# Introduction to Cryptography

J. Jiménez Urroz, UPC

CIMPA School, Manila, July 29, 2013

- Bit operation.

- Bit operation.

Multiplication of a $k$ digits integer by an $l$ digits integer.

$$\begin{array}{r} 11101 \\ \underline{\times 1101} \\ 101111001 \end{array}$$

.

- Bit operation.

Multiplication of a $k$ digits integer by an $l$ digits integer.

$$\begin{array}{r} 11101 \\ \times 1101 \\ \hline 101111001 \end{array}$$

It needs $kl$ bit operations.

- Bit operation.

Multiplication of a $k$ digits integer by an $l$ digits integer.

$$\begin{array}{r} 11101 \\ \times 1101 \\ \hline 101111001 \end{array}$$

It needs $kl$ bit operations.

- $f = O(g)$ if $f(n) \leq g(n)$ for all $n \in \mathbb{Z}^r$.

- Bit operation.

Multiplication of a $k$ digits integer by an $l$ digits integer.

$$\begin{array}{r} 11101 \\ \underline{\times 1101} \\ 101111001 \end{array}$$

It needs $kl$ bit operations.

- $f = O(g)$ if $f(n) \leq g(n)$ for all $n \in \mathbb{Z}^r$.

An algorithm in $n$-variables of $k_i$ bits each is called a polynomial time algorithm if the number of bit operations is $O(\prod_{i=1..n} k_i^{d_i})$

- The Euclidean algorithm to Find the gcd of two integers $a < b$, is a polynomial time algorithm.

- The Euclidean algorithm to Find the gcd of two integers $a < b$, is a polynomial time algorithm.
- Finding the inverse in $(\mathbb{Z}/m\mathbb{Z})^*$ can be done in polynomial time.

• The Euclidean algorithm to Find the gcd of two integers $a < b$, is a polynomial time algorithm.

• Finding the inverse in $(\mathbb{Z}/m\mathbb{Z})^*$ can be done in polynomial time.

• Given $n = pq$ where $p, q$ are distinct primes, it is equivalent to find $\varphi(n)$ than $p$ and $q$.

Exercise: Find an algorithm to compute $[\sqrt{n}]$ in polynomial time.

- The Euclidean algorithm to Find the gcd of two integers $a < b$, is a polynomial time algorithm.

- Finding the inverse in $(\mathbb{Z}/m\mathbb{Z})^*$ can be done in polynomial time.

- Given $n = pq$ where $p, q$ are distinct primes, it is equivalent to find $\varphi(n)$ than $p$ and $q$.

Exercise: Find an algorithm to compute $[\sqrt{n}]$ in polynomial time.

- Let $m = \prod p_i^{\alpha_i}$ If $(a, m) = 1$, $a^{L(m)} \equiv 1 \pmod{m}$, where $L(m) = \mathrm{lcm}\{\varphi(p_i^{\alpha_i})\}$

• The Euclidean algorithm to Find the gcd of two integers $a < b$, is a polynomial time algorithm.

• Finding the inverse in $(\mathbb{Z}/m\mathbb{Z})^*$ can be done in polynomial time.

• Given $n = pq$ where $p, q$ are distinct primes, it is equivalent to find $\varphi(n)$ than $p$ and $q$.

Exercise: Find an algorithm to compute $[\sqrt{n}]$ in polynomial time.

• Let $m = \prod p_i^{\alpha_i}$ If $(a, m) = 1$, $a^{L(m)} \equiv 1 \pmod{m}$, where $L(m) = \text{lcm}\{\varphi(p_i^{\alpha_i})\}$

• Exponentiation $a^n \pmod{m}$ is polynomial in $n$ and $m$

• The Euclidean algorithm to Find the gcd of two integers $a < b$, is a polynomial time algorithm.

• Finding the inverse in $(\mathbb{Z}/m\mathbb{Z})^*$ can be done in polynomial time.

