Asymmetrical Cryptosystems

Asst. Prof. Dr. Serap ŞAHİN

Topics

1. Deffie Hellman (DH) Key Exchange

Asymmetrical Cryptosystems

2. RSA
3. ELGAMAL

Terms

Cryptographic Algorithms

Encryption	Decryption
$E(P) = C$	$D(C) = P$
$E_k(P) = C$	$D_k(C) = P$

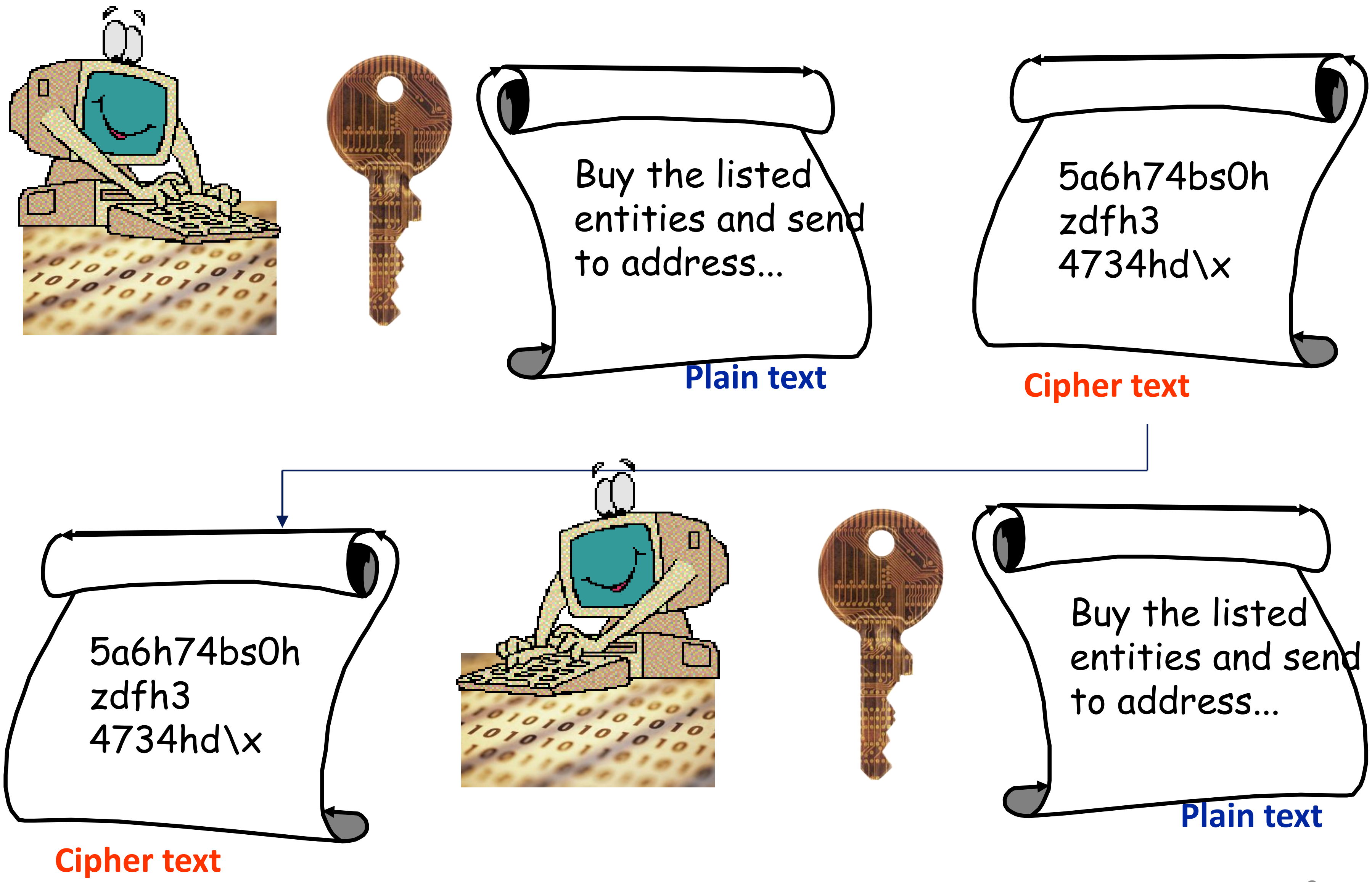
Key(k)

$E_k = D_k \Rightarrow$ **Symmetrical**

$E_k \neq D_k \Rightarrow$ **Asymmetrical**

Ref: *Slides of Assoc. Prof. Dr. Ahmet Koltuksuz – slides from 2 to 8.*

Symmetrical=same key for E and D


Buy the listed entities and send to address...
Plain text
5a6h74bs0h zdfh3 4734hd\x
Cipher text
5a6h74bs0h zdfh3 4734hd\x
Cipher text
Buy the listed entities and send to address...
Plain text

Asymmetrical Cryptosystems

In 1976 invented by Diffie, Hellman and Merkle.

Main idea is using of two different keys for encryption and decryption (public and private keys).

