

CYB650

Introduction to Cryptography

Cryptography Techniques



Prepared by Dr. Esraa Saleh Alomari
Lecture #4



جامعة الجميع الذكية
EVERYONE'S SMART UNIVERSITY

The Part 2 of the module 2 covers the Cryptography Techniques are as follows:

- ❑ **Symmetric Techniques**

- **Stream Cipher**
- **Block Cipher**

- ❑ **Asymmetric Techniques**

- **Diffie–Hellman key exchange protocol.**
- **DSS (Digital Signature Standard), which incorporates the Digital Signature Algorithm.**
- **ElGamal.**
- **Elliptic-curve cryptography**



- ❑ Symmetric-key encryption provides secrecy when two parties, say Alice and Bob, communicate. An adversary who intercepts a message should not get any significant information about its content.

- ❑ To set up a secure communication channel, Alice and Bob first agree on a key k . They keep their shared key k secret. Before sending a message m to Bob, Alice encrypts m by using the encryption algorithm E and the key k . She obtains the ciphertext $c = E(k, m)$ and sends c to Bob. By using the decryption algorithm D and the same key k , Bob decrypts c to recover the plaintext $m = D(k, c)$.

- ❑ A typical stream cipher encrypts plaintext one byte at a time, although a stream cipher may be designed to operate on one bit at a time or on units larger than a byte at a time. Figure 7.7 is a representative diagram of stream cipher structure. In this structure, a key is input to a pseudorandom bit generator that produces a stream of 8-bit numbers that are apparently



The output of the generator, called a keystream, is combined one byte at a time with the plaintext stream using the bitwise exclusive-OR (XOR) operation. For example, if the next byte generated by the generator is 01101100 and the next plaintext byte is 11001100,

then the resulting ciphertext byte is The stream cipher is similar to the one-time pad in the previous lecture. The difference is that a one-time pad uses a genuine random number stream, whereas a stream cipher uses a pseudorandom number stream.

