Cryptography

Diffie-Hellman Key Exchange and ElGamal Cryptosystem

Introduction to Cryptography and Network Security – Behrouz A. Frouzan

Cryptography and Network Security Principals and Practices – Willam Stallings

Serious Cryptography – Jean Phillip Aumasson

Content

Content
Exponentiation and Logarithms
Diffie-Hellman Key Exchange
Diffie-Hellman Protocols
ElGamal Cryptosystem

Exponentiation and Logarithms

- Exponentiation and logarithms are the inverse functions of each other

Exponentiation: $y = a^x$ → **Logarithms**: $x = \log_a y$

  - a is called the base of the logarithm
- In cryptography, we use **modular exponentiation**:

$$y = a^x \bmod n$$

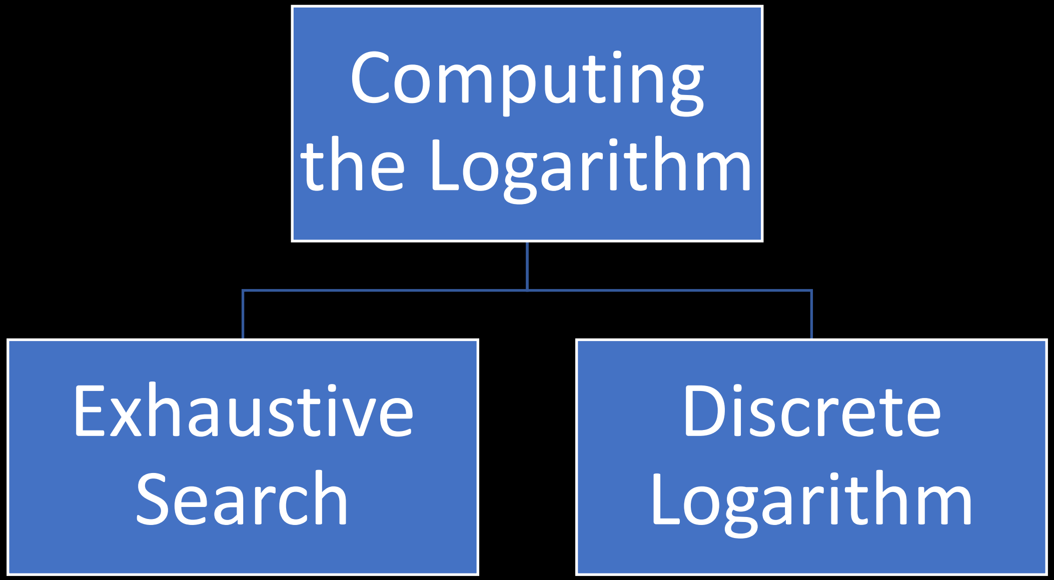
- This is used for encryption/decryption
  - E.g., ElGamal

Exponentiation and Logarithms

- **Exponentiation** for **encryption** → **logarithms** for **breaking a cipher**
- HOW HARD IS IT?

Exponentiation and Logarithms

- **Exponentiation** for **encryption** → **logarithms** for **breaking a cipher**
- HOW HARD IS IT?
- Two ways to reverse the exponentiation → $x = \log_a y \ (mod\, n)$:



Exponentiation and Logarithms

- Exhaustive search = brute force
- Try all possible values of x that satisfy $y = a^x \ mod \ n$
  - Very inefficient!

```
Modular_Logarithm (a, y, n)
{
    for (x = 1 to n −1)
    {
        if (y ≡ a^x mod n) return x
    }
    return failure
}
```

Exponentiation and Logarithms

- The second way is using **discrete logarithm**.
- To understand the discrete logarithms, we need to learn some concepts:

Discrete Logarithm
Finite multiplicative group
Order of the group
Order of an element
Euler's Theorem
Primitive roots
Cyclic group

Exponentiation and Logarithms

Finite Multiplicative Group

Exponentiation and Logarithms

Finite Multiplicative Group $G =< Z_n, \boxed{\cdot} >$

- A <u>group</u> is a set of elements with a binary operation "$\boxed{\cdot}$"
  - The elements are modulo n
- It has four properties:

Axiom	Relation	Meaning
Closure	$a, b \in G \ \rightarrow c = a \boxed{\cdot} b \in G$	The result of applying the operation on two elements in the set is another element in the set
Associativity	$a, b, c \in G \ \rightarrow (a \boxed{\cdot} b) \boxed{\cdot} c = a \boxed{\cdot} (b \boxed{\cdot} c)$	The order of operations doesn't matter
Existence of identity	$\forall a \in G, \exists e \mid e \boxed{\cdot} a = a$	Every element has an identity element
Existence of inverse	$\forall a \in G, \exists a` \mid a \boxed{\cdot} a` = 1$	Every element has an inverse element

Exponentiation and Logarithms

Finite Multiplicative Group $G =< Z_n^*, \times >$

- A group is a set of elements with a binary operation "×"
  - The elements are modulo n
  - The integers in the set are relatively prime to n
- It has four properties:

Axiom	Relation	Meaning
Closure	$a, b \in G \ \rightarrow c = a \times b \in G$	The result of applying the operation on two elements in the set is another element in the set
Associativity	$a, b, c \in G \ \rightarrow (a \times b) \times c = a \times (b \times c)$	The order of operations doesn't matter
Existence of identity	$\forall a \in G, \exists e \mid e \times a = a$	$e = 1$
Existence of inverse	$\forall a \in G, \exists a` \mid a \times a` = 1$	$a` = EGCD(a, n)$

Exponentiation and Logarithms

Discrete Logarithm
Finite multiplicative group
Order of the group
Order of an element
Euler's Theorem
Primitive roots
Cyclic group

- Example: let $n = 7$:
- So $G =< Z_7^*, \times> = \{1, 2, 3, 4, 5, 6\}$

Axiom	**Relation**
Closure	$(4\times5) mod\ 7 = 6$
Associativity	$[(2+3)+4]\ mod\ 7 = [2+(3+4)] mod\ 7 = 2$
Existence of identity	$(5\times1) mod\ 7 = 5$
Existence of inverse	$(4\times2) mod\ 7 = 1\ \mid egcd(4, 7) = 1$

Exponentiation and Logarithms

Order of the group

- $|G|$ = number of elements in the group
- **Example**: $G = < Z_7^*, \times > = 6$

How to find the order of the group with modulus n?

- Note that the order of the group doesn't have to be $n - 1$.
  - We need to count the number of elements that are relatively prime with n.

