# Troca de Chaves Diffie-Helman

- Algoritmo

**Global Public Elements**

| | |
|---|---|
| $q$ | prime number |
| $\alpha$ | $\alpha < q$ *and* $\alpha$ 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 $Y_B$ | $Y_B = \alpha^{X_B} \bmod q$ |

**Calculation of Secret Key by User A**

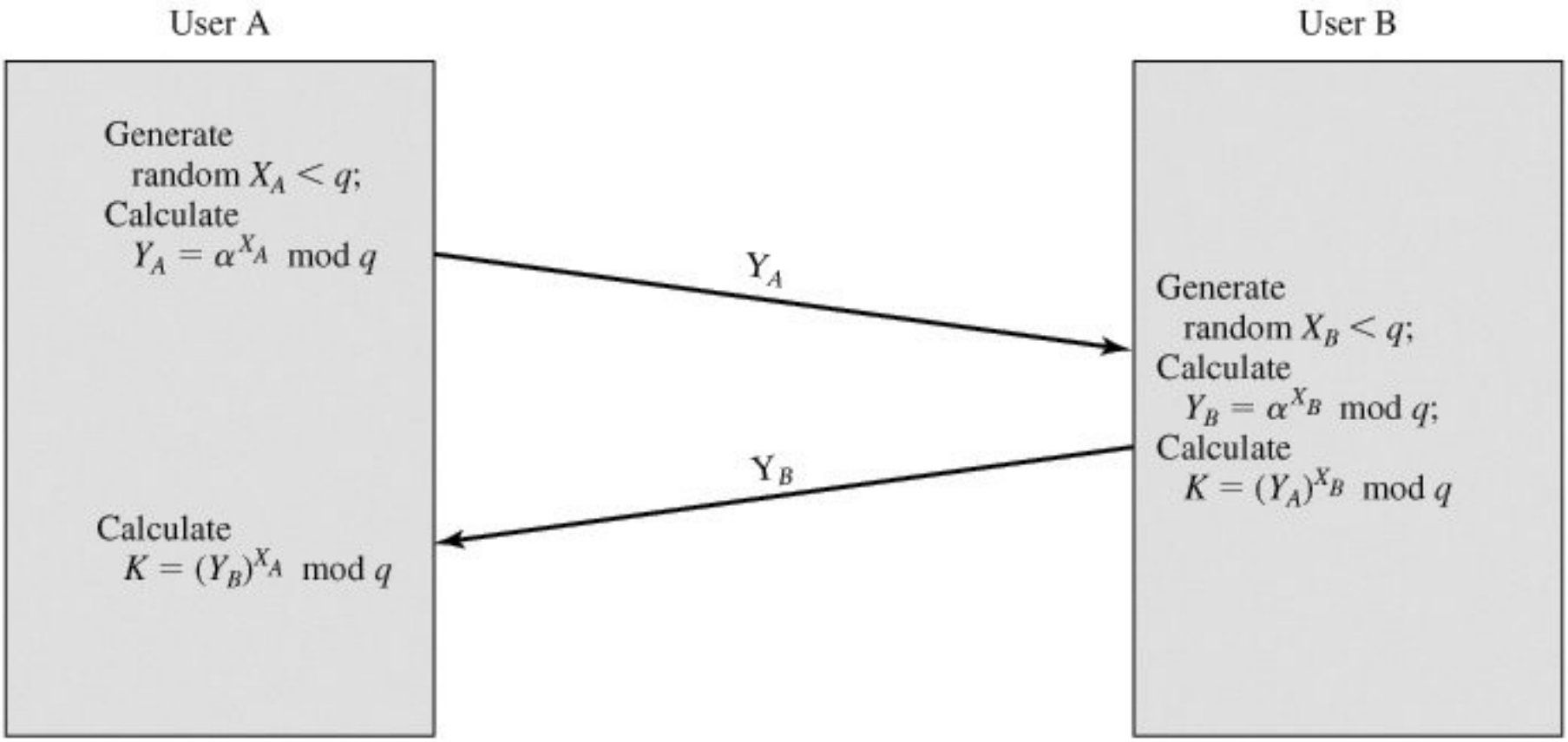
$K = (Y_B)^{X_A} \bmod q$

**Calculation of Secret Key by User B**

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

# Troca de Chaves Diffie-Helman

- Seqüência de Operações



# Troca de Chaves Diffie-Hellman - Ataque

- Ataque Man-in-the-Midlle
  - 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 $Y_{D2}$.
  - Alice transmits $Y_A$ to Bob.
  - Darth intercepts $Y_A$ and transmits $Y_{D1}$ to Bob. Darth also calculates K2 = $(Y_A)^{X}_{D2}$ mod q.
  - Bob receives $Y_{D1}$ and calculates K1 = $(Y_{D1})^{X}_{E}$ mod q.
  - Bob transmits $Y_B$ to Alice.
  - Darth intercepts $Y_B$ and transmits $Y_{D2}$ to Alice. Darth calculates K1 = $(Y_B)^{X}_{D1}$ mod q.
  - Alice receives $Y_{D2}$ and calculates K2 = $(Y_{D2})^{X}_{A}$ mod q.
- Bob a Alice neste ponto imaginam que compartilham a chave secreta, porém Darth possui tanto K1 quanto K2, compromentendo a comunicação da seguinte forma:
  - Alice sends an encrypted message M: E(K2, M).
  - Darth intercepts the encrypted message and decrypts it, to recover M. (passivo)
  - Darth sends Bob E(K1, M) or E(K1, M'), where M' is any message (ativo)
- Troca de chave não realiza a auteniticação dos participantes
- Pode ser resolvido com assinatura digital e certificados de chave pública

# JCA – Troca de Chaves

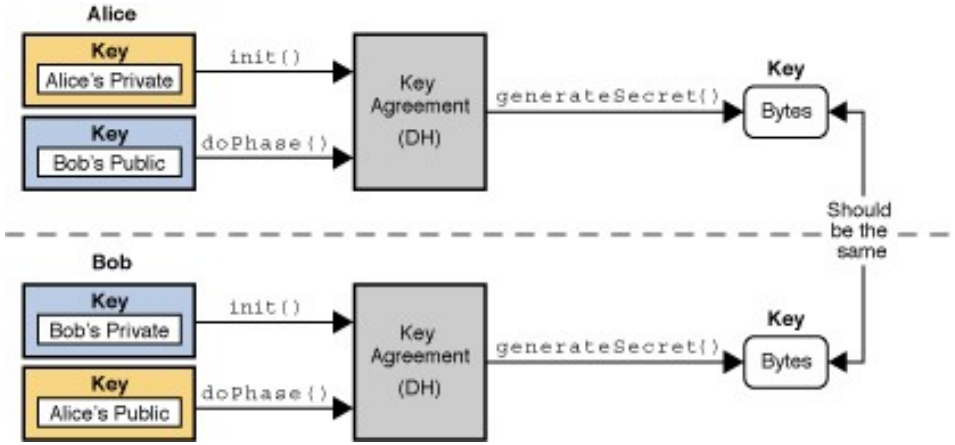
- A classe KeyAgreement permite que duas partes estabeleçam uma mesma chave sem a necessidade do envio desta chave entre estas partes
