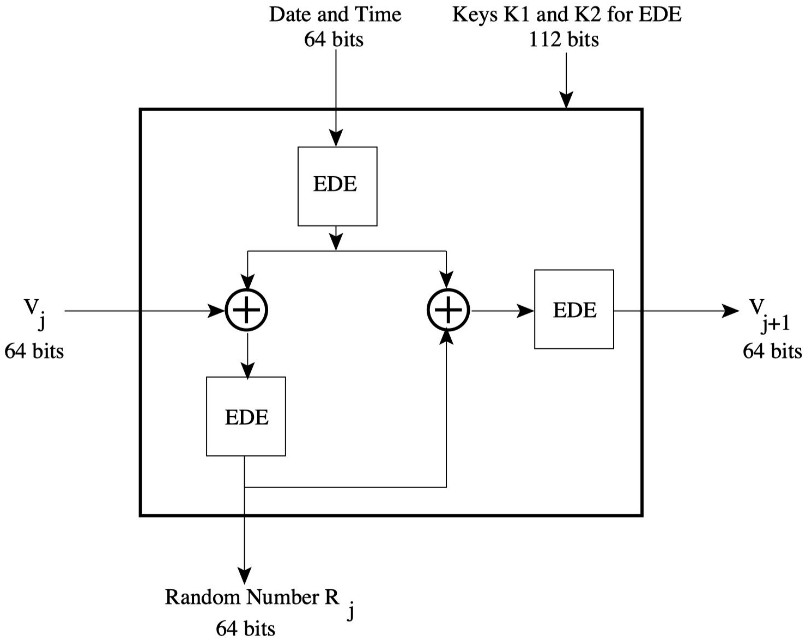
   i. Message 8: The Client-to-ServiceProvider ticket is encrypted by K_{SP}.

   ii. Message 9: The authenticator is encrypted by $K_{Client-ServiceProvider}$.

f. Message 10: 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 $K_{Client-ServiceProvider}$ session key.

Random Number Generation

1. 2 properties of truly random number:
    a. Uniform distribution
    b. Independence
2. Cryptographically secure: It is difficult for an attacker to predict the next number from the numbers already in his/her possession.
3. Pseudorandom number generators (PRNGs): Linear congruential generators
    a. NOT cryptographically secure, only need 3 pieces of info to break
    b. $X_{n+1} = (aX_n + c) \bmod m$ $0 <= X_n <= m$
        i. We want m to be a prime and c = 0. Then certain values of a will guarantee an output sequence with a period of m-1.
4. Cryptographically secure PRNG's (CSPRNG) - X9.17/X9.31
    a. Reasons that contribute to the cryptographic security:



        i. The new seed V_{j+1} cannot be predicted from the current random number R_j.
        ii. Independently pseudorandom input: date and time
        iii. Hard to predict V_{j+1} from V_j because there stand at least two EDE encryptions between them.

Entropy Sources for Generating True Random Numbers

1. Difference between a TRNG (True Random Number Generator) and PRNG:
   A PRNG must have a seed for initialization. A TRNG works without a seed.
2. An entropy source is any source that is capable of yielding a truly random stream of 1's and 0's.
3. Entropy with the bit stream:
   $H \quad = \quad -\sum_i p_i \log_2 p_i$ - the i-th value with probability p_i
   a. The highest entropy possible for 8-bit words is 8 bits:
      p_i = 1/256, H = 256 * (8/256) = 8
   b. If the probability distribution of the values taken by 8-bit words were to become nonuniform, the entropy will become less than its maximum value.
   c. When all the 8-bit patterns are the same, the entropy is zero.

