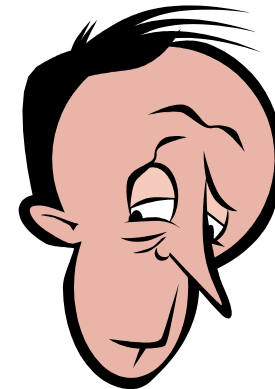
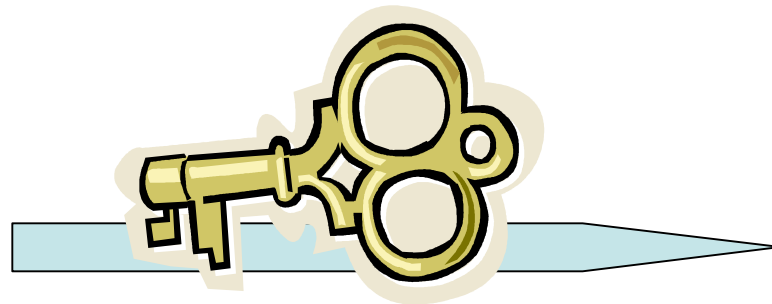


Public Key Cryptography

Introduction

Public Key Cryptography

- Unlike symmetric key, there is no need for Alice and Bob to share a common secret
 - Alice can convey her public key to Bob in a public communication:



Encrypting w/ Public Keys

- Public key schemes encrypt large blocks of data:
 - Smallest system with reasonable security has block sizes at least 160 bits (Elliptic Curves)
 - Key size generally equal to or close to block size
 - Orders of magnitude less efficient than symmetric key encryption

Why public key?

- The reason public keys are used is to establish secure communication when there is no way to exchange a key beforehand.
 - Confidential/authenticated channels for free?
- Must ensure that the public key belongs to the correct party (binding of identity to key). The public key directory may be corrupted:
 - Solution: Use a Public Key Infrastructure to certify your keys (PKI)

Encryption: Details

- Alice knows Bob's public key P_{Bob}
- Uses the encryption algorithm:
 - $\text{Enc}(P_{\text{Bob}}, \text{Message}) = C$
- Anybody may encrypt messages that only Bob may read, since he knows the private key S_{Bob}
- $\text{Message} = \text{Dec}(S_{\text{Bob}}, C)$

How does Bob know S_{Bob} ?

- How did Bob come to know his private key to start with?
 - The answer is that Bob generates the pair $(P_{\text{Bob}}, S_{\text{Bob}})$ jointly. The key generation procedure is probabilistic and one-way.
 - The security of such methods is closely related to a class of mathematical problems from modular arithmetic

The discrete logarithm problem

- Let p be a prime number
 - A large one, say 1000 -- 2000 bits long.
- Take g to be in the interval $[2, p-2]$.
- Consider the exponential function:
 - $Exp(\bullet, \text{mod } p): x \rightarrow g^x \text{ mod } p$
- $Exp(\bullet, \text{mod } p)$ is hard to invert.
 - unless p is a weak prime (rare case and easy to test for)

Encrypting a la ElGamal

- Take p a safe prime:
 - $p = 2q + 1$, with q also prime.
 - This guarantees $Exp(\bullet, \text{mod } p)$ is hard.
- Take g' in $[2, p-2]$ and choose
 - $g = g'^2 \text{ mod } p$.
- Choose private key k at random
 - k in $[2, q-1]$
- Compute the public key $y = g^k \text{ mod } p$.

ElGamal Encryption

- To encrypt m in $[1, p-1]$ for user Bob:
 - public key $y = g^k$, private key k
- Compute a random value r
- Compute
 - $(A, B) = (g^r \text{ mod } p, m y^r \text{ mod } p)$
- To decrypt, Bob computes
 - $m = B(A^{-k}) \text{ mod } p$