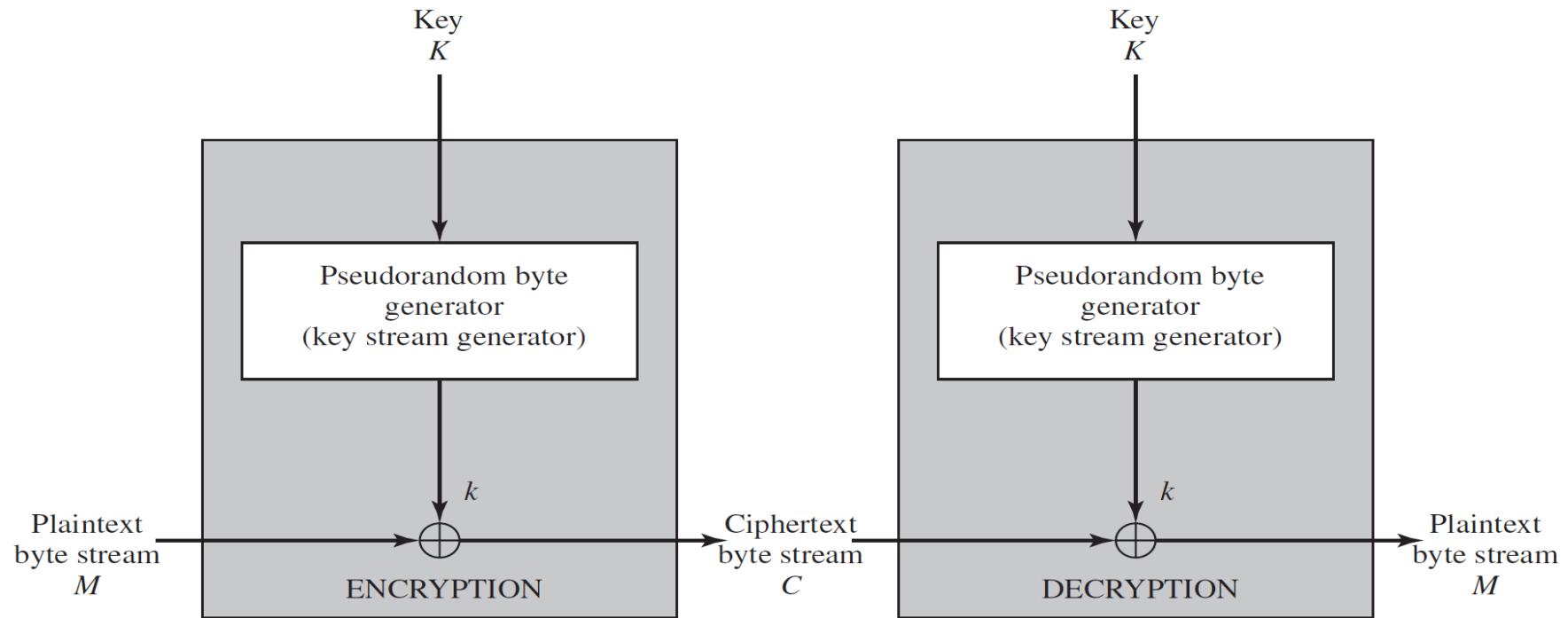
Stream Cipher

$$\begin{array}{rcl} 11001100 & \text{plaintext} & \\ \oplus \quad \underline{01101100} & \text{key stream} & \\ 10100000 & \text{ciphertext} & \end{array}$$

Decryption requires the use of the same pseudorandom sequence:

$$\begin{array}{rcl} 10100000 & \text{ciphertext} & \\ \oplus \quad \underline{01101100} & \text{key stream} & \\ 11001100 & \text{plaintext} & \end{array}$$

Stream Cipher



Stream Cipher Diagram

Stream Cipher

USING EXCLUSIVE OR (XOR) IN CRYPTOGRAPHY

**XOR
LOGIC**

$0 \text{ XOR } 0 = 0$ Same Bits
 $1 \text{ XOR } 1 = 0$ Same Bits
 $1 \text{ XOR } 0 = 1$ Different Bits
 $0 \text{ XOR } 1 = 1$ Different Bits

XOR Symbol
 $\oplus$

ENCRYPT

$\oplus$

$$\begin{array}{r}
 00110101 \text{ Plaintext} \\
 11100011 \text{ Secret Key} \\
 \hline
 = 11010110 \text{ Ciphertext}
 \end{array}$$

DECRYPT

$\oplus$

$$\begin{array}{r}
 11010110 \text{ Ciphertext} \\
 11100011 \text{ Secret Key} \\
 \hline
 = 00110101 \text{ Plaintext}
 \end{array}$$

XOR Logic

A block cipher is a method of encrypting data in blocks to produce ciphertext using a cryptographic key and algorithm. The block cipher processes fixed-size blocks simultaneously, as opposed to a stream cipher, which encrypts data one bit at a time.

A block cipher uses a symmetric key and algorithm to encrypt and decrypt a block of data. A block cipher requires an initialization vector (IV) that is added to the input plaintext in order to increase the key space of the cipher and make it more difficult to use brute force to break the key.

The IV is derived from a random number generator, which is combined with text in the first block and the key to ensure all subsequent blocks result in ciphertext that does not match that of the first encryption block.

The block size of a block cipher refers to the number of bits that are processed together. Data Encryption Standard (DES) and Advanced Encryption Standard (AES) are both symmetric block ciphers.