There is no possibility to produce private key from known public key for any attack.

Asymmetrical Cryptosystems

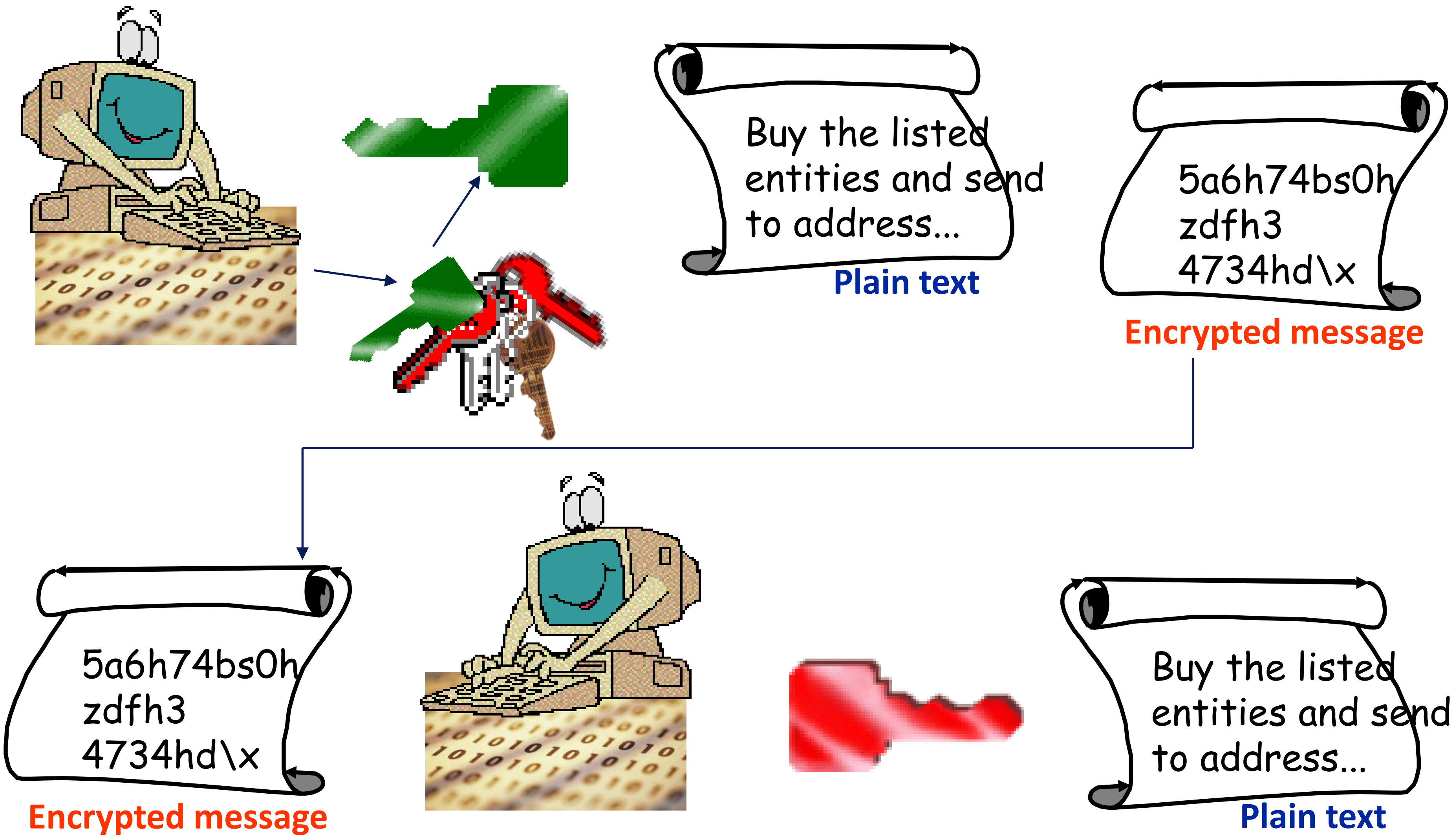
The receivers public key is known by every one.

Any sender can use the receiver's public key to encrypt own message before to send it.