Primality Testing

1. Fermat's Little Theorem $a^{p-1} \equiv 1 \pmod{p}$
   a. a is coprime to p. a is not allowed to be 0 or multiples of p.
   b. a is the probe for primality testing for p.
   c. When $a^{p-1} \not\equiv 1 \pmod{p}$, p is definitely not a prime.
      When $a^{p-1} \equiv 1 \pmod{p}$, p may either be a prime or a composite (non-prime).
   d. Proof:
      {a, 2a, 3a, ..., (p − 1)a} mod p = some permutation of {1, 2, 3, ..., (p − 1)}
      Multiplying all of the terms on the left hand side will yield
      $(a^{p-1})(1)(2)...(p-1) \equiv (1)(2)...(p-1)$
      Canceling out the common factors on both sides then gives the Fermat's Little Theorem.
2. Euler's Totient Function φ(n)
   a. 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.
   b. φ(p) = p − 1 for prime p
   c. Suppose n = (p)(q), then φ(n) = φ(p)φ(q) = (p − 1)(q − 1)
3. Euler's Theorem $a^{\varphi(n)} \equiv 1 \pmod{p}$
   a. For every positive integer n and every a that is coprime to n
4. Difference between Fermat's Little Theorem and Euler's Theorem:
   a. Fermat's Little Theorem: apply to only prime number p
      If gcd(a, p)=1 where p is prime then $a^{p-1} \equiv 1 \pmod{p}$.
   b. Euler's Theorem: apply to every positive integer n
      If gcd(a,n)=1 then $a^{\varphi(n)} \equiv 1 \pmod{p}$

Miller-Rabin Algorithm for Primality Testing

1. For any odd positive integer n:
   $n - 1 = 2^k \cdot q$ for some $k > 0$, and odd q
2. Miller-Rabin Algorithm uses the fact that $x^2 = 1$ has only trivial roots in Z_p for any prime p (only x=1 and x=-1)
3. Miller-Rabin Algorithm - one of the following conditions must be true when p is a prime: (for any integer a in the range 1 < a < p-1)
   Derivation from $a^{p-1} \equiv 1 \pmod{p}$ and $p - 1 = 2^k \cdot q$
   a. Condition 1: $a^q \equiv 1 \pmod{p}$
      For an odd number q
   b. Condition 2: $a^{2^\wedge(j-1)} \equiv -1 \pmod{p}$ $1 <= j <= k$
4. It is computationally efficient because it is easy to check the conditions since every number is the square of the previous number.
5. Right shift the number to find k and q.

CRT (Chinese Remainder Theorem)

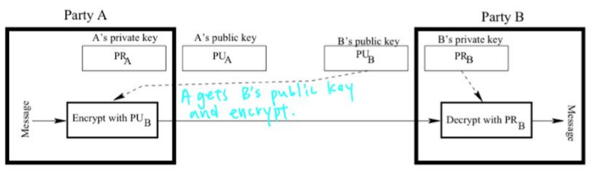
M = 8633
8633 = 89*97 = m1*m2 m1 = 89, m2 = 97
M1 = M/m1 = 97, M2 = M/m2 = 89
$M1^{-1}$ mod m1 = 78, $M2^{-1}$ mod m2 = 12
Let's say we want to add two integers 2345 and 6789 modulo M = 8633
2345 is (2345%m1, 6789%m2) = (31, 17), 6789 is (25, 96)
(31, 17) + (25, 96) = (56, 16) = (a1, a2)
$(a1)(M1)(M1^{-1}) + (a2)(M2)(M2^{-1})$ mod M = (56)(97)(78) + (16)(89)(12) mod 8633 = 501

Public-Key Cryptography

Encryption and decryption use two different keys - public key and private key
Sending message with its own private key provides authentication.

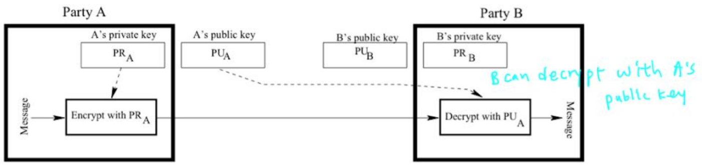
Party A wants to send a message to Party B

When only confidentiality is needed:



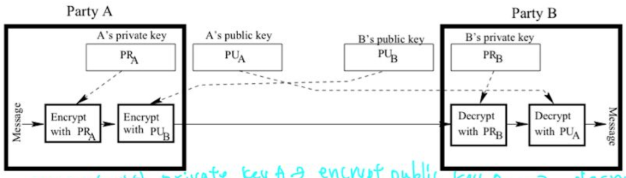
A communicate with B
Confidential
⇒ encrypt message using B's pub key
B decrypt using private key

When only authentication is needed:



Authentication.
A encrypt with A's private
can be decrypt by public A.