Exponentiation and Logarithms

Order of the group

- $|G|$ = number of elements in the group
- **Example**: $\boldsymbol{G} =< \boldsymbol{Z_7^*}, \times > = 6$

- If the group modulus n can be factored into prime factors, we can use Euler's totient function
- **Example**: $\boldsymbol{G} =< \boldsymbol{Z_{21}^*}, \times > = \phi(21) = \phi(3) \times \phi(7) = (3-1) \times (7-1) = 12$
- The elements are $\{1,2,4,5,8,10,11,13,16,17,19,20\}$
  - These are relatively prime with 21

Exponentiation and Logarithms

Order of an element

- $ord(a)$ is the smallest integer $i \mid a^i \equiv 1 \ (mod\ n)$
- **Example**: find the order of all elements in $G =< Z_{10}^{*}, \times >$

$G = \{1, 3, 7, 9\}$ *(use Lagrange theorem for faster computations)*

a) $1^1 \equiv 1 \ (mod\ 10) \rightarrow ord(1) = 1$

b) $3^1 \equiv 3 \ (mod\ 10); 3^2 \equiv 9 \ (mod\ 10); 3^4 \equiv 1 \ (mod\ 10) \rightarrow ord(3) = 4$

c) $7^1 \equiv 7 \ (mod\ 10); 7^2 \equiv 9 \ (mod\ 10); 7^4 \equiv 1 \ (mod\ 10) \rightarrow ord(7) = 4$

d) $9^1 \equiv 9 \ (mod\ 10); 9^2 \equiv 1 \ (mod\ 10) \rightarrow ord(9) = 2$

Exponentiation and Logarithms

Euler's Theorem

- If $a \in G =< Z_n^*, \times>$, then $a^{\phi(n)} \equiv 1 \, mod \, n$
- **Example**: for $G =< Z_8^*, \ \times> = \{1, 3, 5, 7\}$, we have $\phi(8) = 4$
  - For $i = \phi(n) = 4 \rightarrow x = 1$ for every a
  - The values of $x = 1$ for many values of i

	$i = 1$	$i = 2$	$i = 3$	$i = 4$	$i = 5$	$i = 6$	$i = 7$
$a = 1$	**x: 1**	x: 1	x: 1	x: 1	x: 1	x: 1	x: 1
$a = 3$	x: 3	**x: 1**	x: 3	x: 1	x: 3	x: 1	x: 3
$a = 5$	x: 5	**x: 1**	x: 5	x: 1	x: 5	x: 1	x: 5
$a = 7$	x: 7	**x: 1**	x: 7	x: 1	x: 7	x: 1	x: 7

Exponentiation and Logarithms

Euler's Theorem

- If $a \in G =<$
- **Example**: fo
  - For $i = \phi($
  - The values

Useful for

1. Computing large modular exponentiations
2. Computing modular inverses

$$a^{\phi(n)-1} \equiv a^{-1}\ (mod\ n)$$

= 7

$a = 1$	**x: 1**	x: 1	x: 1	x: 1	x: 1	x: 1	x: 1
$a = 3$	x: 3	**x: 1**	x: 3	x: 1	x: 3	x: 1	x: 3
$a = 5$	x: 5	**x: 1**	x: 5	x: 1	x: 5	x: 1	x: 5
$a = 7$	x: 7	**x: 1**	x: 7	x: 1	x: 7	x: 1	x: 7

Exponentiation and Logarithms

Primitive root

- In $G =< Z_n^*,\ \times>$, if $ord(a) = \phi(n)$ → a is called the <u>primitive root</u> of G.
- **Example**: $G =< Z_8^*,\ \times> = \{1, 3, 5, 7\},\ \phi(8) = 4$ has **no** primitive roots.
  - Recall that the order should be the smallest $i \mid a^i \equiv 1\ mod\ n$
  - The order of all elements are all smaller than 4

	$i = 1$	$i = 2$	$i = 3$	$i = 4$	$i = 5$	$i = 6$	$i = 7$
$a = 1$	**x: 1**	x: 1	x: 1	x: 1	x: 1	x: 1	x: 1
$a = 3$	x: 3	**x: 1**	x: 3	x: 1	x: 3	x: 1	x: 3
$a = 5$	x: 5	**x: 1**	x: 5	x: 1	x: 5	x: 1	x: 5
$a = 7$	x: 7	**x: 1**	x: 7	x: 1	x: 7	x: 1	x: 7

Exponentiation and Logarithms

Primitive root

- For $G = < Z_7^*,\ \times>$, $\phi(7) = 6$ has two primitive roots at $a = 3, 5$ because $ord(3) = 6$ and $ord(5) = 6$

		$i = 1$	$i = 2$	$i = 3$	$i = 4$	$i = 5$	$i = 6$
	$a = 1$	x: 1	x: 1	x: 1	x: 1	x: 1	x: 1
	$a = 2$	x: 2	x: 4	x: 1	x: 2	x: 4	x: 1
Primitive root →	$a = 3$	x: 3	x: 2	x: 6	x: 4	x: 5	x: 1
	$a = 4$	x: 4	x: 2	x: 1	x: 4	x: 2	x: 1
Primitive root →	$a = 5$	x: 5	x: 4	x: 6	x: 2	x: 3	x: 1
	$a = 6$	x: 6	x: 1	x: 6	x: 1	x: 6	x: 1

Exponentiation and Logarithms

Primitive root

- Rule: The group $G =< Z_n^*, \times>$ has primitive roots <u>only if</u> n =2, 4, p^t or $2p^t$ | p is an odd prime number (not 2) and t is an integer
- **Example**: For which value of n, does the $G =< Z_n^*, \times>$ have primitive roots: 17, 20, 38, and 50?

a) $G =< Z_{17}^*, \times>$ has primitive roots; 17 is prime and $t = 1$

b) $G =< Z_{20}^*, \times>$ has no primitive roots

c) $G =< Z_{38}^*, \times>$ has primitive roots; $38 = 2 \times 19$ and 19 is a prime

d) $G =< Z_{50}^*, \times>$ has primitive roots; $50 = 2 \times 5^2$ and 5 is a prime

Exponentiation and Logarithms

Primitive root

- Rule: If the group $G = < Z_n^*, \times >$ has primitive roots, the number of primitive roots = $\phi\big(\phi(n)\big)$
- Example: the number of primitive root in $G = < Z_{17}^*, \times >$ is $\phi\big(\phi(17)\big) = \phi(16) = 8$.

Exponentiation and Logarithms

Primitive root

- Three questions arise:

Given an element a and the group $G =< Z_n^*, \times>$, how to check whether a is a primitive root of G? **This is not an easy task**.

a. We need to find $\phi(n)$, which is as difficult as factorization of n.
b. We need to check whether $ord(a) = \phi(n)$.

Given a group $G =< Z_n^*, \times>$, how to **find all primitive roots** of G?
[-] This is **more difficult**; repeat part b for all elements of the group.