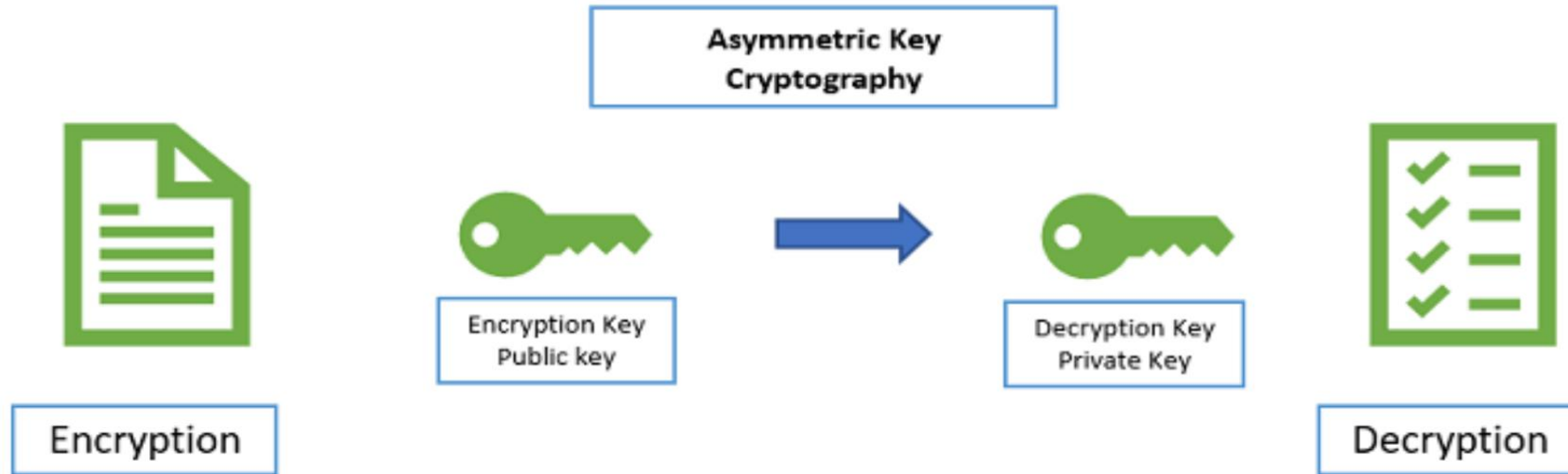
The DES block cipher was originally designed by IBM in 1975 and consisted of 64-bit blocks and a 56-bit key. This cipher is not considered secure anymore, due to the short key size, and was replaced in 1998 by AES. AES uses a 128-bit block size and a 128-, 192- or 256-bit key size.

Asymmetric cryptography is a branch of cryptography where a secret key can be divided into two parts, a public key and a private key. The public key can be given to anyone, trusted or not, while the private key must be kept secret (just like the key in symmetric cryptography).

Asymmetric cryptography has two primary use cases: authentication and confidentiality. Using asymmetric cryptography, messages can be signed with a private key, and then anyone with the public key is able to verify that the message was created by someone possessing the corresponding private key. This can be combined with a proof of identity system to know what entity (person or group) actually owns that private key, providing authentication.

Asymmetric Cipher

Encryption with asymmetric cryptography works in a slightly different way from symmetric encryption. Someone with the public key is able to encrypt a message, providing confidentiality, and then only the person in possession of the private key is able to decrypt it.



Asymmetric key cryptography diagram

At the heart of Asymmetric Encryption lies a cryptographic algorithm. This algorithm uses a key generation protocol (a kind of mathematical function) to generate a key pair. Both the keys are mathematically connected with each other. This relationship between the keys differs from one algorithm to another. The algorithm is basically a combination of two functions – encryption function and decryption function. To state the obvious, the encryption function encrypts the data and decryption function decrypts it.

Difference between Symmetric and Asymmetric Encryption

Symmetric Encryption

Symmetric encryption consists of one of key for encryption and decryption.

Symmetric Encryption is a lot quicker compared to the Asymmetric method.

RC4

AES

DES

3DES

QUAD

Asymmetric Encryption

Asymmetric Encryption consists of two cryptographic keys known as Public Key and Private Key.

As Asymmetric Encryption incorporates two separate keys, the process is slowed down considerably.

RSA

Diffie-Hellman

ECC

El Gamal

DSA

When you visit any HTTPS website/webpage, your browser establishes Asymmetrically encrypted connection with that website. Your browser automatically derives the public key of the SSL/TLS certificate installed on the website (that's why it's called 'Public Key'). This is how it'll look like:

Asymmetric Cipher

30 82 01 0a 02 82 01 01 00 c2 d8 be ec a4 e1 52 20 7f 7f 7d
1a 17 38 99 17 ef 6a 9e af 66 89 67 5a 58 e2 b8 7c 76 f2 b8
c6 8f 98 e4 06 eb 3c 1c 04 34 1e 10 a9 42 c2 34 be 99 3b 98
7b 35 60 3a d5 41 bb 96 19 1a 3c 66 a0 75 77 64 2a 2e 19 42
5a b1 d0 1f 4d ac 32 2e af 4e 20 b8 89 07 83 51 21 e4 35 02
4b 10 45 03 37 ce 26 87 e0 b8 4d dc ba c5 e7 ae 60 68 b3 0c
a3 5c 4f dd 30 1f 95 96 a5 2e e5 6f ae e8 e2 dc df 3a ab 51 74
82 f5 9e 15 3a ab 7c 99 3c 07 5b ad f2 88 a2 23 1c cd 41 d8
66 a4 90 0d 4a 23 05 5c de aa e3 82 13 f4 08 87 b3 34 08 6f
38 fb f8 84 ec 06 99 e0 ab 8a ab 1b 7c 99 fd 57 94 67 17 15
b7 27 67 c1 bc d1 a7 f6 c6 7e 01 63 02 0c 03 c4 bb 1f 70 0d
db 27 ab 79 57 d9 92 35 f3 92 3c ad f4 fb f0 36 82 33 5a a0 f9
82 78 04 a6 e7 d6 ee 01 23 68 36 68 3b 41 fe 68 56 0b 6b 36
3b 83 b1 02 03 01 00 01

Diffie-Hellman Algorithm

The purpose of the algorithm is to enable two users to exchange a key securely that can then be used for subsequent encryption of messages.

The Diffie-Hellman algorithm depends for its effectiveness on the difficulty of computing discrete logarithms. First, we define a primitive root of a prime number p as one whose power generate all the integers from 1 to $(p-1)$ i.e., if „ a “ is a primitive root of a prime number p , then the numbers $a \bmod p$, $a^2 \bmod p$, ... $a^{p-1} \bmod p$ are distinct and consists of integers from 1 to $(p-1)$ in some permutation. For any integer „ b “ and a primitive root „ a “ of a prime number „ p “, we can find a unique exponent „ i “ such that

$$b \equiv a^i \bmod p, \text{ where } 0 \leq i \leq (p-1)$$

Diffie-Hellman Algorithm

The exponent „i” is referred to as discrete logarithm. With this background, we can define Diffie-Hellman key exchange as follows:

There are publicly known numbers: a prime number „q” and an integer α that is primitive root of q. suppose users A and B wish to exchange a key. User A selects a random integer $X_A < q$ and computes $Y_A = \alpha^{X_A} \bmod q$. Similarly, user B independently selects a random integer $X_B < q$ and computes $Y_B = \alpha^{X_B} \bmod q$. Each side keeps the X value private and makes the Y value available publicly to the other side. User A computes the key as $K = (Y_B)^{X_A} \bmod q$ and User B computes the key as $K = (Y_A)^{X_B} \bmod q$