• Given $n = pq$ where $p, q$ are distinct primes, it is equivalent to find $\varphi(n)$ than $p$ and $q$.

Exercise: Find an algorithm to compute $[\sqrt{n}]$ in polynomial time.

• Let $m = \prod p_i^{\alpha_i}$ If $(a, m) = 1$, $a^{L(m)} \equiv 1 \pmod{m}$, where $L(m) = \text{lcm}\{\varphi(p_i^{\alpha_i})\}$

• Exponentiation $a^n \pmod{m}$ is polynomial in $n$ and $m$

• Multiplying two elements of $\mathbb{F}_q$ needs $O((\log q)^3)$ operations while $a^k$, for $a \in \mathbb{F}_q$ and $k \in \mathbb{Z}$ needs $O((\log q)^3 \log k^3)$

**Theorem (Quadratic Reciprocity law)**

*Let $m, n$ two odd integers. Then*

$$\left(\frac{m}{n}\right)\left(\frac{n}{m}\right) = (-1)^{\frac{n-1}{2}\frac{m-1}{2}}$$

### Theorem (Quadratic Reciprocity law)

*Let $m, n$ two odd integers. Then*

$$\left(\frac{m}{n}\right)\left(\frac{n}{m}\right) = (-1)^{\frac{n-1}{2}\frac{m-1}{2}}$$

Proof. Consider $p, q$ primes, and $G = \sum_{i=0}^{q-1}\left(\frac{i}{q}\right)\xi^i$, where $\xi \in \mathbb{F}_{p^k}$ is a $q$-th root of unity. Then,

$$G^p = \sum_{i=0}^{q-1}\left(\frac{i}{q}\right)\xi^{ip} = \left(\frac{p}{q}\right)\sum_{i=0}^{q-1}\left(\frac{ip}{q}\right)\xi^{ip} = \left(\frac{p}{q}\right)G$$

But also

$$G^p = (G^2)^{(p-1)/2}G = ((-1)^{(q-1)/2}q)^{(p-1)/2}G$$

which finish the result.

$P$ is the set of plaintext messages, $C$ is the set of ciphertext message. A cryptosystem is a (biyective) function $f : P \to C$ such that given $m \in P$, $c = f(m)$ is easy to compute, but $m = f^{-1}(c)$ is very hard, unless an extra information is provided, which is called the key.

$P$ is the set of plaintext messages, $C$ is the set of ciphertext message. A cryptosystem is a (biyective) function $f : P \to C$ such that given $m \in P$, $c = f(m)$ is easy to compute, but $m = f^{-1}(c)$ is very hard, unless an extra information is provided, which is called the key.

Example: $f(m) = m + 3 \pmod{26}$. Will convert philippines into sklolsslqhv.

$P$ is the set of plaintext messages, $C$ is the set of ciphertext message. A cryptosystem is a (biyective) function $f : P \to C$ such that given $m \in P$, $c = f(m)$ is easy to compute, but $m = f^{-1}(c)$ is very hard, unless an extra information is provided, which is called the key.

Example: $f(m) = m + 3 \pmod{26}$. Will convert philippines into sklolsslqhv.

bjqhtrjytymjhnrufwjxjfwhmxhmttq

$P$ is the set of plaintext messages, $C$ is the set of ciphertext message. A cryptosystem is a (biyective) function $f : P \to C$ such that given $m \in P$, $c = f(m)$ is easy to compute, but $m = f^{-1}(c)$ is very hard, unless an extra information is provided, which is called the key.

Example: $f(m) = m + 3 \pmod{26}$. Will convert philippines into sklolsslqhv.

bjqhtrjytymjhnrufwjxjfwhmxhmttq

welcometothecimparesearchschool

- Hash Functions. Is any algorithm that maps data of variable length to data of a fixed length. (SHA-1,2,3. Secure Hash algorithm.) It does not need a key.