When both confidentiality and authentication are needed:



encrypt with private key A → encrypt public key B → decrypt privateB key → decrypt public key A

Figure 1: *This figure shows how public-key cryptography can be used for confidentiality, for digital signatures, and for both.* (*This figure is from Lecture 12 of "Computer and Network Security" by Avi Kak.*)

Confidential + Authenticatio

encrypt with A private

↓

encrypt with B public

↓

decrypt B private

decrypt A public

RSA for Public-Key Cryptography

1. An immediate consequence of Euler's Theorem:
   If a and n are relatively prime, then

$$a^k \equiv a^{k_1 \cdot \phi(n) + k_2} \equiv a^{k_1 \cdot \phi(n)} a^{k_2} \equiv a^{k_2} \pmod{n}$$

e.g. $a=4$, $n=15$, $\phi(15)=8$

$4^7 \cdot 4^4 \bmod 15 = 4^{(7+4) \bmod 8} \bmod 15 = 4^3 \bmod 15 = 4$

$(4^3)^5 \bmod 15 = 4^{(3\times5) \bmod 8} \bmod 15 = 4^7 \bmod 15 = 4$

2. $M^{exd} \equiv M^{exd(\bmod (\phi(n))} \equiv M \bmod n$
   a. e is relatively prime to φ(n) (to guarantee that e will possess a MI modulo φ(n)).
   b. d is the MI of e modulo φ(n).
3. C is the ciphertext and M is the plaintext.
   a. $C = M^e$ mod n B's public key {e, n}
   b. $M = C^d$ mod n B's private key {d, n}
4. B can choose to use RSA as a block cipher. When RSA is used as a block cipher, the block size is half the size of the key size (modulus n).
5. Steps for key generation in RSA:
   a. Generate 2 different primes p and q:
      i. p size = q size = block size = size of modulus / 2
      ii. Set the lowest bit and the 2 highest bits
      iii. Use Miller-Rabin to check if p and q are prime
      iv. p ≠ q
   b. Calculate n = p * q
   c. Calculate φ(n) = (p-1) * (q-1)
   d. Select for e such that 1 < e < φ(n) and gcd(φ(n), e) = 1
   e. Calculate $d = e^{-1}$
   f. Public key {e, n} Private key {d, n}

6. Decryption M = C^d mod n can be speed up using CRT:

$$\begin{aligned} V_p &= C^d \bmod p & X_p &= q \times (q^{-1} \bmod p) \\ V_q &= C^d \bmod q & X_q &= p \times (p^{-1} \bmod q) \end{aligned}$$

$$C^d \bmod n = (V_p X_p + V_q X_q) \bmod n$$

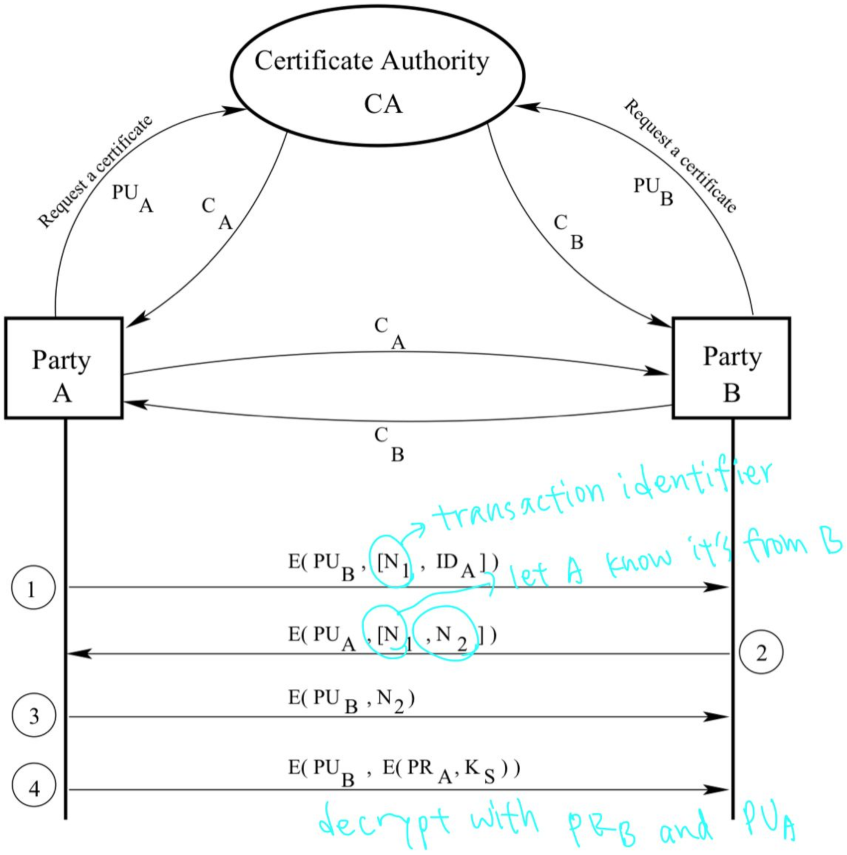
7. Vulnerabilities of RSA:
    a. Using CRT - Side Channel Attack
    b. Fault Injection Attack - get p, q miscalc
    c. Timing Attack
    d. Lack of forward secrecy
        i. Solutions: DH, and DHE-RSA
    e. Chosen Cipher Attack
    f. Low entropy random number