Diffie-Hellman Algorithm

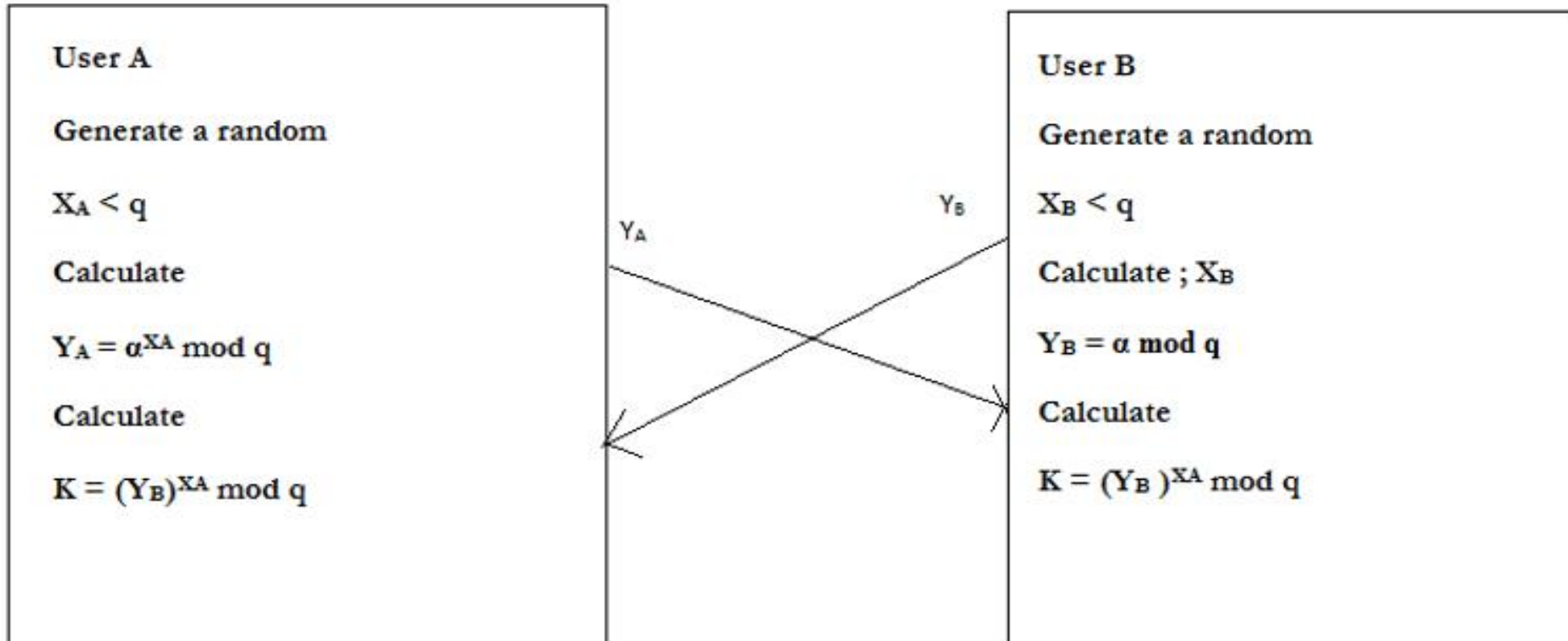
These two calculations produce identical results.

$$\begin{aligned} K &= (YB)XA \bmod q \\ &= (\alpha XB \bmod q)XA \bmod q \\ &= (\alpha XB)XA \bmod q \\ &= (\alpha XA)XB \bmod q \\ &= (\alpha XA \bmod q)XB \bmod q \\ &= (YA)XB \bmod q \end{aligned}$$