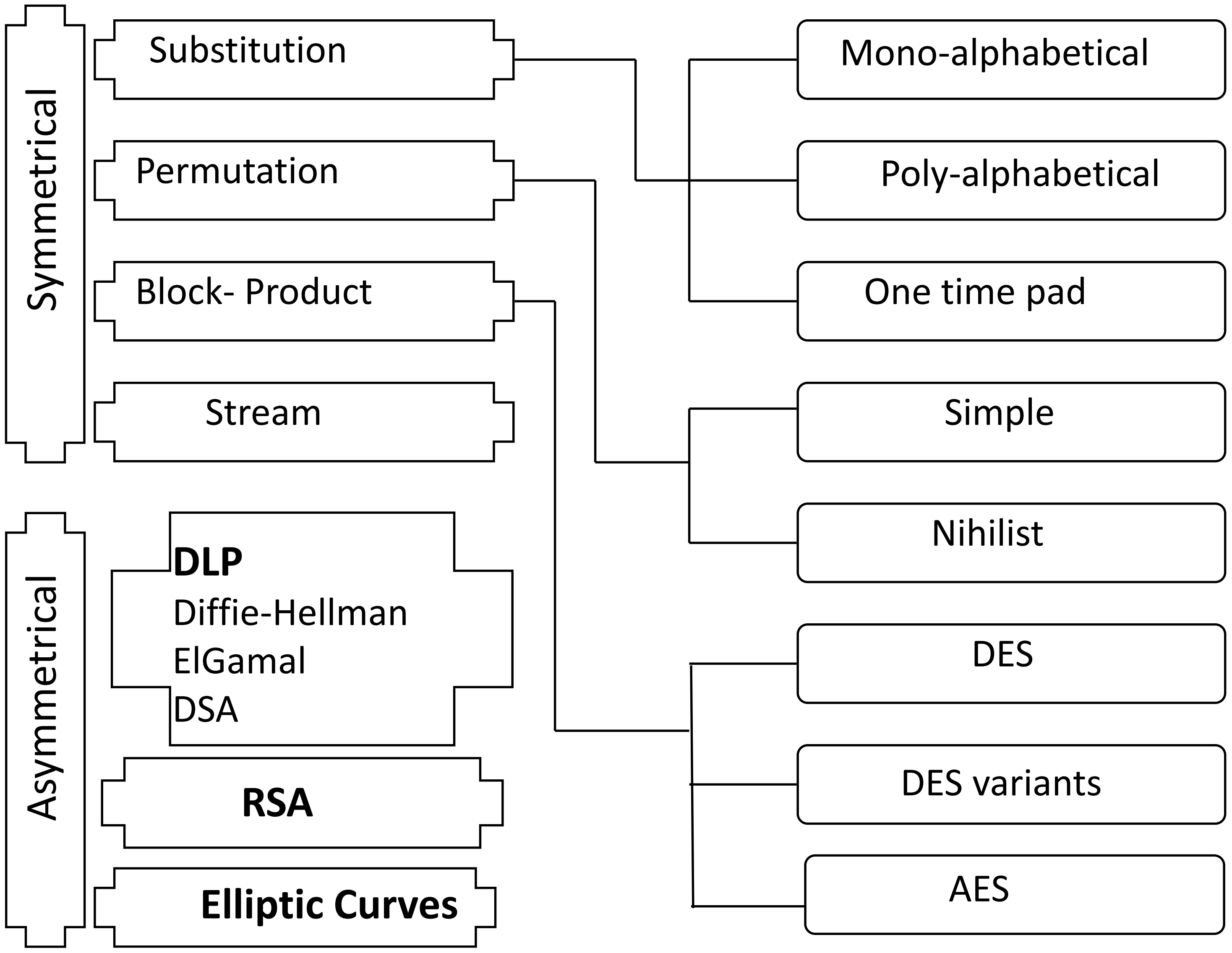
Private key is only known by the receiver and receiver use it to decrypt the received cipher text.

So, everyone should have (public key, private key) pair. Public key is announced to every one. But the secrecy of private key is crucial.

Asymmetrical= two keys


Buy the listed entities and send to address...
Plain text
5a6h74bs0h zdfh3 4734hd\x
Encrypted message
5a6h74bs0h zdfh3 4734hd\x
Encrypted message
Buy the listed entities and send to address...
Plain text

Terms


Symmetrical
Substitution
Permutation
Block- Product
Stream
Asymmetrical
DLP
Diffie-Hellman
ElGamal
DSA
RSA
Elliptic Curves
Mono-alphabetical
Poly-alphabetical
One time pad
Simple
Nihilist
DES
DES variants
AES

