Lecture 14 Key Exchange

A Scenario

- Alice and Bob want to communicate with each other with symmetric encryption.
- How to distribute the shared secret key between Alice and Bob?
  - Secret key=encryption key=decryption key

Diffie-Hellman Key Exchange

- public-key type scheme
  - proposed in 1976
  - *note*: now know that Williamson (UK CESG) secretly proposed the concept in 1970
- A practical method for public exchange of a secret key
- Used in a number of commercial products
- Turing Award 2016

Diffie-Hellman Key Exchange

- public-key distribution scheme
  - cannot be used to exchange an arbitrary message
  - rather it can establish a common key

- **security relies on the difficulty of computing discrete logarithms**
  - Recall RSA

Diffie-Hellman Setup

- all users agree on global parameters:
  - large prime integer or polynomial **p**
  - **g** = primitive root mod p
    - for every integer *a* that has gcd(*a*, *p*) = 1, there is an integer *k* such that $g^k \equiv a \pmod{p}$
- each user generates their key
  - Alice (Bob) chooses a secret key (number): a (b) < p
  - Alice (Bob) compute their public key: $A = g^a \bmod p$ $(B = g^b \bmod p)$

Diffie-Hellman Key Exchange

- Alice sends $A = g^{a} \bmod p$ to Bob
- Bob sends $B = g^{b} \bmod p$ to Alice
- shared session key for users is K_{AB}:
  - $K_{AB} = g^{ab} \bmod p$

    $= A^{b} \bmod p$ (which Bob can compute)

    $= B^{a} \bmod p$ (which Alice can compute)

Diffie-Hellman Key Exchange

- Once Alice and Bob obtain K_{AB}:
  - They can use it as the shared secret key for symmetric encryption direclty

Diffie-Hellman Key Exchange

- g can be small
  - 2 or 5 is common
- a, b, p should be large
- attacker needs a or b to obtain the session key
  - $A = g^a \bmod p$ and $B = g^b \bmod p$
  - must solve discrete logarithm (not logarithm)

Diffie-Hellman Example

- Alice & Bob who wish to establish a shared secret key
  - agree on prime $p=353$ and $g=3$
- select random secret keys:
  - A chooses $a=97$, B chooses $b=233$
- compute respective public keys:
  - $A=3^{97} \bmod 353 = 40$ (Alice)
  - $B=3^{233} \bmod 353 = 248$ (Bob)
- compute shared session key as:
  - $K_{AB} = B^{a} \bmod 353 = 248^{97} = 160$ (Alice)
  - $K_{AB} = A^{b} \bmod 353 = 40^{233} = 160$ (Bob)

Why Discrete Log is Difficult to Solve ?

- Something to do with approximation
- The calculators tells us that base-10 logarithm of 17 is 1.2304489 – it is only an approximation result, as $10^{1.2304489}$=16.999999
  - In fact, the base-10 logarithm of 17 can't be represented exactly using a finite number of digits

Why Discrete Log is Difficult to Solve ?

- Why we are satisfied with this result?
  - We know that base-10 logarithm of 17 is very close to 1.2304489, given that 10 to the power of 1.2304489 is very close to 17
  - Besides, we can infer that the exact base-10 logarithm of 17 is larger than 1.2304489
- However, we do not have such estimation in modular arithmetic

Why Discrete Log is Difficult to Solve ?

- To calculate k such that

$$2^k \equiv 88319671 \pmod{97330327}$$

- By trying different exponents, the exponent 28305819 happens to lead to a different by similar number

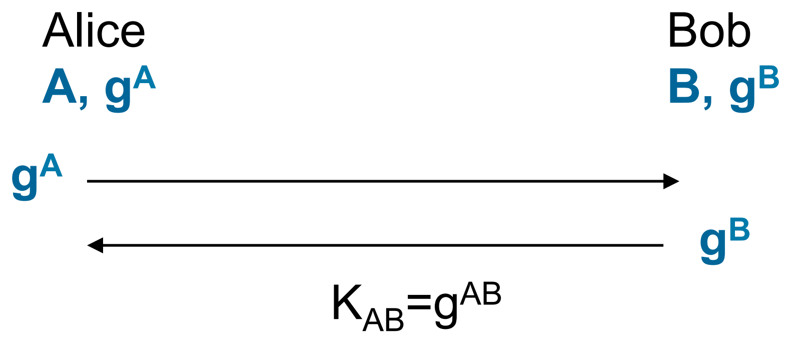
$$2^{28305819} \equiv 88032151 \pmod{97330327}$$

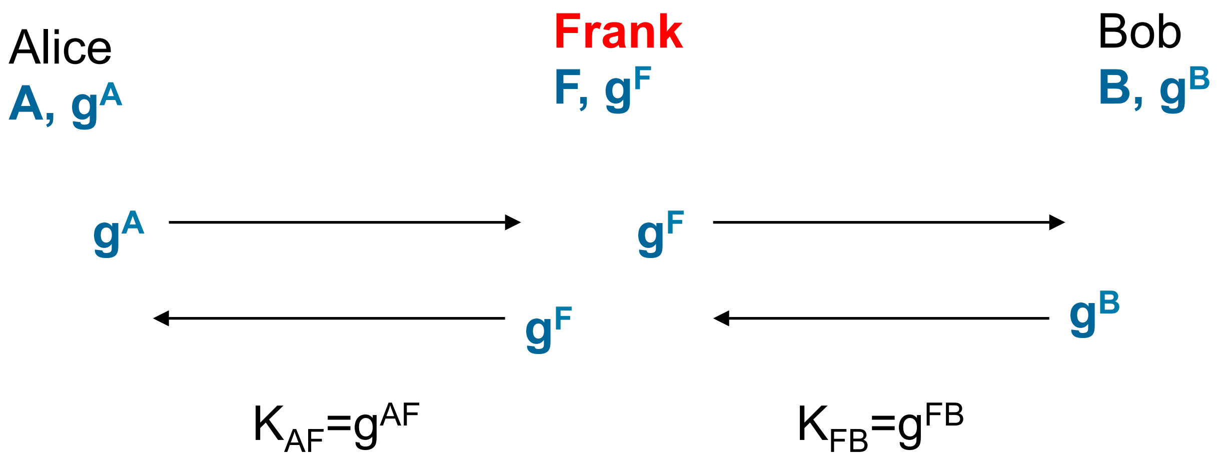
  - 28305819 might be a good approximation to the log of 88319671
  - 28305819 might be smaller than the exact logarithm of 88319671

Why Discrete Log is Difficult to Solve ?

- However, the true logarithm is 12314, which is very different from our guess of 28305819
- Conclusion: in modular arithmetic in contrast to ordinary arithmetic, just because two numbers are close does not mean their logarithms are close. To calculate discrete log, one needs to enumerate all the possible values

Man-in-the-middle-attack





Countermeasures

- Key idea: to ensure that $A = g^a \bmod p$ ($B= g^b \bmod p$) indeed comes from Alice (Bob)
- Need some mechanism to bind pubic keys to the identities of entities
- A certificate authority (CA) certifies that A (B) is indeed the public key of Alice (Bob)
  - CA signs the public keys of Alice and Bob: $Sig_{CA}(A)$ and $Sig_{CA}(B)$
  - Alice (Bob) obtains B (A) by decrypting $Sig_{CA}(B)$ ($Sig_{CA}(A)$) with CA's public decryption key