Given a group $G =< Z_n^*, \times>$, how can we **select** a primitive root of G given that the value of n is chosen by the user and **only the user knows** $\phi(n)$.
[-] The user tries several elements until he or she finds the first one.

Exponentiation and Logarithms

Discrete Logarithm
Finite multiplicative group
Order of the group
Order of an element
Euler's Theorem
Primitive roots
Cyclic group

Cyclic group

- If $G =< Z_n^*, \times >$ has primitive roots → it's <u>cyclic group</u>
- Each root works as a **generator** to generate all the elements in the set.
  - $Z_n^* = \{g^1, g^2, \ldots, g^{\phi(n)}\}$, where g is primitive root

- **Example**: $G =< Z_{10}^*, \times >$ has $\phi(\phi(10)) = 2$ primitive roots, which are 3, 7
  - Both 3 and 7 can be used to generate whole set:

$g = 3 \rightarrow$	$g^1 \bmod 10 = 3$	$g^2 \bmod 10 = 9$	$g^3 \bmod 10 = 7$	$g^4 \bmod 10 = 1$
$g = 7 \rightarrow$	$g^1 \bmod 10 = 7$	$g^2 \bmod 10 = 9$	$g^3 \bmod 10 = 3$	$g^4 \bmod 10 = 1$

Exponentiation and Logarithms

- The idea of discrete logarithm is to solve

$$y = g^x \ (mod\ n)$$

We know n, g, y and interested to compute x → $x = \log_g y$

- **Example**: Find x in each of the following cases:

a. $4 \equiv 3^x \ (mod\ 7)$ → $x = \log_3 4 \ (mod\ 7) = 4$

b. $6 \equiv 5^x \ (mod\ 7)$ → $x = \log_5 6 \ (mod\ 7) = 3$

- Check this calculator: https://www.alpertron.com.ar/DILOG.HTM

Exponentiation and Logarithms

- It is HARD to compute the discrete logarithms when n is very LARGE

The **discrete logarithm** problem has the same complexity as the **factorization problem**.

Content

Content
Exponentiation and Logarithms
Diffie-Hellman Key Exchange
Diffie-Hellman Protocols
ElGamal Cryptosystem

Diffie-Hellman Key Exchange

- DH allows two users to securely exchange secret values.
  - E.g., the secret keys of symmetric encryption algorithms.

- DH depends on the problem of **discrete logarithms** as a hard problem.
- For any integer b and a primitive root a of prime number p, we can find a unique exponent i such that

$$b \equiv a^i \ (mod\ p)$$

The exponent i is called the **discrete logarithm** → $dlog_{a,p}(b)$

Diffie-Hellman Key Exchange

Diffie-Hellman Key Exchange

Alice		Bob
Alice and Bob share a prime number q and an integer α, such that $\alpha < q$ and α is a primitive root of q	α and q are public values	Alice and Bob share a prime number q and an integer α, such that $\alpha < q$ and α is a primitive root of q

Diffie-Hellman Key Exchange

Alice	Bob
Alice and Bob share a prime number q and an integer α, such that $\alpha < q$ and α is a primitive root of q	Alice and Bob share a prime number q and an integer α, such that $\alpha < q$ and α is a primitive root of q
Alice generates a private key X_A such that $X_A < q$	Bob generates a private key X_B such that $X_B < q$

Diffie-Hellman Key Exchange

Alice

Alice and Bob share a prime number q and an integer α, such that $\alpha < q$ and α is a primitive root of q

Alice generates a private key X_A such that $X_A < q$

Alice calculates a public key $Y_A = \alpha^{X_A} \bmod q$

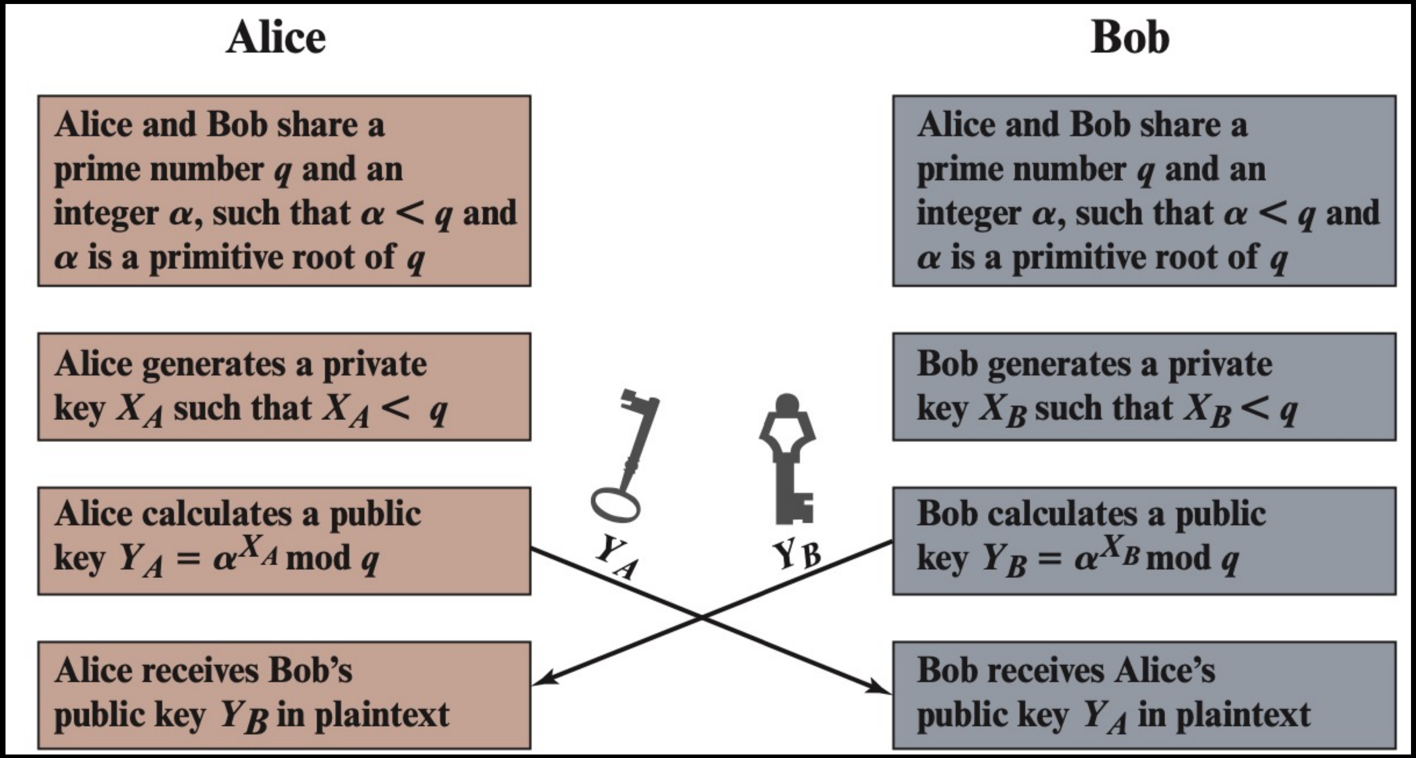
Bob

Alice and Bob share a prime number q and an integer α, such that $\alpha < q$ and α is a primitive root of q