Terms

Discrete logarithm problem

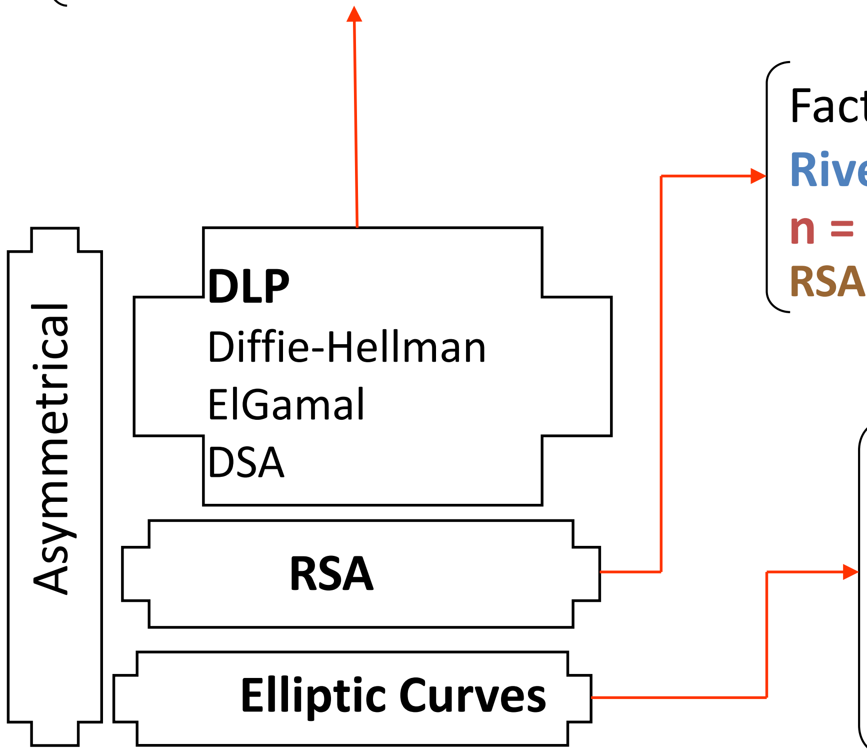
Diffie-Hellman, 1976

$y = g^x \pmod{p}$

Diffie-Hellman key exchange

ElGamal cryptosystem

Digital Signature Algoritm (DSA)



Factorization

Rivest-Shamir-Adleman, 1978

$n = pq$

RSA, PGP

Elliptic Curves

EC Discrete logarithm problem

Miller-Koblitz, 1985-87

$y^2 = x^3+ax+b \pmod{p}$

ECC, ECDSA

Asymmetrical Cryptosystems

Security of

Rivest-Shamir-Adleman (RSA) Cryptosystem

Asst. Prof. Dr. Serap ŞAHİN

References: *Complexity and Cryptography, An Introduction, J.Talbot, D.Welsh, Cambridge*
Stinson_Cryptography_Theory_and_Practice, ch04
Introduction to Cryptography and Security Mechanisms 2005, Lecture Notes of Dr Keith Martin, keith.martin@rhul.ac.uk

Rivest

Shamir

Adleman

Euler

Fermat

RSA Cryptosystem

- The idea of a public-key system was due to Diffie and Hellman in 1976.
- The first realization of a public-key system came in 1977 by Rivest, Shamir, and Adleman, who invented the well-known **RSA Cryptosystem**.
- The security of **RSA** is based on the difficulty of factoring large integers.

RSA

- Step 1: $N = p \times q$, p and q very large prime
- Step 2: $T = \Phi(N) = (p-1)(q-1)$
- Step 3: Choose e s.t. $\gcd(e, T) = 1$
- Step 4: Find $de \equiv 1 \pmod{T}$
- Public key: (N, e)
- Private key: (N, d)

RSA

- Example 1
  - Let $p = 7, q = 11, e = 13$
  - Give the public and private keys in RSA cryptosystem

RSA

- Step 1: $N = 7 \times 11 = 77$
- Step 2: $T = 6 \times 10 = 60$
- Step 3: $\gcd(13, 60) = 1$, the choice is ok
- Step 4:

$$\begin{aligned} 60 &= 4 \times 13 + 8 \\ 13 &= 1 \times 8 + 5 \\ 8 &= 1 \times 5 + 3 \\ 5 &= 1 \times 3 + 2 \\ 3 &= 1 \times 2 + 1 \end{aligned} \quad \Longleftrightarrow \quad d = 37$$

RSA

- Public key: $(77, 13)$
- Private key: $(77, 37)$
- Example 2: Encrypt 5

$$C = 5^{13} \bmod 77 = 26$$

- Example 3: Decrypt

$$M = 26^{37} \bmod 77$$

RSA

- Example 3

$$37 = 32 + 4 + 1$$

$$\begin{aligned}
26^2 \bmod 77 &= 60 \\
26^4 \bmod 77 &= 58 \quad \Longrightarrow \quad 26^5 \bmod 77 = 45 \\
26^8 \bmod 77 &= 53 \qquad\qquad\qquad \Downarrow \\
26^{16} \bmod 77 &= 37 \\
26^{32} \bmod 77 &= 60 \quad \Longrightarrow \quad 26^{37} \bmod 77 = 5
\end{aligned}$$

One way functions

- They play a central role in cryptography; they are important for constructing public-key cryptosystems.

- Example; Suppose ***n*** is the product of two large primes ***p*** and ***q***, and let ***b*** be a positive integer. Then define $f : Z_n \to Z_n$ to be $f(x) = x^b \bmod n$

- For a suitable choice of *b* and *n*, this is in fact the **RSA** encryption function.

Trapdoor One way functions

- If we construct a public-key cryptosystem, We do not want $\mathbf{e_K}$ to be a one-way function from Bob's (receiver's) point of view, since he wants to be able to decrypt messages that he receives in an efficient way.
- It is necessary that Bob possesses a trapdoor, which consists of secret information that permits easy inversion of $\mathbf{e_K}$.
  - That is, Bob can decrypt efficiently because he has some extra secret knowledge about K.
- So, we say that a function is a trapdoor one-way function if it is a one-way function, but it becomes easy to invert with the knowledge of a certain trapdoor.

The RSA trapdoor 1-to-1 function

- Parameters: $\begin{cases} N=pq. \quad N \approx 1024 \text{ bits.} \quad p,q \approx 512 \text{ bits.} \\ e - \text{encryption exponent.} \quad \gcd(e, \varphi(N)) = 1 . \end{cases}$

- 1-to-1 function: $\mathbf{RSA(M) = M^e}$ (mod N) where $M \in Z_N^*$

- Trapdoor: **d** – decryption exponent.