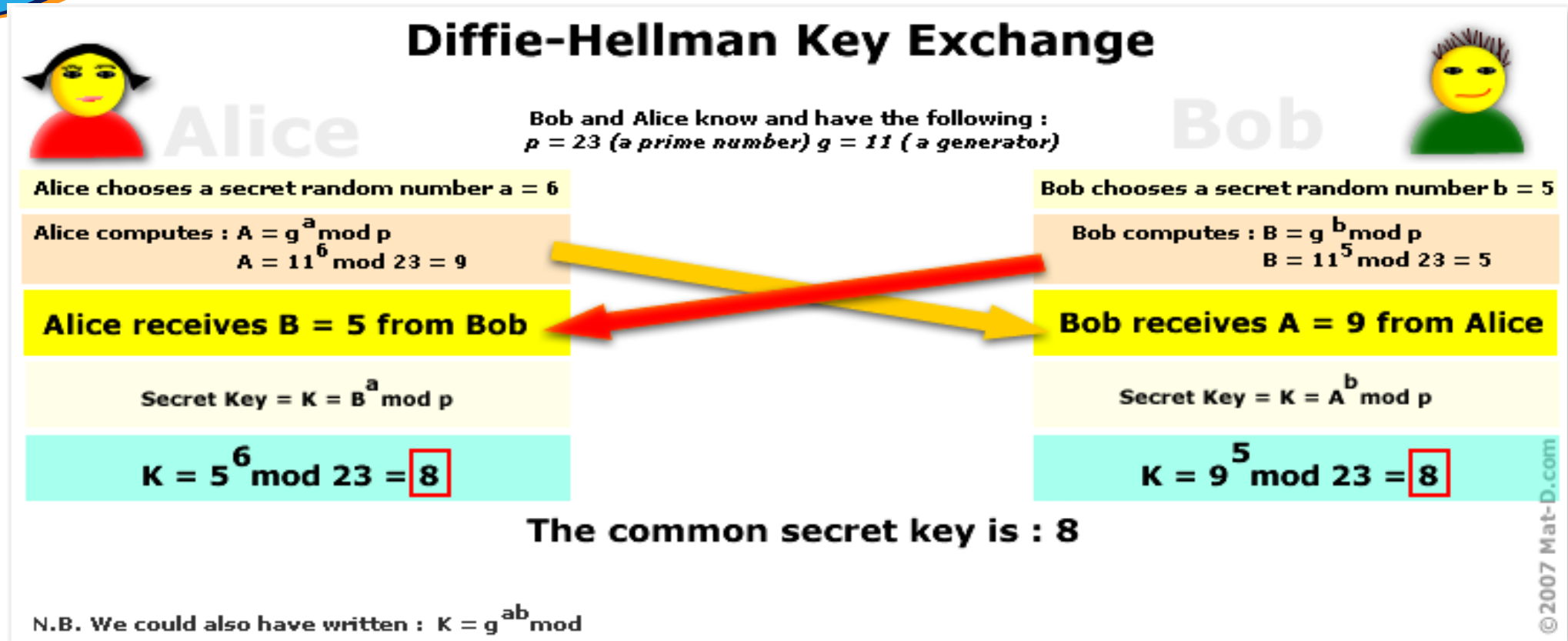
Diffie-Hellman Algorithm

The result is that two sides have exchanged a secret key. The security of the algorithm lies in the fact that, while it is relatively easy to calculate exponentials modulo a prime, it is very difficult to calculate discrete logarithms. For large primes, the latter task is considered infeasible.