CA (Certificate Authorities) for Authenticate Your Public Key

1. A little background:
    a. Public key cryptography is not suitable for the encryption of the actual message content, but it leads to superior protocols for managing and distributing secret session keys that can subsequently be used for the encryption of actual message content using symmetric-key algorithm such as AES, 3DES, RC4, etc.
    b. Direct key exchange protocol is vulnerable to the man-in-the-middle attack, so switch to the key exchange protocols involving CA.
2. The CAs operate through a strict hierarchical organization.
   Root CAs -> Intermediate-Level CAs
3. A certificate issued by a CA authenticates your public key. = A certificate is your public key signed by the CA’s private key (or encrypted with CA’s private key). There are 3 kinds of certificates:
    a. Extended Validation (EV)
    b. Organization Validation (OV)
    c. Domain Validation (DV)
4. Upper half: User and public key authentication
    a. C_A and C_B are the certificates.
    b. C_A is A’s public key encrypted with CA’s private key.
       $C_A = E(PR_{CA}, [T, ID_A, PU_A])$
    c. After A gives C_A to B, B can verify the legitimacy by decrypting it with the CA’s public key.
5. Lower half: Using authenticated public keys to exchange a secret session key

Parties A and B want to establish a secure and authenticated communication link

(Party A initiates a request for the link)



Diffie-Hellman Key Exchange Algorithm

1. (p, g) is public with a large prime number p and an element g of Z_p^* that generates a large order cyclic subgroup.
    a. $Z_p^* = \{1,2,3,...,p-1\}$ with group operator being modulo p multiplication.
    b. A group is cyclic if N=2 or 4 or p^m where p is an odd prime
    c. Z_{17}^* is a cyclic group with g = 3. That is, if you compute 3^i mod 17 for all i = 0,1,2,..., you will get the 16 numbers in the multiplicative group Z_{17}^*.
2. Select private keys:
   Select a random number X_A from the set {2, …, p-2}
   Same for X_B
3. Select public keys:

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

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

4. A makes Y_A available to B, and B makes Y_B available to A.
5. Calculate secret key:

$$K_A = Y_B^{X_A} \bmod p$$

$$K_B = Y_A^{X_B} \bmod p$$

where $K_A = K_B$

6. Why is DH protocol so magical?
   a. An eavesdropper having access to the public keys for both A and B would still not be able to figure out the secret key K.
   b. DH protocol allows two parties to create a shared secret K without either party having to send it directly to the other.
7. It is extremely hard to compute the discrete logarithms.
8. Vulnerability of DH - Man-in-the-middle attack:

If the attacker can intercept and change public key $Y_A \rightarrow Y_A'$, $Y_B \rightarrow Y_B'$

The secret key generated by would be different from that generated by B.