  Where $e \cdot \mathbf{d} = 1 \pmod{\varphi(N)}$

- Inversion: $\mathbf{RSA(M)^d} = M^{ed} = M^{k\varphi(N)+1} = \mathbf{M} \pmod N$

- (n,e,t,ε)-RSA Assumption: For any t-time alg. A:

$$\Pr\left[A(N,e,x) = x^{1/e} \ (N) \ : \ \begin{matrix} p,q \xleftarrow{R} \text{n-bit primes,} \\ N \leftarrow pq, \quad x \xleftarrow{R} Z_N^* \end{matrix} \right] < \varepsilon$$

Security of RSA Cryptosystem

- The security of **RSA** is based on the hope that the encryption function $e_K(x) = x^b \bmod n$ is one-way, so it will be computationally infeasible for an opponent to decrypt a ciphertext.
- The trapdoor that allows Bob to decrypt is the knowledge of the factorization $n = pq$.
- Since Bob knows this factorization, he can compute $\phi(n) = (p - 1)(q - 1)$ and then compute the decryption exponent a using the **Extended Euclidean algorithm**.

Security of RSA Cryptosystem

- One obvious attack on the cryptosystem is for a cryptanalyst to attempt to factor n.
- If this can be done, it is a simple manner to compute $\phi(n)=(p - 1)(q - 1)$ and then compute the decryption exponent a from b exactly as Bob did.
- if the **RSA Cryptosystem** is to be secure, it is certainly necessary that $n = pq$ must be large enough that factoring it will be computationally infeasible.
  – Current factoring algorithms are able to factor numbers having up to **130 decimal digits.**

Security of RSA

- Possible security holes:
  - Need to use “safe” primes p and q. In particular p-1 and q-1 should have large prime factors …
  - p and q should not have the same number of digits. Can use a middle attack starting at sqrt(n).
  - e (encryption key) cannot be too small
  - Don’t use same n for different e’s.

Factoring in the Real World

- **Quadratic Sieve (QS):**

$$T(n) = e^{(1+o(n))(\ln n)^{1/2}(\ln(\ln n))^{1/2}}$$

  - Used in 1994 to factor a 129 digit (428-bit) number. 1600 Machines, 8 months.

- **Number field Sieve (NFS):**

$$T(n) = e^{(1.923+o(1))(\ln n)^{1/3}(\ln(\ln n))^{2/3}}$$

  - Used in 1999 to factor 155 digit (512-bit) number. 35 CPU years. At least 4x faster than QS

RSA in the "Real World"

- **Part of many standards**: PKCS, ITU X.509, ANSI X9.31, IEEE P1363
- **Used by**: SSL, PEM, PGP, Entrust, …

- The standards specify many details on the implementation, e.g.
  - e should be selected to be small, but not too small
  - "multi prime" versions make use of n = pqr… this makes it cheaper to decode especially in parallel (uses Chinese remainder theorem).

RSA security summary

There are two one-way functions involved in the security of RSA.

One-way function	Description
Encryption function	The encryption function is a trapdoor one-way function, whose trapdoor is the private key. The difficulty of reversing this function without the trapdoor knowledge is **believed** (but not known) to be as difficult as factoring.
Multiplication of two primes	The difficulty of determining an RSA private key from an RSA public key is **known** to be equivalent to factoring n. An attacker thus cannot use knowledge of an RSA public key to determine an RSA private key unless they can factor n. Because multiplication of two primes is believed to be a one-way function, determining an RSA private key from an RSA public key is believed to be very difficult.

Rivest-Shamir-Adleman (RSA) Cryptosystem

- RSA seems to be a secure public-key cryptosystem, despite our best efforts.
- Still no proof that it is theoretically safe.
- Most attacks on RSA come from poor configuration or bad implementations.

Asymmetrical Cryptosystems

ElGamal, Diffie-Hellman Key Exchange

Asst. Prof. Dr. Serap ŞAHİN

Ref: Introduction to Cryptography and Security Mechanisms, Lecture Notes of Dr Keith Martin, keith.martin@rhul.ac.uk

Section 3#

ElGamal

Public key blueprint

- The keys used to encrypt and decrypt are different.
- Anyone who wants to be a receiver needs to "publish" an encryption key, which is known as the public key.
- Anyone who wants to be a receiver needs a unique decryption key, which is known as the private key.
- It should not be possible to deduce the plaintext from knowledge of the ciphertext and the public key.
- Some guarantee needs to be offered of the authenticity of a public key.

Important question

Do public key cipher systems solve all the problems of symmetric key cipher systems?

One way functions

A **one-way function** is a function that is “easy” to compute and “difficult” to reverse.

How might we express this notion of a one way function informally in complexity theoretic terms?

OWF: Modular exponentiation

The process of **exponentiation** just means raising numbers to a power.