- Hash Functions. Is any algorithm that maps data of variable length to data of a fixed length. (SHA-1,2,3. Secure Hash algorithm.) It does not need a key.

It is easy to generate hash values from input data and easy to verify that the data matches the hash, but hard to 'fake' a hash value to hide malicious data.

- Hash Functions. Is any algorithm that maps data of variable length to data of a fixed length. (SHA-1,2,3. Secure Hash algorithm.) It does not need a key.

It is easy to generate hash values from input data and easy to verify that the data matches the hash, but hard to 'fake' a hash value to hide malicious data.

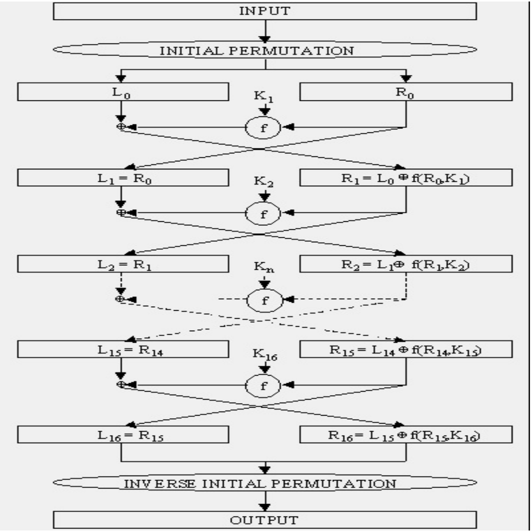
Good for ensuring data integrity. Any change made to the contents of a message will result in a different hash.

The same key is used to encrypt and decrypt the messages. It is also called symmetric encryption.

**Example:** DES (Data Encryption Standard, IBM, 1970)

The same key is used to encrypt and decrypt the messages. It is also called symmetric encryption.

**Example:** DES (Data Encryption Standard, IBM, 1970)



Secret key cryptography is ideally suited to encrypting messages.

- Advantages:
- – Encryption is fast and simple.
- – Less computer resources.
- – Good to encrypt your own files.

Secret key cryptography is ideally suited to encrypting messages.

- Advantages:

– Encryption is fast and simple.

– Less computer resources.

– Good to encrypt your own files.

- Disadvantages:

–Secure channel for secret key exchange.

–Ensuring privacy of keys is difficult.

–Origin and authenticity of message cannot be guaranteed.

The encryption key $k_e$ and the decryption key $k_d$ are different.
Depends upon the existence of so-called one-way functions, or mathematical functions that are easy to compute whereas their inverse function is very difficult to compute without the key.

The encryption key $k_e$ and the decryption key $k_d$ are different. Depends upon the existence of so-called one-way functions, or mathematical functions that are easy to compute whereas their inverse function is very difficult to compute without the key.

The algorithm to encrypt is public, while the keys are secret. Up to 1976 to know how to encipher and decipher were regarded as equivalent. Is it in this year when Diffie-Hellman invented public key cryptography.

The encryption key $k_e$ and the decryption key $k_d$ are different. Depends upon the existence of so-called one-way functions, or mathematical functions that are easy to compute whereas their inverse function is very difficult to compute without the key.

The algorithm to encrypt is public, while the keys are secret. Up to 1976 to know how to encipher and decipher were regarded as equivalent. Is it in this year when Diffie-Hellman invented public key cryptography.

It is based on the use of a trapdoor function. A biyective function $f : P \to P$ easy to compute, but very hard to find $f^{-1}$ in any single value, unless an additional information is provided, the deciphering key $K_d$, which is kept secret.

# Authentication

One of the most important algorithms is digital signatures.

# Authentication

One of the most important algorithms is digital signatures.

A can send, together with the message, compute $f_B(f_A^{-1}(P))$

# Authentication

One of the most important algorithms is digital signatures.

A can send, together with the message, compute $f_B(f_A^{-1}(P))$

In digital signatures it is often used hash functions. Changing the person, content or the date of the message would change the hash value.