But the attacker would know both keys

Digital Signature Standard (DSS) Based on The ElGamal Algorithm

1. The ElGamal protocol is a variant of the Diffie-Hellman protocol. Y_A Remains fixed for a long time. B encrypts message M by M × K mod p. The decryption by A consists of dividing the received ciphertext by K modulo p.
2. Digitally signature: first calc a hash of the document, then encrypted the hash with the private key, and make this encrypted block available along with the document. When a party wants to verify that the document is authentic, they use your public key to extract the hash from the encrypted block, and compare this hash with the hash their computer calc directly from the document.
3. If want to sign the documents you make available to others on the internet:
    a. Create a public key - private key pair
    b. Select a large prime p and then randomly select two numbers g and X, less than p. Make (p, g) publicly available. Treat X as your private key
    c. Now calc your public key $Y = g^X \bmod p$
    d. Generate a one-time random number K such that $0 < K < p-1$ and $\gcd(K, p-1) = 1$
        i. Must keep K private because attacker can know your private key via K
    e. Let M be the integer that represents whatever it is you want to sign. To construct the signature sig1 and sig2:
        i. $sig1 = g^k \bmod p$
        ii. $sig2 = K^{-1} \times (M - X \times sig1) \bmod (p-1)$ where K^{-1} is the MI of K in modulo p-1
    f. After sending the message M along with your signature (sig1, sig2) to some recipient and the recipient wishes to make sure that he/she is not receiving a modified message. Check:

$$Y^{sig_1} \times sig_1{}^{sig_2} \quad \equiv \quad g^M \quad (mod\ p)$$

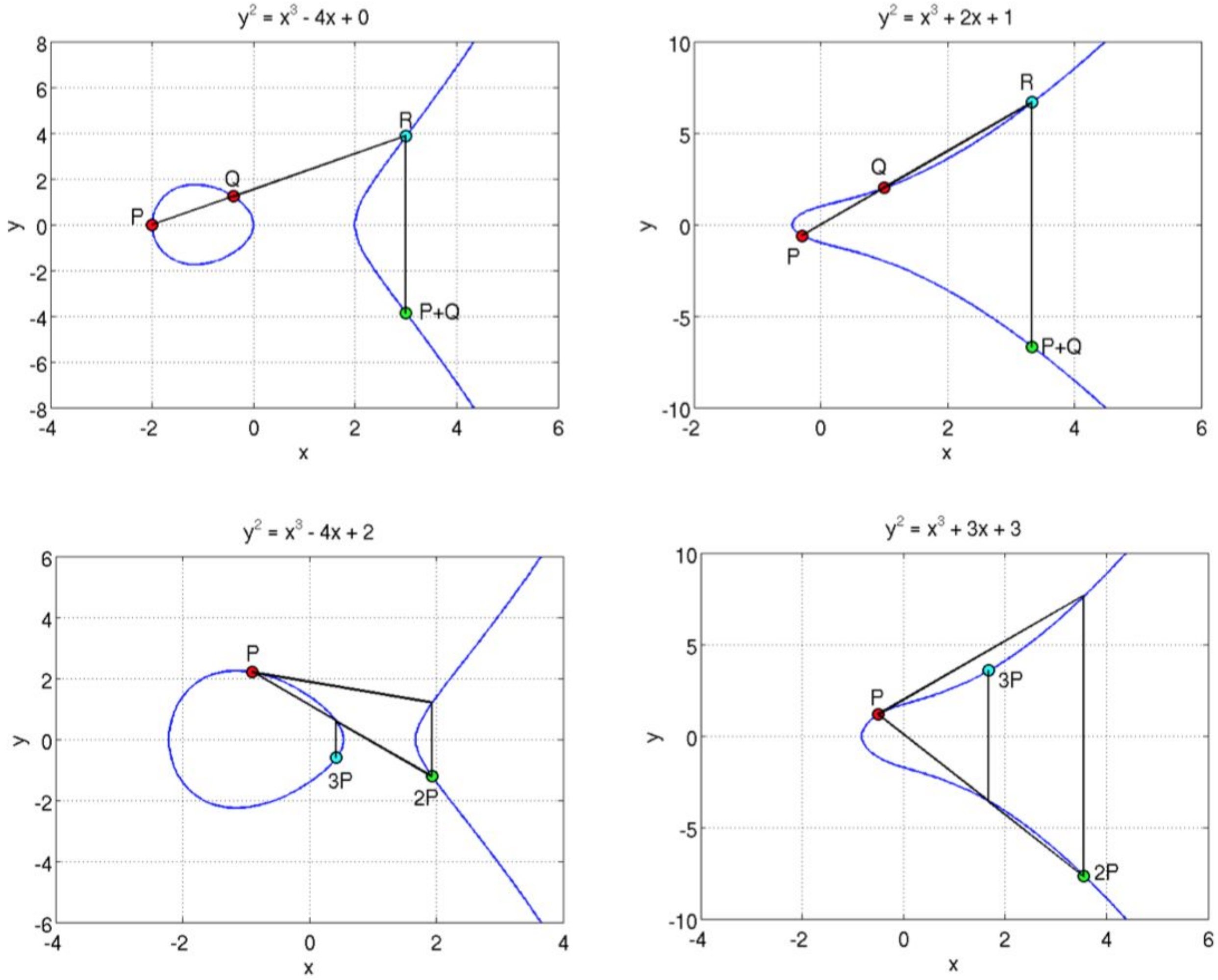
ECC (Elliptic Curve Cryptography)