Raising **a** to the power **b**, normally denoted a^b just means multiplying **a** by itself **b** times. In other words:

$$a^b = a \times a \times a \times \ldots \times a$$

Modular exponentiation means computing a^b modulo some other number **n**. We tend to write this as

$$a^b \bmod n.$$

Modular exponentiation is “easy”.

OWF: Modular exponentiation

However, given **a**, **x**, and **x**=**a^b mod n** (when **n** is prime), calculating **b** is regarded by mathematicians as a hard problem. (**b=log**$_a$x)

This difficult problem is often referred to as the **discrete logarithm problem**.

In other words, given a number **a** and a prime number **n**, the function

$$f(b) = a^b \bmod n$$

is believed to be a one-way function.

Suitable OWFs

We have seen that the encryption process of a public key cipher system requires a one way function.

Is every one way function suitable for implementation as the encryption process of a public key cipher system?

2. ElGamal

ElGamal

We will also take a look at the ElGamal public key cipher system for a number of reasons:

- To show that RSA is not the only public key system
- To exhibit a public key system based on a different one way function
- ElGamal is the basis for several well-known cryptographic primitives

Setting up ElGamal

- Let **p** be a large prime
  - By “large” we mean here a prime rather typical in length to that of an RSA modulus
- Select a special number **g**
  - The number g must be a **primitive element** modulo **p**.
- Choose a private key **x**
  - This can be any number bigger than 1 and smaller than **p**-1
- Compute public key **y** from **x**, **p** and **g**
  - The public key **y** is **g** raised to the power of the private key **x** modulo **p**. In other words:

$$y = g^x \bmod p$$

Setting up ElGamal: example

Step 1: Let p = **23**

Step 2: Select a primitive element g = **11**

Step 3: Choose a private key x = **6**

Step 4: Compute y = $\mathbf{11^{6}}$ (mod **23**)
= **9**

Public key is **9**
Private key is **6**

ElGamal encryption

The first job is to represent the plaintext as a series of numbers modulo p. Then:

1. Generate a random number k
2. Compute two values C_1 and C_2, where

   $\mathbf{C_1 = g^k \bmod p}$ and $\mathbf{C_2 = My^k \bmod p}$

3. Send the ciphertext C, which consists of the two separate values C_1 and C_2.

ElGamal encryption: example

To encrypt M = 10 using Public key **9**

1 - Generate a random number k = **3**

2 - Compute $C_1 = \mathbf{11}^3 \bmod \mathbf{23} = \mathbf{20}$

$C_2 = 10 \times \mathbf{9}^{\mathbf{3}} \bmod \mathbf{23}$

$= 10 \times 16 = 160 \bmod \mathbf{23} = \mathbf{22}$

3 - Ciphertext C = (**20** , **22**)

ElGamal decryption

$$\boxed{\mathbf{C_1 = g^k \bmod p \qquad C_2 = My^k \bmod p}}$$

1 - The receiver begins by using their private key **x** to transform $\mathbf{C_1}$ into something more useful:

$$\mathbf{C_1^x = (g^k)^x \bmod p}$$

NOTE: $\mathbf{C_1^x = (g^k)^x = (g^x)^k = (y)^k = y^k \bmod p}$

2 - This is a very useful quantity because if you divide $\mathbf{C_2}$ by it you get **M**. In other words:

$$\mathbf{C_2 / y^k = (My^k) / y^k = M \bmod p}$$

ElGamal decryption: example

To decrypt C = (20 , 22)

1 - Compute $20^6 = 16 \bmod 23$

2 - Compute $22 / 16 = 22.16^{-1} = 10 \bmod 23$

3 - Plaintext = 10

ElGamal v RSA

PROS of ElGamal

- Does not rely on factorisation being hard

CONS of ElGamal

- Requires a random number generator
- Message expansion

While regarded as similar from a security perspective, are there any differences between ElGamal and RSA from an efficiency perspective?

Public key systems in practice

- Public key cipher systems led to mini revolution in cryptography in the mid 1970’s, with a further boom in interest since the development of the Internet in the 1990’s.
- Public key cipher systems are only likely to grow in importance in the coming years.

Summary

- Public key systems replace the problem of distributing symmetric keys with one of authenticating public keys
- Public key encryption algorithms need to be trapdoor one-way functions
- RSA is a public key encryption algorithm whose security is believed to be based on the problem of factoring large numbers
- ElGamal is a public key encryption algorithm whose security is believed to be based on the discrete logarithm problem
- RSA is generally favoured over ElGamal for practical rather than security reasons
- RSA and ElGamal are less efficient and fast to operate than most symmetric encryption algorithms because they involve modular exponentiation
- DH key exchange is an important protocol on which many real key exchange protocols are based

Diffie-Hellman

Diffie-Hellman

The **Diffie–Hellman (DH) key exchange** technique was first defined in their seminal paper in 1976.

DH key exchange is a method of exchanging public (i.e. non-secret) information to obtain a shared secret.

DH is not an encryption algorithm.

DH key exchange has the following important properties:

1. The resulting shared secret cannot be computed by either of the parties without the cooperation of the other.
2. A third party observing all the messages transmitted during DH key exchange cannot deduce the resulting shared secret at the end of the protocol.