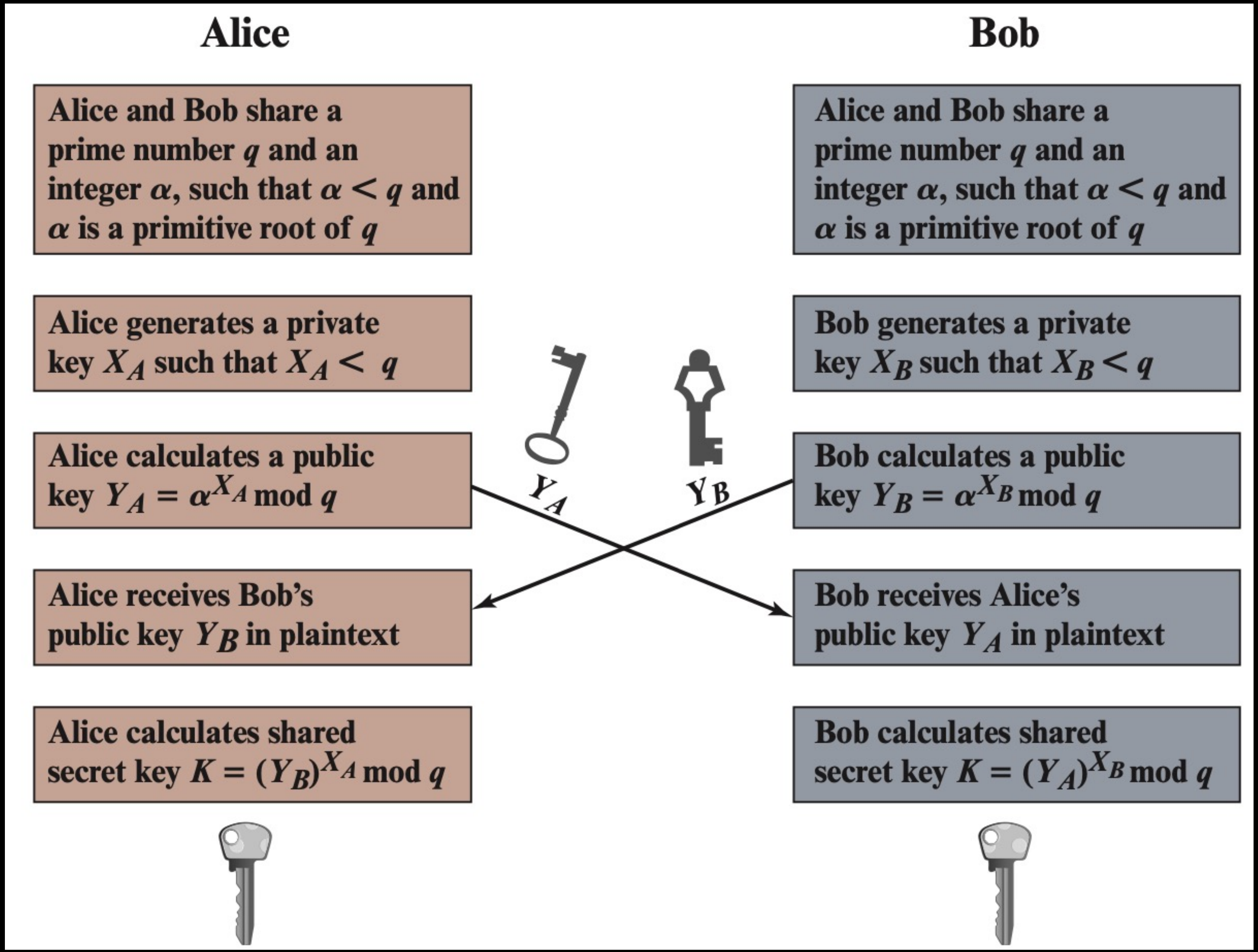
Bob generates a private key X_B such that $X_B < q$

Bob calculates a public key $Y_B = \alpha^{X_B} \bmod q$

Diffie-Hellman Key Exchange



Diffie-Hellman Key Exchange



Diffie-Hellman Key Exchange

- For the previous algorithm

Public	**Private**
Prime number q	Alice's PR X_A
The primitive root α	Bob's PR X_B
Alice's PU $Y_A = \alpha^{X_A}\ mod\ q$	The shared secret key $K = (Y_B)^{X_A}\ mod\ q = (Y_A)^{X_B}\ mod\ q$
Bob's PU $Y_B = \alpha^{X_B}\ mod\ q$	-

Diffie-Hellman Key Exchange

- The calculated shared secret key is the same at both sides because:

$$
\begin{aligned}
K &= (Y_B)^{X_A} \ mod\ q \\
&= (\alpha^{X_B}\ mod\ q)^{X_A}\ mod\ q \\
&= (\alpha^{X_B})^{X_A}\ mod\ q \\
&= \alpha^{X_B X_A}\ mod\ q \\
&= (\alpha^{X_A})^{X_B}\ mod\ q \\
&= (\alpha^{X_A}\ mod\ q)^{X_B}\ mod\ q \\
&= (Y_A)^{X_B}\ mod\ q
\end{aligned}
$$

Diffie-Hellman Key Exchange

- The calculated shared secret key is the same at both sides because:

$$
\begin{aligned}
K &= (Y_B)^{X_A} \ mod\ q \\
&= (\alpha^{X_B}\ mod\ q)^{X_A}\ mod\ q \\
&= (\alpha^{X_B})^{X_A}\ mod\ q \\
&= \alpha^{X_B X_A}\ mod\ q \\
&= (\alpha^{X_A})^{X_B}\ mod\ q \\
&= (\alpha^{X_A}\ mod\ q)^{X_B}\ mod\ q \\
&= (Y_A)^{X_B}\ mod\ q
\end{aligned}
$$

How Secure?!

Diffie-Hellman Key Exchange

How Secure?!

An adversary only sees α, q, Y_A, and Y_B

Diffie-Hellman Key Exchange

How Secure?!

An adversary only sees α, q, Y_A, and Y_B

Calculating K requires knowing X_A or X_B
→ $K = (Y_A)^{X_B} \bmod q$

Diffie-Hellman Key Exchange

How Secure?!