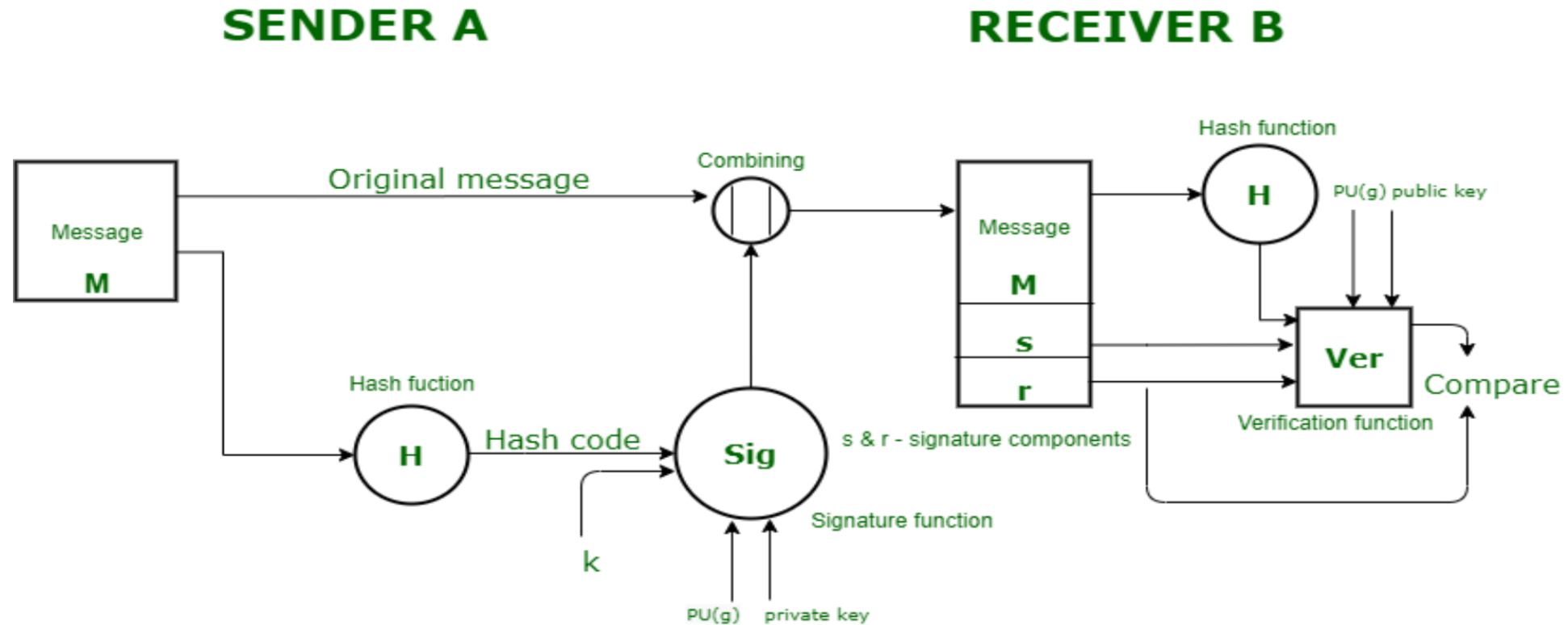
Diffie-Hellman Algorithm



Diffie-Hellman Algorithm



Digital Signature Standard (DSS) is a Federal Information Processing Standard (FIPS) which defines algorithms that are used to generate digital signatures with the help of Secure Hash Algorithm (SHA) for the authentication of electronic documents. DSS only provides us with the digital signature function and not with any encryption or key exchanging strategies.



Sender Side :

In DSS Approach, a hash code is generated out of the message and following inputs are given to the signature function

1. The hash code.
2. The random number 'k' generated for that particular signature.
3. The private key of the sender i.e., $PR(a)$.
4. A global public key(which is a set of parameters for the communicating principles) i.e., $PU(g)$.



These input to the function will provide us with the output signature containing two components – ‘s’ and ‘r’. Therefore, the original message concatenated with the signature is sent to the receiver.



Receiver Side :

At the receiver end, verification of the sender is done. The hash code of the sent message is generated. There is a verification function which takes the following inputs:

1. The hash code generated by the receiver.
2. Signature components 's' and 'r'.
3. Public key of the sender.
4. Global public key.



The output of the verification function is compared with the signature component 'r'. Both the values will match if the sent signature is valid because only the sender with the help of its private key can generate a valid signature.

ElGamal encryption is a public-key cryptosystem. It uses asymmetric key encryption for communicating between two parties and encrypting the message.

This cryptosystem is based on the difficulty of finding discrete logarithm in a cyclic group that is even if we know g_a and g_k , it is extremely difficult to compute g_{ak} .

Idea of ElGamal cryptosystem

Suppose Alice wants to communicate with Bob

1. Bob generates public and private keys:

- Bob chooses a very large number q and a cyclic group F_q .
- From the cyclic group F_q , he choose any element g and an element a such that $\gcd(a, q) = 1$.
- Then he computes $h = g^a$.
- Bob publishes F , $h = g^a$, q , and g as his public key and retains a as private key.



2. Alice encrypts data using Bob's public key :

- Alice selects an element k from cyclic group F such that $\gcd(k, q) = 1$.
- Then she computes $p = g^k$ and $s = hk = g^ak$.
- She multiplies s with M .
- Then she sends $(p, M*s) = (g^k, M*s)$.

Bob decrypts the message :

Bob calculates $s' = p^a = g^ak$.

He divides $M*s$ by s' to obtain M as $s = s'$.



In this cryptosystem, the original message M is masked by multiplying g^a to it. To remove the mask, a clue is given in form of g^k . Unless someone knows a , he will not be able to retrieve M . This is because finding discrete log in a cyclic group is difficult and simplifying knowing g^a and g^k is not good enough to compute g^a .