Principle behind DH

DH key exchange was first proposed before there were any known public key algorithms, but the idea behind it motivated the hunt for practical public key algorithms.

DH key exchange is not only a useful and practical key establishment technique, but also a significant milestone in the history of modern cryptography.

DH key exchange assumes first that there exists:

1. A public key cipher system that has a special property (we come to this shortly).

2. A carefully chosen, publicly known function F that takes two numbers x and y as input, and outputs a third number F(x,y) (for example, multiplication is such a function).

Principle behind DH

Assume that Alice and Bob are the parties who wish to establish a shared secret, and let their public and private keys in the public key cipher system be denoted by (PA , SA) and (PB , SB) respectively.

The basic principle behind Diffie–Hellman key exchange is as follows:

1. Alice and Bob exchange their public keys PA and PB.
2. Alice computes F(SA , PB)
3. Bob computes F(SB, PA)
4. The special property of the public key cipher system, and the choice of the function F, are such that F(SA , PB) = F(SB, PA). If this is the case then Alice and Bob now share a secret.
5. This shared secret can easily be converted by some public means into a bitstring suitable for use as, for example, a DES key.

Diffie-Hellman key exchange

The most commonly described implementation of DH key exchange uses the keys of the ElGamal cipher system and a very simple function F.

The system parameters (which are public) are:

- **a large prime number p – typically 1024 bits in length**
- **a primitive element g**

1. Alice generates a private random value **a**, calculates $\mathbf{g}^{\mathbf{a}}$ (mod **p**) and sends it to Bob. Meanwhile Bob generates a private random value **b**, calculates $\mathbf{g}^{\mathbf{b}}$ (mod **p**) and sends it to Alice.
2. Alice takes $\mathbf{g}^{\mathbf{b}}$ and her private random value **a** to compute
   $(\mathbf{g}^{\mathbf{b}})^{\mathbf{a}} = \mathbf{g}^{\mathbf{ab}}$ (mod **p**).
3. Bob takes $\mathbf{g}^{\mathbf{a}}$ and his private random value **b** to compute
   $(\mathbf{g}^{\mathbf{a}})^{\mathbf{b}} = \mathbf{g}^{\mathbf{ab}}$ (mod **p**).
4. Alice and Bob adopt $\mathbf{g}^{\mathbf{ab}}$ (mod **p**) as the shared secret.

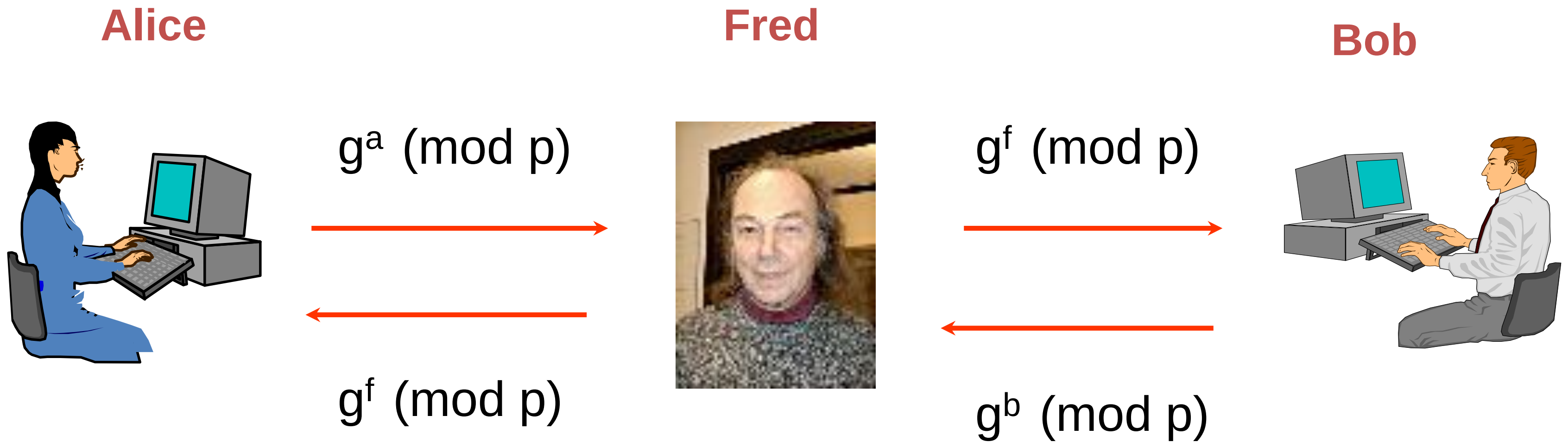
DH questions

1. What is the hard problem on which the DH key exchange algorithm is based?
2. The example of DH key exchange that we described is based on ElGamal keys. Can you use the public and private keys of any established public key encryption algorithm to implement DH key exchange?