An adversary only sees α, q, Y_A, and Y_B

Calculating K requires knowing X_A or X_B
→ $K = (Y_A)^{X_B}\ mod\ q$

But X_A and X_B are private!

Diffie-Hellman Key Exchange

How Secure?!

An adversary only sees α, q, Y_A, and Y_B

Calculating K requires knowing X_A or X_B
→ $K = (Y_A)^{X_B} \; mod \; q$

But X_A and X_B are private!

To compute X_A or X_B from the public values, it requires computing the discrete logarithm: $X_B = dlog_{\alpha,q}(Y_B)$

Diffie-Hellman Key Exchange

How Secure?!

An adv

C

-

Security of DH: (assuming large primes)

1. Easy to calculate exponentials modulo a prime.
2. Very difficult to calculate discrete logarithms.

To compute X_A or X_B from the public values, it requires computing the discrete logarithm: $X_B = dlog_{\alpha,q}(Y_B)$

Diffie-Hellman Key Exchange

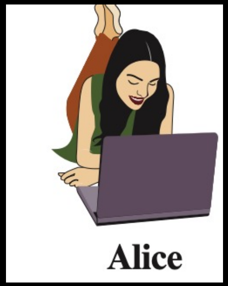
- Example:

Select q and α
Alice and Bob select their PRs: X_A, X_B
Alice computes her PU: $Y_A = \alpha^{X_A}\ mod\ q$
Bob computes his PU: $Y_B = \alpha^{X_B}\ mod\ q$
Alice and Bob compute $K = (Y_B)^{X_A}\ mod\ q,\ \mathrm{K} = (Y_A)^{X_B}\ mod\ q$

Diffie-Hellman Key Exchange

- Example:

Select q and α
Alice and Bob select their PRs: X_A, X_B
Alice computes her PU: $Y_A = \alpha^{X_A}\ mod\ q$
Bob computes his PU: $Y_B = \alpha^{X_B}\ mod\ q$
Alice and Bob compute $K = (Y_B)^{X_A}\ mod\ q,\ \mathrm{K} = (Y_A)^{X_B}\ mod\ q$

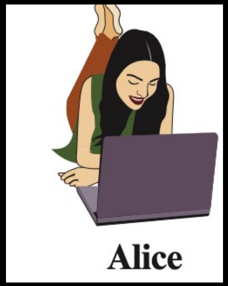


$$q = 353$$
$$\alpha = 3$$

Diffie-Hellman Key Exchange

- Example:

Select q and α
Alice and Bob select their PRs: X_A, X_B
Alice computes her PU: $Y_A = \alpha^{X_A} \ mod\ q$
Bob computes his PU: $Y_B = \alpha^{X_B} \ mod\ q$
Alice and Bob compute $K = (Y_B)^{X_A} \ mod\ q,\ \ K = (Y_A)^{X_B} \ mod\ q$



$$X_A = 97$$

$$q = 353$$
$$\alpha = 3$$

$$X_B = 233$$

Diffie-Hellman Key Exchange

- Example:

Select q and α
Alice and Bob select their PRs: X_A, X_B
Alice computes her PU: $Y_A = \alpha^{X_A} \mod q$
Bob computes his PU: $Y_B = \alpha^{X_B} \mod q$
Alice and Bob compute $K = (Y_B)^{X_A} \mod q,\ \mathrm{K} = (Y_A)^{X_B} \mod q$

$\boxed{X_A = 97}$

$\boxed{Y_A = 3^{97} mod\ 353 = 40}$

$q = 353$
$\alpha = 3$

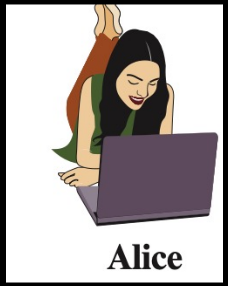
$\boxed{X_B = 233}$

$\boxed{Y_B = 3^{233} mod\ 353 = 248}$

Diffie-Hellman Key Exchange

- Example:

Select q and α
Alice and Bob select their PRs: X_A, X_B
Alice computes her PU: $Y_A = \alpha^{X_A} \ mod \ q$
Bob computes his PU: $Y_B = \alpha^{X_B} \ mod \ q$
Alice and Bob compute $K = (Y_B)^{X_A} \ mod \ q, \ \mathrm{K} = (Y_A)^{X_B} \ mod \ q$



$$q = 353$$
$$\alpha = 3$$

$$\boxed{X_A = 97}$$

$$\boxed{X_B = 233}$$

$$\boxed{Y_A = 3^{97} mod \ 353 = 40}$$

$$\boxed{Y_B = 3^{233} mod \ 353 = 248}$$

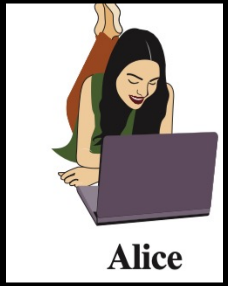
$$\boxed{K = 248^{97} mod \ 353 = 160}$$

$$\boxed{K = 40^{233} mod \ 353 = 160}$$

Diffie-Hellman Key Exchange

- Example:

Select q and α
Alice and Bob select their PRs: X_A, X_B
Alice computes her PU: $Y_A = \alpha^{X_A} \ mod \ q$
Bob computes his PU: $Y_B = \alpha^{X_B} \ mod \ q$
Alice and Bob compute $K = (Y_B)^{X_A} \ mod \ q, \ \mathrm{K} = (Y_A)^{X_B} \ mod \ q$