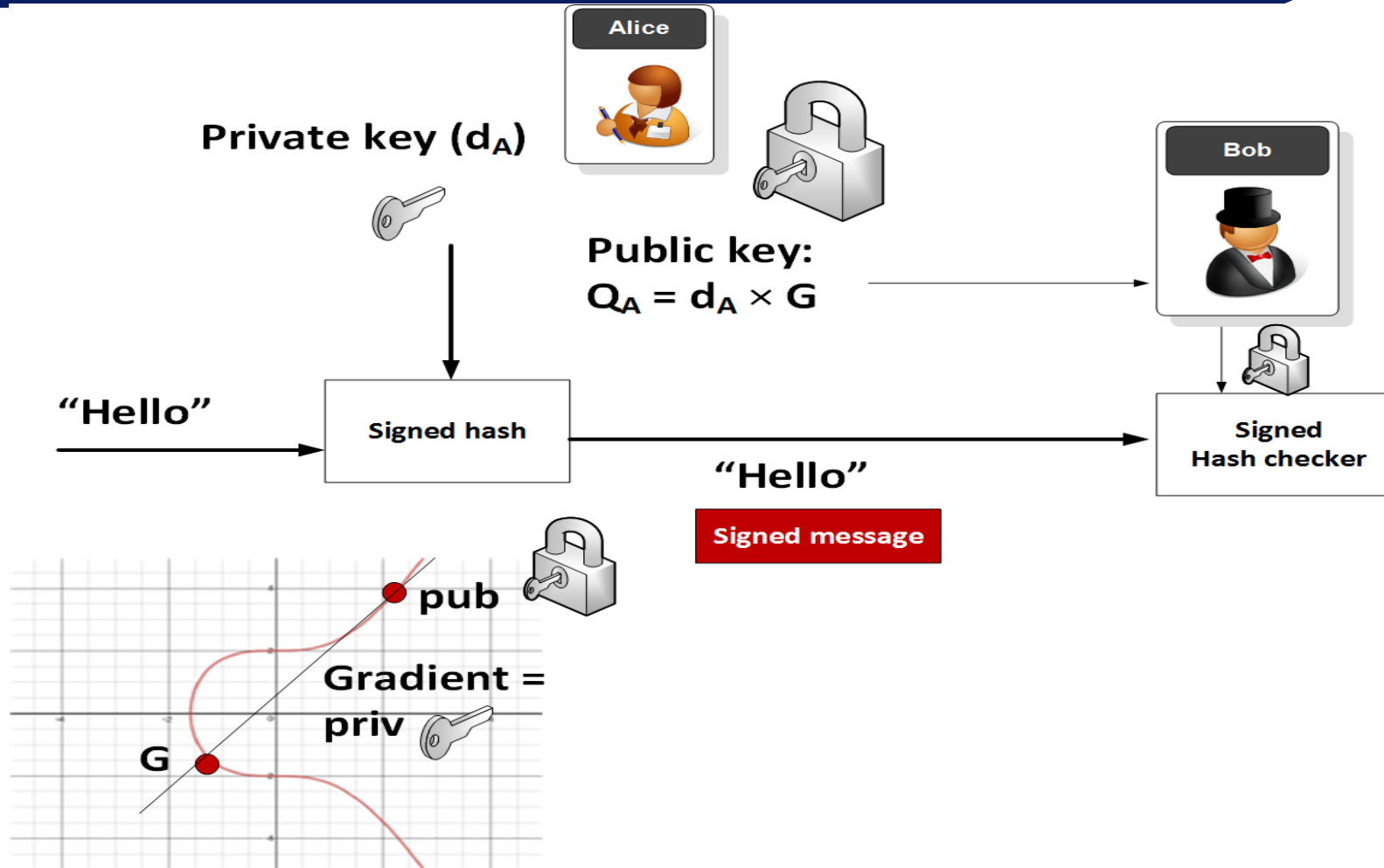
The addition operation in ECC is the counterpart of modular multiplication in RSA, and multiple addition is the counterpart of modular exponentiation. To form a cryptographic system using elliptic curves, we need to find a "hard problem" corresponding to factoring the product of two primes or taking the discrete logarithm.

Consider the equation $Q = kP$ where $Q, P \in E_p(a, b)$ and $k < p$. It is relatively easy to calculate Q given k and P , but it is relatively hard to determine k given Q and P . This is called the discrete logarithm problem for elliptic curves.

Because $9P = (4, 5) = Q$, the discrete logarithm $Q = (4, 5)$ to the base $P = (16, 5)$ is $k = 9$. In a real application, k would be so large as to make the brute-force approach infeasible.



Elliptic-curve cryptography



Thank You

ANY
QUESTIONS?