# Authentication

One of the most important algorithms is digital signatures.

A can send, together with the message, compute $f_B(f_A^{-1}(P))$

In digital signatures it is often used hash functions. Changing the person, content or the date of the message would change the hash value.

Public key cryptosystems are often used to send the keys of a symmetric scheme. This is called key exchange. In order to ensure security, probabilistic cryptosystems are used: the same plaintext has many different cipher text, depending on a random parameter.

**Idea:** Inverting without the trapdoor function, allows to solve a difficult mathematical problem.

**Idea:** Inverting without the trapdoor function, allows to solve a difficult mathematical problem. RSA is based on the factorization problem: given $n = pq$, a product of two large primes, find $p$ and $q$.

**Idea:** Inverting without the trapdoor function, allows to solve a difficult mathematical problem. RSA is based on the factorization problem: given $n = pq$, a product of two large primes, find $p$ and $q$.

4294967297

**Idea:** Inverting without the trapdoor function, allows to solve a difficult mathematical problem. RSA is based on the factorization problem: given $n = pq$, a product of two large primes, find $p$ and $q$.

$4294967297 = 2^{2^5} + 1$

**Idea:** Inverting without the trapdoor function, allows to solve a difficult mathematical problem. RSA is based on the factorization problem: given $n = pq$, a product of two large primes, find $p$ and $q$.

$4294967297 = 2^{2^5} + 1 = 641 \times 6700417$

**Idea:** Inverting without the trapdoor function, allows to solve a difficult mathematical problem. RSA is based on the factorization problem: given $n = pq$, a product of two large primes, find $p$ and $q$.

4294967297$= 2^{2^5} + 1 =$ 641 $\times$ 6700417
Each user $A$ selects two huge primes, $p_A$ and $q_A$ and computes $n_A = p_A q_A$. Then the user selects a random $1 < e_A < \varphi(n_A)$ coprime to $\varphi(n_A)$to be the public key and computes the inverse $e_A^{-1} = d_A \pmod{\varphi(n_A)}$, which will be the private key. $c = m^{e_A}$.
$m = c_A^d$

**Idea:** Inverting without the trapdoor function, allows to solve a difficult mathematical problem. RSA is based on the factorization problem: given $n = pq$, a product of two large primes, find $p$ and $q$.

4294967297$= 2^{2^5} + 1 = 641 \times 6700417$
Each user $A$ selects two huge primes, $p_A$ and $q_A$ and computes $n_A = p_A q_A$. Then the user selects a random $1 < e_A < \varphi(n_A)$ coprime to $\varphi(n_A)$to be the public key and computes the inverse $e_A^{-1} = d_A \pmod{\varphi(n_A)}$, which will be the private key. $c = m^{e_A}$. $m = c_A^d$

Suppose we know $n = pq$ and $m$ such that $a^m \equiv 1 \pmod{n}$ for all $(a, m) = 1$. Find the factorization of $n$.

**Idea:** Inverting without the trapdoor function, allows to solve a difficult mathematical problem. RSA is based on the factorization problem: given $n = pq$, a product of two large primes, find $p$ and $q$.

4294967297$= 2^{2^5} + 1 = 641 \times 6700417$
Each user $A$ selects two huge primes, $p_A$ and $q_A$ and computes $n_A = p_A q_A$. Then the user selects a random $1 < e_A < \varphi(n_A)$ coprime to $\varphi(n_A)$to be the public key and computes the inverse $e_A^{-1} = d_A \pmod{\varphi(n_A)}$, which will be the private key. $c = m^{e_A}$.
$m = c_A^d$

Suppose we know $n = pq$ and $m$ such that $a^m \equiv 1 \pmod{n}$ for all $(a, m) = 1$. Find the factorization of $n$.

**Exercise:** How to make the digital signature $f_A^{-1} f_B$ when $n_A$ and $n_B$ are different?