$$q = 353$$
$$\alpha = 3$$

$\boxed{X_A = 97}$

$\boxed{X_B = 233}$

$\boxed{Y_A = 3^{97} mod \ 353 = 40}$

$\boxed{Y_B = 3^{233} mod \ 353 = 248}$

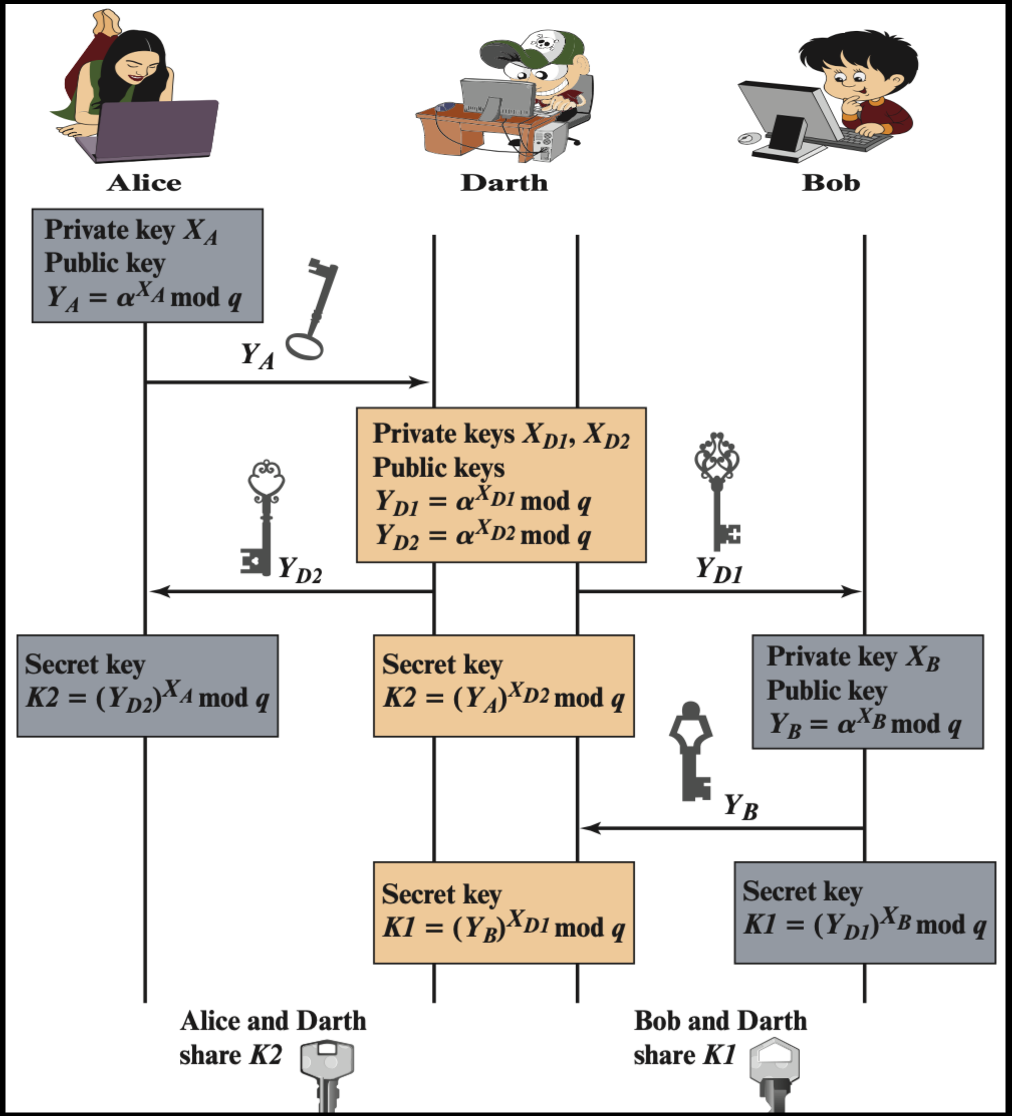
$\boxed{K = 248^{97} mod \ 353 = 160}$

$\boxed{K = 40^{233} mod \ 353 = 160}$

$\boxed{q = 353, \alpha = 3, Y_A = 40, Y_B = 248}$

Diffie-Hellman Key Exchange

- The DH protocol is vulnerable to **Man-in-the-Middle-Attack**.
  1. Darth intercepts the key-exchange process between Alice and Bob
  2. Darth share key values with Alice, and different key values with Bob
  3. Darth receives messages from Alice, he re-encrypts them with Bob's key and re-send them to Bob
- The attack works because no authentication on participants



Content

Content
Exponentiation and Logarithms
Diffie-Hellman Key Exchange
Diffie-Hellman Protocols
ElGamal Cryptosystem

Diffie-Hellman Protocols

Authenticated Diffie–Hellman

- Developed to address MitM attacks that can affect anonymous DH.

- Use a signature scheme; allowing Alice and Bob to sign their messages to stop Darth from sending messages on their behalf.

- The signatures are computed using another scheme, e.g., RSA-PSS

Diffie-Hellman Protocols

Authenticated Diffie–Hellman

$priv_A, priv_B, pub_A, pub_B$ are private and public key values for signatures

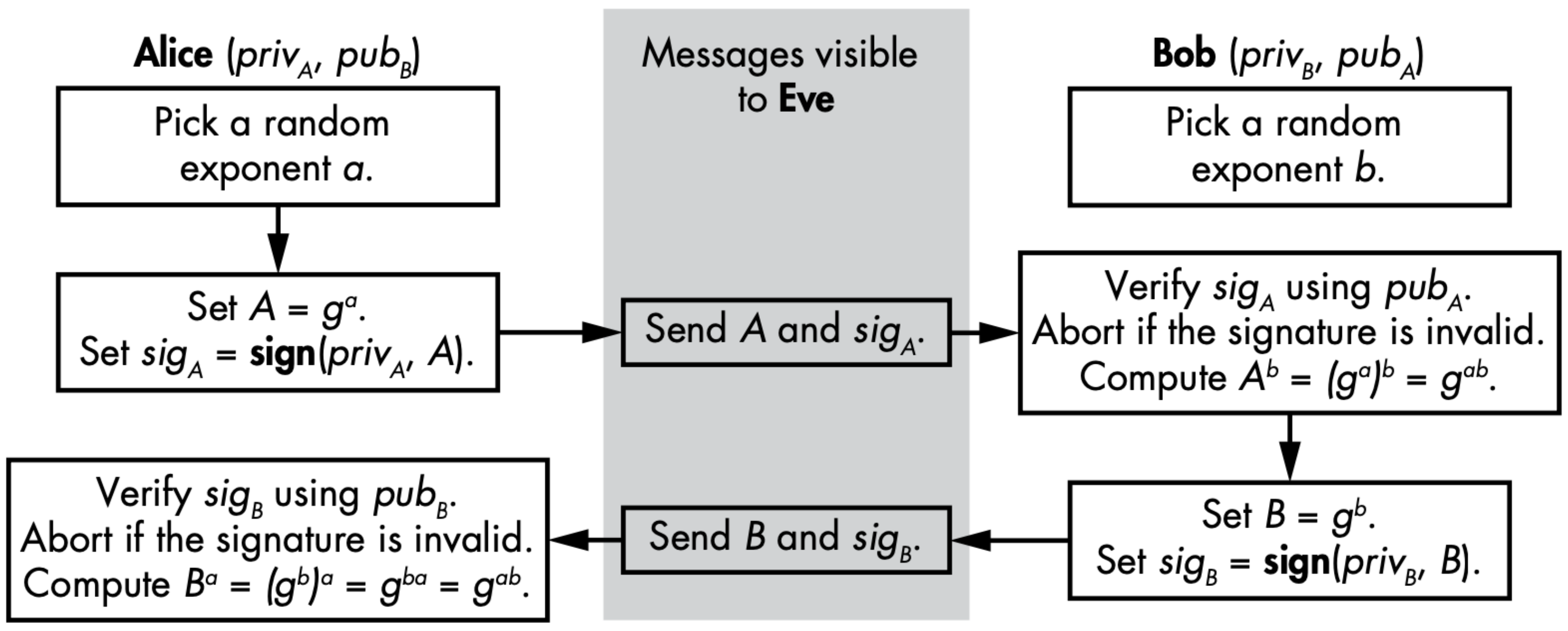


Figure 11-4: The authenticated Diffie–Hellman protocol

Diffie-Hellman Protocols

Menezes–Qu–Vanstone (MQV)

- More secure and better performance than authenticated DH.