- Esta classe implementa os seguintes protocolos para troca de chaves:

| NOME ALGORITMO | DESCRIÇÃO |
|---|---|
| DiffieHellman | Diffie-Hellman Key Agreement as defined in PKCS3: Diffie-Hellman Key-Agreement Standard, RSA Laboratories, version 1.4, November 1993. |
| ECDH | Elliptic Curve Diffie-Hellman as defined in ANSI X9.63 and as described in RFC 3278: "Use of Elliptic Curve Cryptography (ECC) Algorithms in Cryptographic Message Syntax (CMS)." |
| ECMQV | Elliptic Curve Menezes-Qu-Vanstone as defined in "Elliptic Curve Cryptography" from www.secg.org. |

- Inicialmente cada parte cria um par de chaveS, sendo uma pública e a outra privada
- Cada parte inicializa o objeto KeyAgreement com sua chave privada
- As chaves públicas devem então ser trocadas. Normalmente são duas partes, porém algoritmos como Diffie-Hellman permitem a participação de várias partes (3 ou mais)
- A partir das chaves públicas recebidas cada objeto KeyAgreement, existente em cada parte, produzirá a mesma chave secreta

# JCA – Troca de Chaves KeyAgreement - Uso

- Configuração – Outras classes
  - Estabelecer Parâmetros Algoritmo
    - Classe DHParameterSpec




  - Geração Chaves (KeyPairGenerator)
    - Iniciar gerador de chaves com os parâmetros do algoritmo
      - java.security.KeyPairGenerator initialize(DHParameterSpec)
    - Gerar o par de chaves
- Uso Classe KeyAgreement
  - Obtenção do objeto informando o algoritmo
    - javax.crypto.KeyAgreement.getInstance(String algorithm)
  - Iniciar cada objeto com a sua chave privada
    - javax.crypto.KeyAgreement .init(Key privatekey)
  - Informar chave publica das outras partes.
    - javax.crypto.KeyAgreement . doPhase(Key publickey, boolean lastPhase)
  - Gerar chave screta - byte[] javax.crypto.KeyAgreement.generateSecret()