1. ECC can provide the same level and type of security as RSA (or Diffie-Hellman) but with much shorter keys.
2. Why is ECC so hard to crack?
   A set E of number, C = M × G, G belongs to E
   Assumptions satisfied only when the set E of points is drawn from an elliptic curve
   It is difficult to recover M from C even if we know the curve and G. When an adversary tries to recover M, he/she has to try every possibility from G, 2G, 3G, …
   Given a point G in the set, using the group operator, it is easy to add G to itself k times. However, it is extremely difficult to figure out the value of k even when you know G and the value of the k-fold addition of G to itself.
3. Elliptic curves $y^3 = x^3 + ax + b$ with discriminant $4a^3 + 27b^2$
    a. Only have curves: discriminant < 0
    b. Have both curves and ellipses: discriminant > 0
    c. Singular: discriminant = 0

i. (Singular & has sharp end: x^3 w/ a=0 and b=0)

ii. (Singular & has elliptical end: x^3 w/ nonzero a and b)

4. Non-singular: connect P and Q, fine intersection, and mirror to get P + Q
   Singular: draw tangent line of P, find intersection, and mirror to get 2P

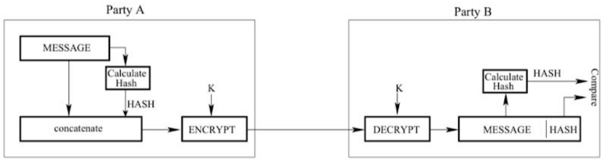


ECDH. How well do you understand the ECDH part of this protocol?

Private key: XA, XB
Public key: YA = XA * G, YB = XB * G
K as calculated by A = XA * XB = K as calculated by B

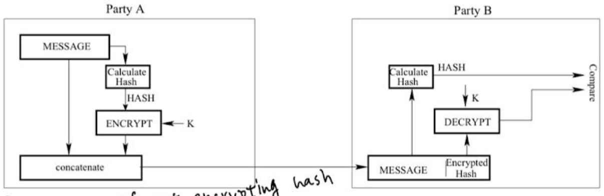
Hashing

1. A hash function takes a variable sized input message and produces a fixed-sized output (hashcode, hash value, or message digest MD).
2. Check for forgeries by comparing MD and the original message



(a)

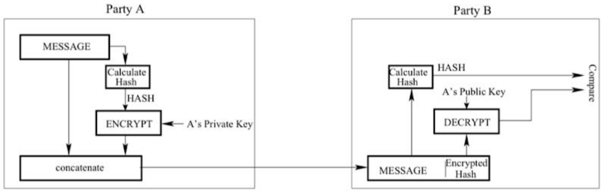
→ Hashcode for authentication
Encryption for confidentiality



(b)

→ For message authentication.

What happen if not encrypting hash
→ someone can change the hash value,

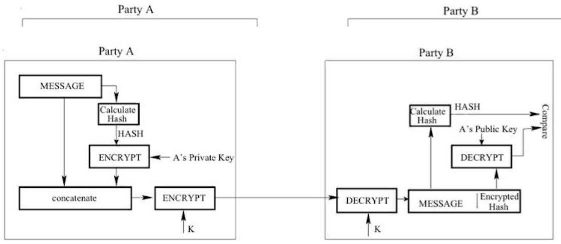


(c)

→ public key encryption
The sender encrypting with private key the hashcode of message constitute the idea of digital structure.

