Diffie-Hellman Key Exchange

- first public-key type scheme proposed
- by Diffie & Hellman in 1976 along with the exposition of public key concepts
  - note: now know that James Ellis (UK CESG) secretly proposed the concept in 1970
- is a practical method for public exchange of a secret key
- 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 common key
  - 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

Global Public Elements

q	prime number
α	$\alpha < q$ and α a primitive root of q

User A Key Generation

Select private X_A	$X_A < q$
Calculate public Y_A	$Y_A = \alpha^{X_A} \bmod q$

User B Key Generation

Select private X_B	$X_B < q$
Calculate public Y_B	$Y_B = \alpha^{X_B} \bmod q$

Generation of Secret Key by User A

$K = (Y_B)^{X_A} \bmod q$

Generation of Secret Key by User B

$K = (Y_A)^{X_B} \bmod q$

Figure 10.7 The Diffie-Hellman Key Exchange Algorithm

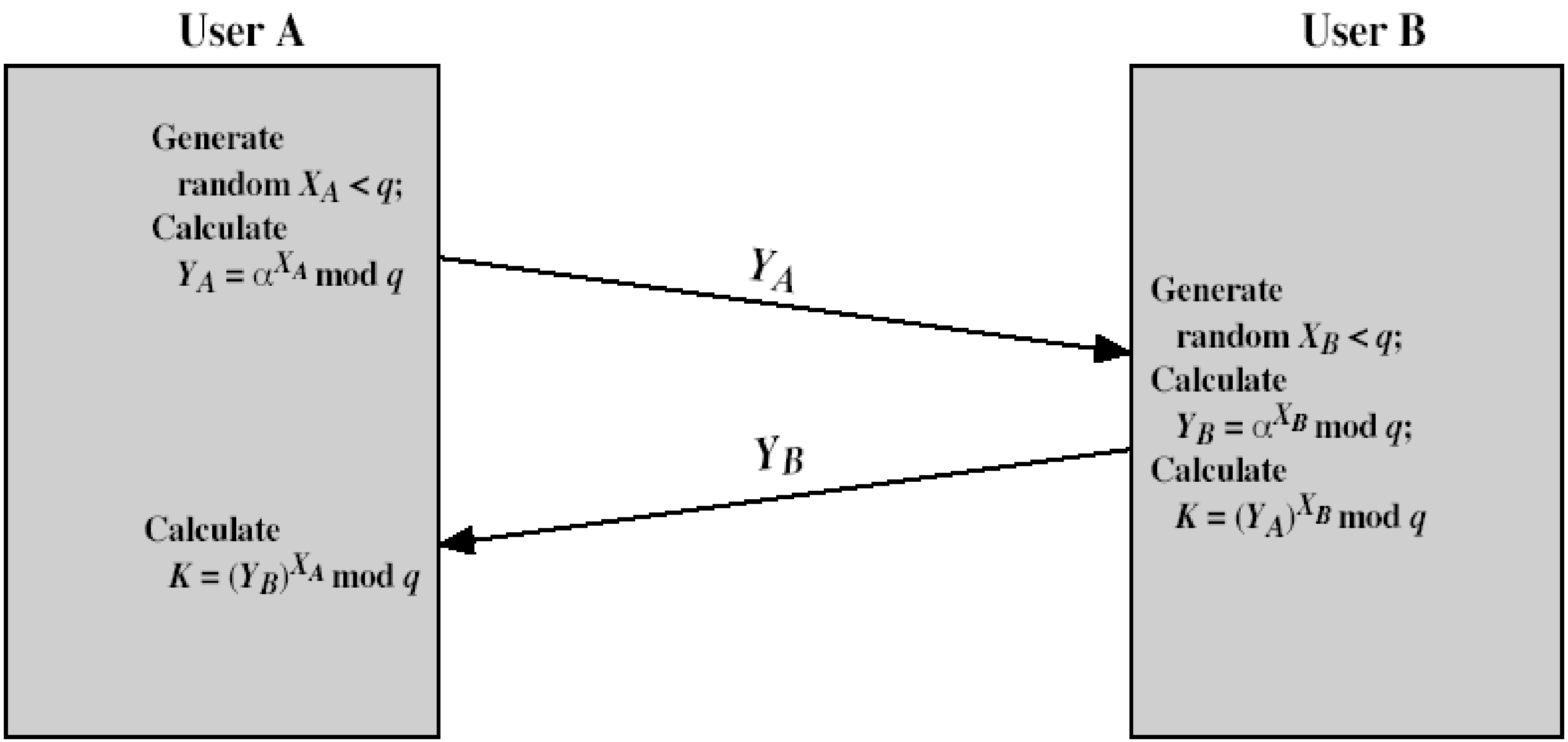

Figure 10.8 Diffie-Hellman Key Exchange

Diffie-Hellman Example

- users Alice & Bob who wish to swap keys:
- agree on prime $q=353$ and $\alpha=3$
- select random secret keys:
  - A chooses $x_A=97$, B chooses $x_B=233$
- compute 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:

$K_{AB}= y_B^{x_A} \bmod 353 = \mathbf{248}^{97} = 160$ (Alice)

$K_{AB}= y_A^{x_B} \bmod 353 = \mathbf{40}^{233} = 160$ (Bob)

Elgamal encryption algorithm

Prime p and generator g are public keys of Bob

Alice:

Bob:

+ public key

$Y = g^{x} \bmod p$

Chooses private x

Chooses random k

Calculates

$K = Y^{k} \bmod p$

calculates

$C_1 = g^{k} \bmod p$

$C_2 = M K \bmod p$

(C_1, C_2)

Calculates $C_1^{x} \bmod p$

$= K$

and recovers message

$M = K^{-1} C_2 \bmod p$

K^{-1} = the inverse of K mod p

Elgamal = Diffie Hellman key exhange + encryption by multiplying mod p

Elgamal example

Alice sends a message M = 100 to Bob

Prime p = 139 and g =3

Alice:

public key

$44 = 3^{12} \bmod 139$

Bob:

Chooses k = 52

Chooses private x = 12

Calculates

$K = 44^{52} \bmod 139 = 112$

Calculates

$C_1 = 3^{52} \bmod 139 = 38$

$C_2 = 100 * 112 \bmod 139 = 80$

$(C_1, C_2) = (39, 80)$

Calculates $K = 38^{12} \bmod 139 = 112$

$K^{-1} = 112^{-1} \bmod 139 = 36$

and recovers message

$M = K^{-1} C_2 \bmod p =$

$36 * 80 \bmod 139 = 100$

Elgamal = Diffie Hellman key exhange + encryption by multiplying mod p

Elgamal security

- Each user has a private key x
- Each user has three public keys: prime modulus p, generator g and public $Y = g^x \bmod p$
- Security is based on the difficulty of DLP
- Secure key size > 1024 bits (today even 2048 bits)
- Elgamal is quite slow, it is used mainly for key authentication protocols
- Now widely used, but Elliptic Curve variant is increasingly popular