- Allows users to send only two messages, independently of each other, in arbitrary order.

- No need to use a signature scheme

Diffie-Hellman Protocols

Menezes–Qu–Vanstone (MQV)

- x is Alice's PrK, y is Bob's PrK, $Y = g^y$ is Bob's PuK, and $X = g^x$ is Alice's PuK

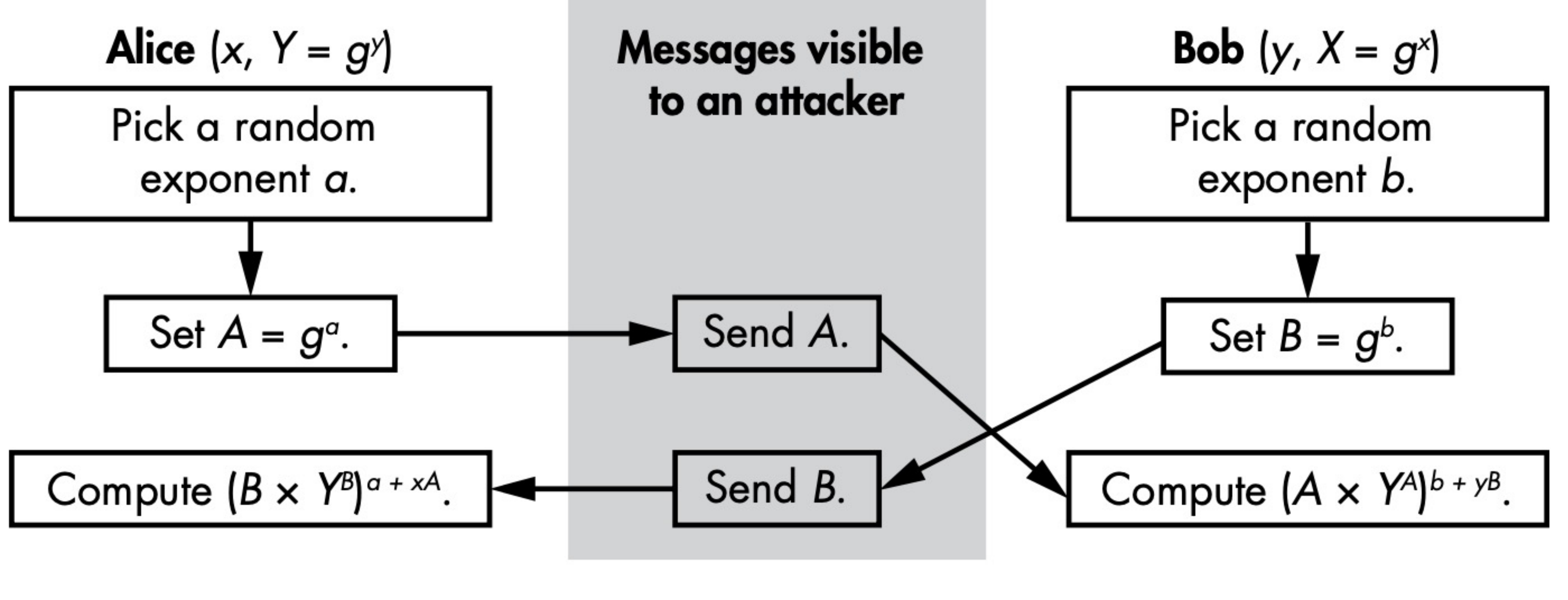


Figure 11-6: The MQV protocol

Content

Content
Exponentiation and Logarithms
Diffie-Hellman Key Exchange
Diffie-Hellman Protocols
ElGamal Cryptosystem

ElGamal Cryptosystem

- ElGamal is a public key cryptosystem that is used in:
  - Digital Signature Standard (DSS)
  - S/MIME email standard

- It relies on the discrete logarithm as Diffie-Hellman protocol
  - DH protocol is a key-exchange protocol
  - ElGamal is a cryptosystem

ElGamal Cryptosystem

- ElGamal public elements

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

- Key generation

Key Generation by Alice	
Select private X_A	$X_A < q - 1$
Calculate Y_A	$Y_A = \alpha^{X_A} \bmod q$
Public key	$\{q, \alpha, Y_A\}$
Private key	X_A

ElGamal Cryptosystem

- Encryption

Encryption by Bob with Alice's Public Key

Plaintext:	$M < q$
Select random integer k	$k < q$
Calculate K	$K = (Y_A)^k \bmod q$
Calculate C_1	$C_1 = \alpha^k \bmod q$
Calculate C_2	$C_2 = KM \bmod q$
Ciphertext:	(C_1, C_2)

ElGamal Cryptosystem

- Decryption

Decryption by Alice with Alice's Private Key	
Ciphertext:	(C_1, C_2)
Calculate K	$K = (C_1)^{X_A} \bmod q$
Plaintext:	$M = (C_2 K^{-1}) \bmod q$

ElGamal Cryptosystem

- K serves as a one-time key used to encrypt and decrypt the message
- How K is recovered during the decryption process

$K = (Y_A)^k \ mod \ q$ (K is defined during the encryption process)
$K = (\alpha^{X_A} \ mod \ q)^k \ mod \ q$
$K = \left(\alpha^{kX_A}\right) mod \ q$
$K = C_1^{X_A} \ mod \ q$ (substituting $C_1 = \alpha^k \ mod \ q$)

- Using K, we recover the plaintext as

$C_2 = KM \ mod \ q$
$M = C_2K^{-1} \ mod \ q$
$M = KMK^{-1} \ mod \ q = M \ mod \ q$

ElGamal Cryptosystem

- Example: Alice generates keys

Select a prime number q and a primitive root
Select a private $X_A \mid \mathrm{X_A} < \mathrm{q} - 1$
Calculate $Y_A = \alpha^{X_A} \: mod \: q$
Publish public key $\{q, \alpha, Y_A\}$ Keep the private key X_A

$q = 19$
The primitive roots of 19 are {2, 3, 10, 13, 14, 15}
Select $\alpha = 10$