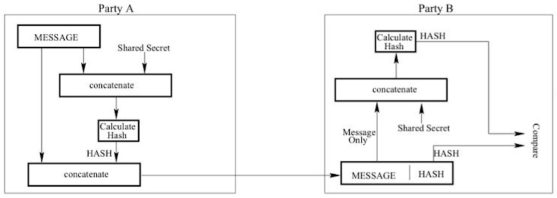
puter and Network Security by Avi Kak Lecture 15

idea of digital structure.



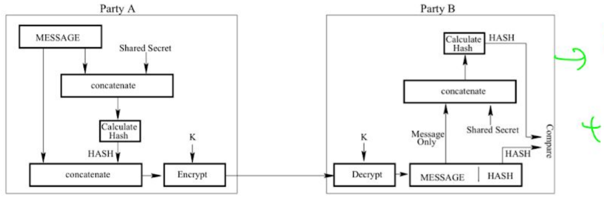
(a)

→ both confidential and authentication



(b)

→ hashcode for authentication



(c)

→ hashcode for authentication
+ confidentiality based transmission between sender & receiver.

3. What would happen if did not encrypt the hash?
   Someone could change both the message and the hash.
4. Cryptographically secure:
   a. One-way property
   b. Strong collision resistance
      i. It is computationally infeasible to find two different messages that hash to the same hashcode value.
      ii. Hash functions that are not collision resistant can fall prey to birthday attack.
5. If you use n bits to represent the hashcode, there are only 2^n distinct hashcode values.
6. In XOR hash algorithm, the hashcode is a longitudinal parity check. XOR hash algorithm is not cryptographically secure because a forgery can be easily created by replacing X_1 through X_{m-1} with any desired Y_1 through Y_{m-1} and then replacing X_m with an $Y_m = Y_1$ ^ Y_2 ^ ... ^ Y_{m-1} ^ the hashcode

Probability Theory

we are interested in knowing whether any of the messages is going to have its hashcode equal to a particular value h.

Consider a pool of k messages produced randomly by the message generator.

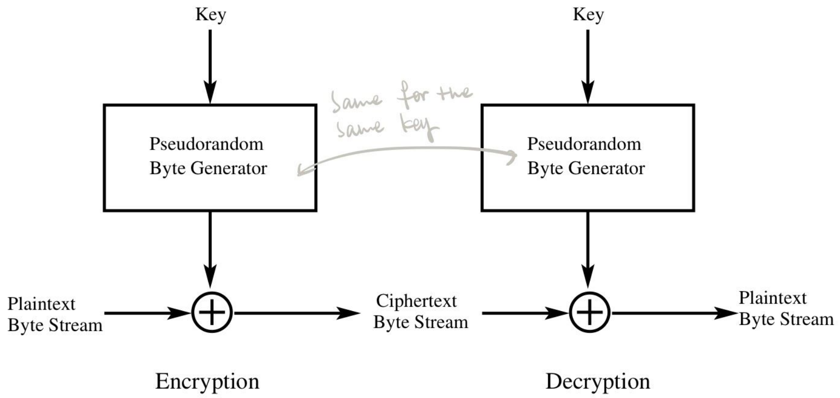
What is the value of k so that the pool contains at least one message whose hashcode is equal to h with probability 0.5?

Given a pool of k randomly produced messages, the probability there will exist at least one message in this pool whose hashcode equals the given value h is k/N.

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.

So if we use an n-bit hashcode, we have $N = 2^n$. In this case, a pool of $2^{n/2}$ randomly generated messages will contain at least one pair of messages with the same hashcode with a probability of 0.5.

Stream Ciphers



1. A typical stream cipher encrypts plaintext one byte at a time.
2. The main processing step is the generation of a stream of pseudorandom bytes that depend on the encryption key.
3. For a given encryption key, the stream of pseudorandom bytes will be the same at the both the encryption end and the decryption end of a data link.
4. For a stream cipher to be secure:
    a. The pseudorandom sequence of bytes should have as long a period as possible.
    b. The sequence should be as random as possible.
    c. The byte sequence should be as uncorrelated as possible.
    d. The encryption key should be as long as possible (128 bits these days).