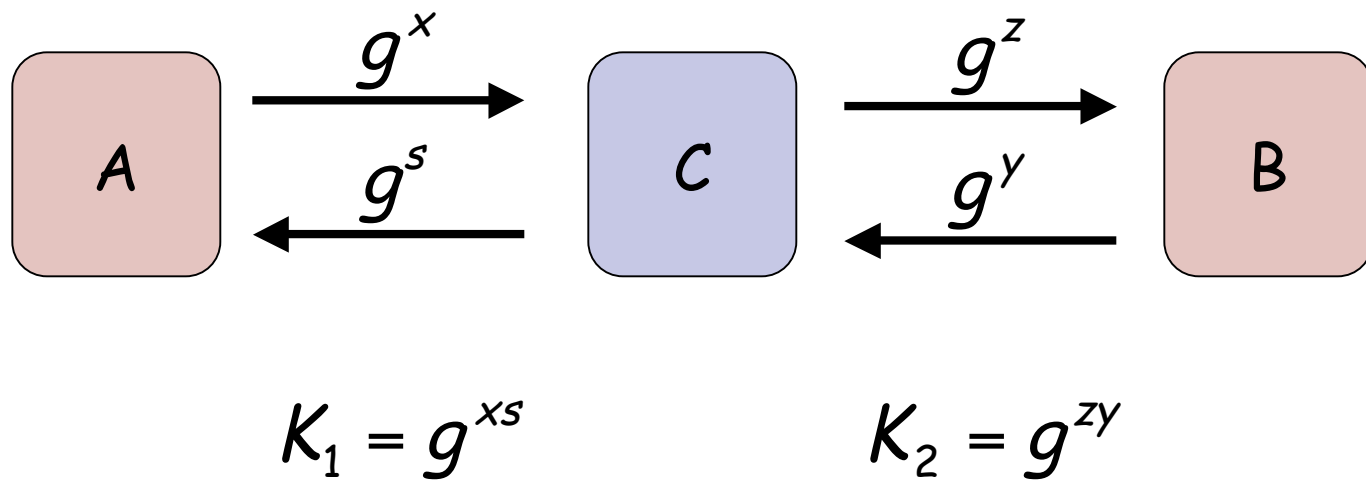
Hybrid scheme

- Use the public key scheme to share a session key, then use the session key with a symmetric encryption scheme:
 - Alice to Bob: $Enc(P_{Bob}, R_1)$
 - Bob to Alice: $Enc(P_{Alice}, R_2)$
 - Use $K = R_1 \oplus R_2$ for the session key
 - No forward security, if you recover a private key can get older session keys

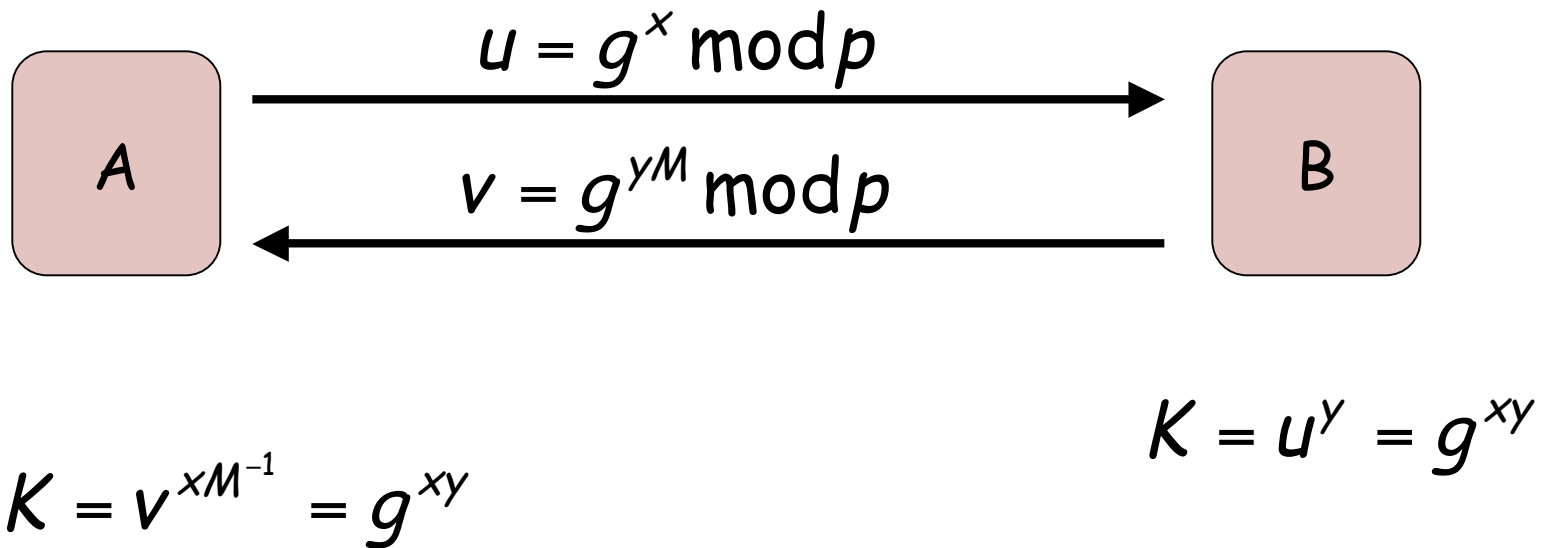
Better:

- Alice to Bob:
 - $g^a \text{ mod } p$, with a random
- Bob to Alice:
 - $g^b \text{ mod } p$, with b random
- Session key derived from shared secret, but without authentication:
 - $g^{ab} \text{ mod } p$

Man-in-the-middle attack

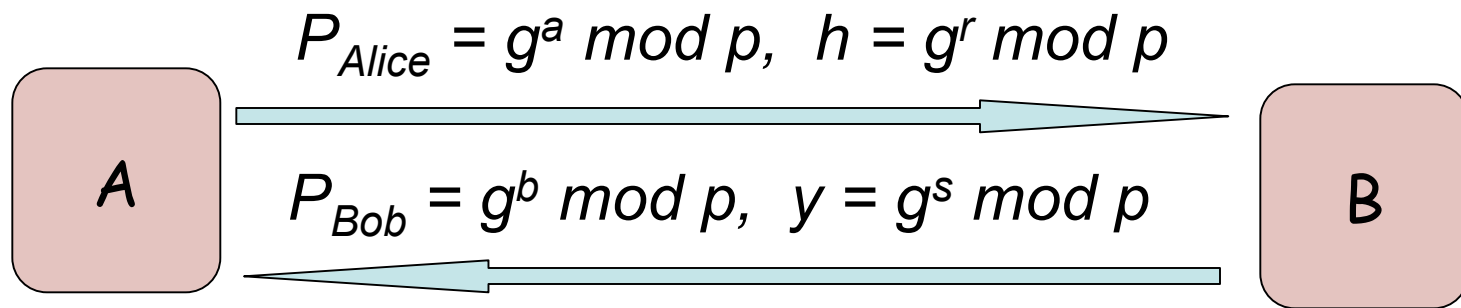


Adding authentication



Here, $g^M = P_{Alice}$, the public key of Alice.

MTI: Authenticated DH



$$K = (P_{Bob})^r y^a \text{ mod } p$$

$$K = (P_{Alice})^s h^b \text{ mod } p$$

$$K = g^{sa + rb} \text{ mod } p$$

DSA keys

- Generate large prime $p = kq + 1$,
 - p originally 512 bits, today 1024 or more
 - q originally 160 bits (still safer today).
- Generator g such that $g^q = 1 \pmod{p}$.
 - Take $h \in [1, p - 1]$; set $g = h^{(p-1)/q} \pmod{p}$
- Choose private-public key pair: $\langle T, S \rangle$
 - S random in $[1, q]$; $T = g^S \pmod{p}$

Signing w/ DSA

- Generate a per-message private/public key pair:
 - $\langle T_m, S_m \rangle : T_m = g^{S_m} \text{ mod } p$
- d_m = digest of message (e.g., SHA-1)
- Compute the signature
 - $X = S_m^{-1} (d_m + S T_m) \text{ mod } q$
- The signing pair is (T_m, X)

Verifying the DSA

- Calculate the inverse of X :
 $-X^{-1} \text{ mod } q$
- Calculate d_m from the message m
- Compute $a = d_m X^{-1} \text{ mod } q$
- Compute $b = T_m X^{-1} \text{ mod } q$
- Compute $z = (g^a T^b \text{ mod } p) \text{ mod } q$
- If $z = T_{m'}$, signature verifies correctly.