$$b = a^x \bmod p \longrightarrow \textbf{Easy}$$

$$x = \log_a b \bmod \mathrm{p} \longrightarrow \textbf{Hard problem !}$$

Man-in-the-middle attack



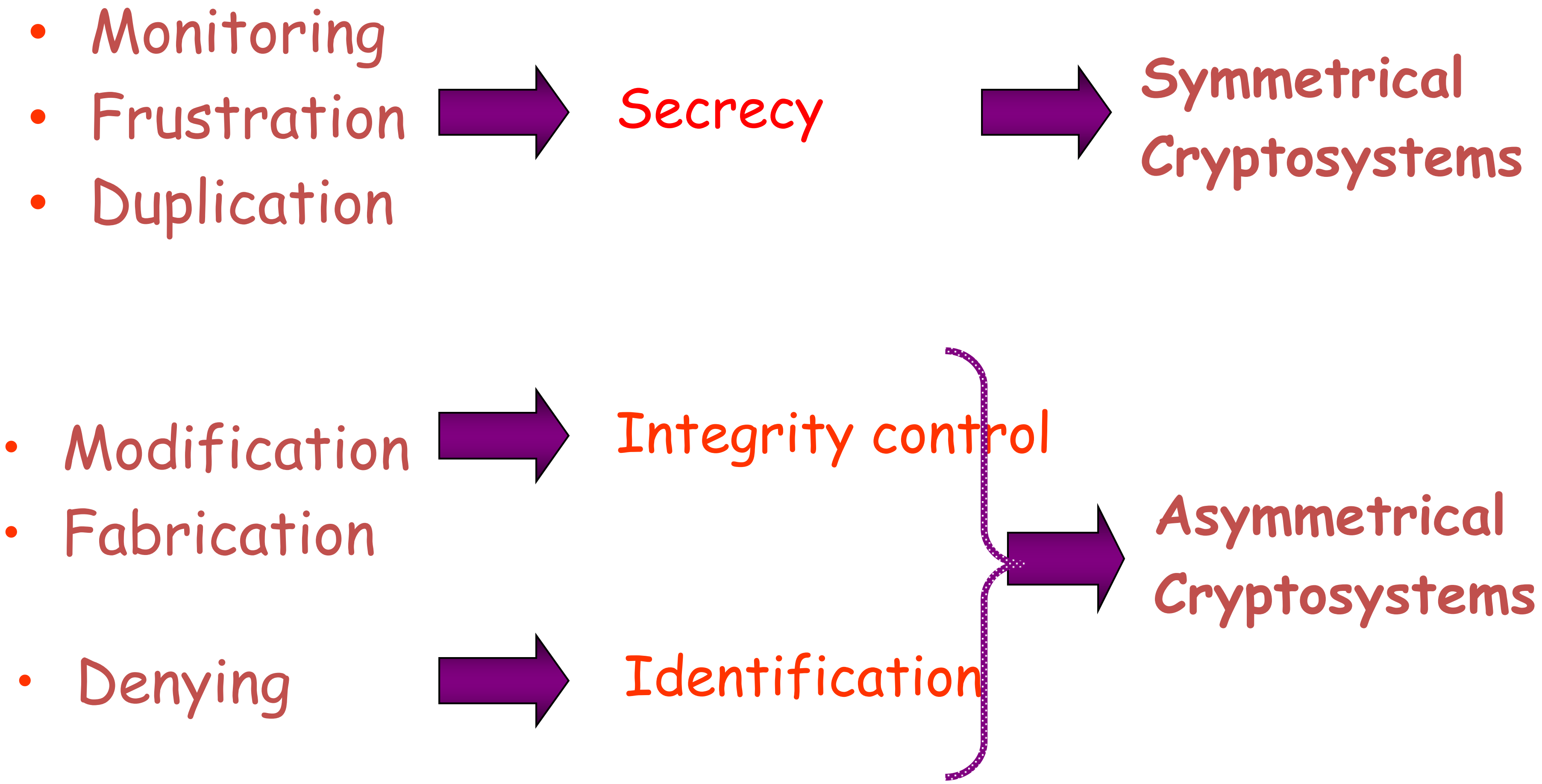
1. What will happen when Alice tries to send a message to Bob, encrypted with a key based on her DH shared secret?
2. Can Fred obtain the correct DH shared secret that would have been established had he not interfered?

Symmetrical vs Asymmetrical Cryptosystems

Symmetrical vs Asymmetrical Cryptosystems

Symmetrical	ASİMETRİK
☺Fast encryption, decryption	☹Slow encryption, decryption
☺Proved security (one time pad)	☹ Security has not proven
☹Key Distribution problem	☺ No key distribution problem
☹Open for cryptanalysis	☺ Resilient for cryptanalysis
Information Theory	Complexity Theory

Symmetrical vs Asymmetrical Cryptosystems



Q. Which cryptosystem is more secure?

Security is depend on key length and secrecy of private key.

For instance; DES is approximately 100 times faster than RSA but it is breakable...

Suggestions

To improve security:

Use symmetrical and asymmetrical cryptosystems combinations.

For instance: Use Triple DES for secrecy of your communication, and use Asymmetrical cryptosystems to exchange the keys and for their secrecy. (such that use PGP). Periodicaly generate and share new keys for Triple-DES.