The RC4 Stream Cipher Algorithm

RC4 is a variable key length stream cipher with byte-oriented operations.
RC4 is vulnerable if the beginning portion of keystream is not discarded.

1. Initialize the state vector S, an array of 256 bytes/integers:
    a. S[i] = i for 0 <= i <= 255
       Result: S[] = {0, 1, 2, …, 254, 255}
2. KSA (Key Scheduling Algorithm):
   The encryption key has no further use in the operation of the stream cipher.
    a. Initialize the vector K.
        i. Suppose we have a 128-bit key.
        ii. Then, K is an array of 16 bytes/integers whose value between 0 and 255.
    b. Initialize the vector T, an array of 256 bytes/integers.
        i. T[i] = K[i % keylen]   for 0 <= i <= 255
           Where keylen is the number of bytes in the encryption key (or the size of K)
    c.

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

3. Generate the pseudorandom byte stream - keystream:

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

4. XOR the plaintext byte with the keystream

WiFi Security Protocols - WEP, WPA, and WPA2

1. WiFi protocols are used for data encryption & user authentication.
   a. Encryption: WEP & WPA use RC4; WPA2 uses AES.
   b. User authentication: PSK (Pre-Shared Key)
      i. WEP: 10 hex digits
      ii. WPA & WPA2: a shared passphrase (deriving from WEP PSK)
   c. WPA2-PSK: WPA2-Personal & WPA2-Enterprise
   WPA2-Enterprise is carried out on a per user basis. It involves 3 agents: a supplicant (a client), an authenticator (AP), and an authentication server (based on EAP)
2. WP2-PSK is vulnerable to KRACK (Key Reinstallation AttaCK). The vulnerability is in the WiFi standard itself. It is in the 4-way handshake.

Vulnerability of RC4 Encryption in WEP and WPA and Why Switch to WPA2

1. No two packets should be encrypted with the same RC4 key in WEP because
   C1 ^ C2 = (P1 ^ S) ^ (P2 ^ S) = P1 ^ P2
2. Differences among WEP, WPA, and WPA2:
   a. WEP
      i. Data integrity check was provided by the ICV value.
      ii. The RC4 key for each packet is a concatenation of a 24-bit IV and the root key.
      iii. The data followed by its ICV value is encrypted.
   b. WPA: TKIP (Temporal Key Integrity Protocol - enhancements over WEP)
      i. Uses 48-bit IV
      ii. Data integrity check: Uses MIC (Message Integrity Check) to check the packet header, and the payload. Also, MIC adds a sequence number field to the wireless frames. Both the payload and its MIC are encrypted.
      iii. Generates the unique starting encryption key for each user authentication
   c. WPA2: CCMP
      i. Data integrity check is carried out by computing the CBC-MAC message authentication code for the packet

ii. CCMP: AES CTR mode and the CBC-MAC based message integrity check

iii. WPA2 separates user authentication services from the services needed for encryption and message integrity.

3. Vulnerabilities of WEP, WPA, and WPA2:
    a. Vulnerability of WEP is that the root key remains fixed for long periods of time and the IV has only 24 bits in it, which implies that the same keystream will be used for different packets in a long session. Also, the IV is sent in plaintext.
    b. WPA suffers from the basic RC4-based weakness as WEP.
    c. WP2-PSK is vulnerable to KRACK (Key Reinstallation AttaCK). The vulnerability is in the WiFi standard itself. It is in the 4-way handshake.

Attacks on WEP

1. Klein Attack
    a. For figuring out the WEP root key
    b. Based on analysis of the pseudorandom sequence: Some plaintext bytes can be XOR'ed with the ciphertext bytes to recover several initial bytes of the pseudorandom sequence
    c. 2 main theorems:
        i. $\text{Prob}(S[j] + S[k] = i \% n) = 2 / n$
        ii. $\text{Prob}(S[j] + S[k] = c \% n) = (n-2) / n(n-1)$
    d. Shortcoming: calculate the key bytes recursively
2. PTW attack: the key bytes are calculated independently