Diffie-Hellman Key Exchange

- first public-key type scheme proposed
- by Diffie & Hellman in 1976 along with the exposition of public key concepts
- is a practical method for public exchange of a <u>secret key</u>
- used in a number of commercial products

Diffie-Hellman Key Exchange

- a public-key distribution scheme
  - cannot be used to exchange an arbitrary message
  - rather it can establish a <u>common key</u> known only to the two participants
- value of key depends on the participants (and their private and public key information)
- based on exponentiation in a finite (Galois) field (modulo a prime or a polynomial) - easy
- security relies on the difficulty of computing discrete logarithms (similar to factoring) – hard

Diffie-Hellman Setup

- all users agree on global parameters:
  - large prime integer or polynomial q
  - a being a primitive root mod q
    - a number whose powers successively generate all the elements mod q
- each user (eg. A) generates their key
  - chooses a secret key (number): $x_A < q$
  - compute their **public key**: $y_A = a^{x_A} \bmod q$
  - each user makes public that key y_A

Diffie-Hellman Key Exchange

- shared session key for users A & B is K_{AB}:

  $K_{AB} = a^{x_A . x_B} \bmod q$

  $= y_A^{x_B} \bmod q$ (which **B** can compute)

  $= y_B^{x_A} \bmod q$ (which **A** can compute)
- K_{AB} is used as session key in private-key encryption scheme between Alice and Bob
- if Alice and Bob subsequently communicate, they will have the **same** key as before, unless they choose new public-keys
- attacker needs an x, must solve discrete log

Diffie-Hellman Example

- users Alice & Bob who wish to swap keys:
- agree on prime $q=353$ and $a=3$
- select random secret keys:
  - A chooses $x_A=97$, B chooses $x_B=233$
- compute respective **public** keys:
  - $y_A=\mathbf{3}^{97} \bmod 353 = 40$ (Alice)
  - $y_B=\mathbf{3}^{233} \bmod 353 = 248$ (Bob)
- compute **shared session** key as:
  - $K_{AB}= y_B^{x_A} \bmod 353 = 248^{97} = 160$ (Alice)
  - $K_{AB}= y_A^{x_B} \bmod 353 = 40^{233} = 160$ (Bob)

Key Exchange Protocols

- users could create <u>random</u> private/public D-H keys each time they communicate
- users could create a known private/public D-H key and publish in a directory, then consulted and used to securely communicate with them
- both of these are vulnerable to **Meet-in-the-Middle Attack**
- authentication of the keys is needed

Man-in-the-Middle Attack

1. Darth prepares for the attack by generating two random private keys X_{D1} and X_{D2} and then computing the corresponding public keys Y_{D1} and Y_{D2}
2. Alice transmits Y_A to Bob.
3. Darth intercepts Y_A and transmits Y_{D1} to Bob. Darth also calculates K2 = (Y_A)^ X_{D2} mod q
4. Bob receives Y_{D1} and calculates K1=(Y_{D1})^ X_B mod q
5. Bob transmits Y_B to Alice.
6. Darth intercepts Y_B and transmits Y_{D2} to Alice. Darth calculates K1=(Y_B)^ X_{D1} mod q
7. Alice receives Y_{D2} and calculates K2=(Y_{D2})^ X_A mod q .

Man-in-the-Middle Attack

- Bob and Alice think that they share a secret key, but instead
  - Bob and Darth share secret key K1 and
  - Alice and Darth share secret key K2.
- All future communication between Bob and Alice is compromised in the following way:
  1. Alice sends an encrypted message M: E(K2, M).
  2. Darth intercepts the encrypted message and decrypts it, to recover M.
  3. Darth sends Bob E(K1, M) or E(K1, M'), where M' is any message.

In (2), Darth simply wants to eavesdrop on the communication without altering it.

In (3), Darth wants to modify the message going to Bob.

ElGamal Cryptography

- public-key cryptosystem related to D-H
- so uses exponentiation in a finite (Galois)
- with security based difficulty of computing discrete logarithms, as in D-H
- each user (eg. A) generates their key
  - chooses a secret key (number): $1 < x_A < q-1$
  - compute their **public key**: $y_A = a^{x_A} \bmod q$

ElGamal Message Exchange

- Bob encrypt a message to send to A computing
  - represent message M in range $0 <= M <= q-1$
    - longer messages must be sent as blocks
  - chose random integer k with $1 <= k <= q-1$
  - compute <u>one-time key</u> $K = y_A^k \bmod q$
  - encrypt M as a pair of integers (C_1, C_2) where
    - $C_1 = a^k \bmod q$; $C_2 = KM \bmod q$
- A then recovers message by
  - recovering key K as $K = C_1^{x_A} \bmod q$
  - computing M as $M = C_2 K^{-1} \bmod q$
- a unique k must be used each time
  - otherwise result is insecure

ElGamal Example

- use field GF(19) $q=19$ and $a=10$
- Alice computes her key:
  - A chooses $x_A=5$ & computes $y_A=10^5 \bmod 19 = 3$
- Bob sends message $m=17$ as $(11,5)$ by
  - chosing random $k=6$
  - computing $K = y_A^k \bmod q = 3^6 \bmod 19 = 7$
  - computing $C_1 = a^k \bmod q = 10^6 \bmod 19 = 11$;

    $C_2 = KM \bmod q = 7.17 \bmod 19 = 5$
- Alice recovers original message by computing:
  - recover $K = C_1^{x_A} \bmod q = 11^5 \bmod 19 = 7$
  - compute inverse $K^{-1} = 7^{-1} = 11$
  - recover $M = C_2 K^{-1} \bmod q = 5.11 \bmod 19 = 17$