ElGamal Cryptosystem

- Example: Alice generates keys

Select a prime number q and a primitive root
Select a private X_A | $\mathrm{X_A} < \mathrm{q} - 1$
Calculate $Y_A = \alpha^{X_A} \; mod \; q$
Publish public key $\{q, \alpha, Y_A\}$ Keep the private key X_A

$$q = 19, \;\; \alpha = 10$$

$$X_A = 5$$

ElGamal Cryptosystem

- Example: Alice generates keys

Select a prime number q and a primitive root
Select a private X_A | $\mathrm{X_A} < \mathrm{q} - 1$
Calculate $Y_A = \alpha^{X_A} \; mod \; q$
Publish public key $\{q, \alpha, Y_A\}$ Keep the private key X_A

$$q = 19, \;\; \alpha = 10$$

$$X_A = 5$$

$$Y_A = 10^5 \; mod \; 19 = 3$$

ElGamal Cryptosystem

- Example: Alice generates keys

Select a prime number q and a primitive root
Select a private $X_A \mid \mathrm{X_A} < \mathrm{q} - 1$
Calculate $Y_A = \alpha^{X_A} \mod q$
Publish public key $\{q, \alpha, Y_A\}$ Keep the private key X_A

$q = 19, \ \alpha = 10$

$X_A = 5$

$Y_A = 10^5 \mod 19 = 3$

Publish $\{19, 10, 3\}$
Keep 5

ElGamal Cryptosystem

- Encrypt the message $M = 17$:

Public key: $\{19, 10, 3\}$

Select random $k \mid k < q$
Calculate $K = (Y_A)^k \ mod \ q$
Calculate $C_1 = \alpha^k \ mod \ q$
Calcuate $C_2 = KM \ mod \ q$
Send the ciphertext (C_1, C_2)

ElGamal Cryptosystem

- Encrypt the message $M = 17$:

Public key: $\{19, 10, 3\}$

Select random $k \mid k < q$
Calculate $K = (Y_A)^k \ mod \ q$
Calculate $C_1 = \alpha^k \ mod \ q$
Calcuate $C_2 = KM \ mod \ q$
Send the ciphertext (C_1, C_2)

Select $k = 6$

ElGamal Cryptosystem

- Encrypt the message $M = 17$:

Public key: $\{19, 10, 3\}$

Select random $k \mid k < q$
Calculate $K = (Y_A)^k \ mod \ q$
Calculate $C_1 = \alpha^k \ mod \ q$
Calcuate $C_2 = KM \ mod \ q$
Send the ciphertext (C_1, C_2)

Select $k = 6$

$$K = 3^6 mod\ 19 = 7$$

ElGamal Cryptosystem

- Encrypt the message $M = 17$:

Public key: $\{19, 10, 3\}$

Select random $k \mid k < q$
Calculate $K = (Y_A)^k \ mod\ q$
Calculate $C_1 = \alpha^k \ mod\ q$
Calcuate $C_2 = KM \ mod\ q$
Send the ciphertext (C_1, C_2)

Select $k = 6$

$K = 3^6 mod\ 19 = 7$

$C_1 = 10^6 mod\ 19 = 11$

ElGamal Cryptosystem

- Encrypt the message $M = 17$:

$\boxed{\text{Public key: } \{19, 10, 3\}}$

Select random $k \mid k < q$
Calculate $K = (Y_A)^k \bmod q$
Calculate $C_1 = \alpha^k \bmod q$
Calcuate $C_2 = KM \bmod q$
Send the ciphertext (C_1, C_2)

$\boxed{\text{Select } k = 6}$

$\boxed{K = 3^6 mod\ 19 = 7}$

$\boxed{C_1 = 10^6 mod\ 19 = 11}$

$\boxed{C_2 = 7 \times 17\ mod\ 19 = 5}$

ElGamal Cryptosystem

- Encrypt the message $M = 17$:

Public key: $\{19, 10, 3\}$

Select random $k \mid k < q$
Calculate $K = (Y_A)^k \ mod\ q$
Calculate $C_1 = \alpha^k \ mod\ q$
Calcuate $C_2 = KM \ mod\ q$
Send the ciphertext (C_1, C_2)

Select $k = 6$

$K = 3^6 mod\ 19 = 7$

$C_1 = 10^6 mod\ 19 = 11$

$C_2 = 7 \times 17\ mod\ 19 = 5$

$(11, 5)$

ElGamal Cryptosystem

- Decrypt $(11, 5)$:

Calculate $K = C_1^{X_A} \ mod \ q$
Compute $K^{-1} = egcd(K, q)$
Calculate $M = C_2 K^{-1}$

Private key: $X_A = 5$

ElGamal Cryptosystem

- Decrypt $(11, 5)$:

Calculate $K = C_1^{X_A} \bmod q$
Compute $K^{-1} = egcd(K, q)$
Calculate $M = C_2 K^{-1} \bmod q$

Private key: $X_A = 5$

$$K = 11^5 \bmod 19 = 7$$

ElGamal Cryptosystem

- Decrypt $(11, 5)$:

Calculate $K = C_1^{X_A} \; mod \; q$
Compute $K^{-1} = egcd(K, q)$
Calculate $M = C_2 K^{-1} \; mod \; q$

Private key: $X_A = 5$

$K = 11^5 \; mod \; 19 = 7$

$K^{-1} = 11$

ElGamal Cryptosystem

- Decrypt $(11, 5)$:

Calculate $K = C_1^{X_A}\ mod\ q$
Compute $K^{-1} = egcd(K, q)$
Calculate $M = C_2 K^{-1}\ mod\ q$

Private key: $X_A = 5$

$K = 11^5\ mod\ 19 = 7$

$K^{-1} = 11$

$M = 11 \times 5\ mod\ q = 17$

ElGamal Cryptosystem

- If a message is broken up into blocks, use a unique k for each block.
- If k is used for more than one block and a block, M_1, is known, other blocks are decrypted as follows:
  - Let

$$C_{1,1} = \alpha^k mod\ q; \;\; C_{2,1} = KM_1\ mod\ q$$
$$C_{1,2} = \alpha^k mod\ q; \;\; C_{2,2} = KM_2\ mod\ q$$

  - Then,

$$\frac{C_{2,1}}{C_{2,2}} = \frac{KM_1\ mod\ q}{KM_2\ mod\ q} = \frac{M_1\ mod\ q}{M_2\ mod\ q}$$

  - Since M_1 is known, then $M_2 = \left(C_{2,1}\right)^{-1}\left(C_{2,2}\right)M_1\ mod\ q$

ElGamal Cryptosystem

- Implement the ElGamal cryptosystem in Python

Task

- Implement the DH protocol using a client-server architecture
- What are the common attacks on the ElGamal cryptosystem?