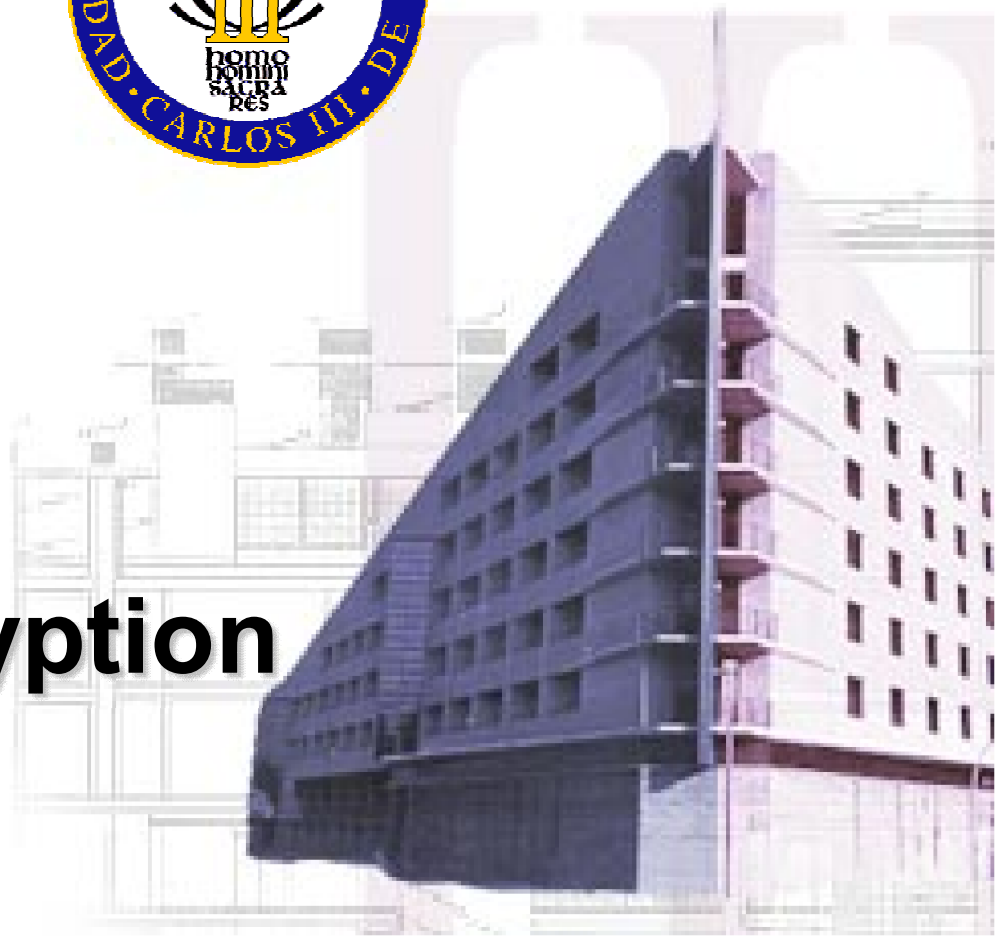
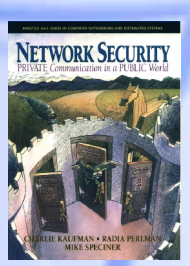


CHAPTER 4: Public Key Encryption

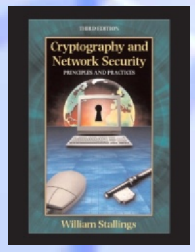
Coding Techniques

Mario Muñoz Organero

Basic bibliography

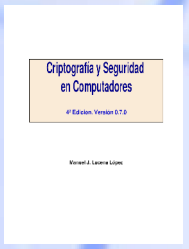


- **CHARLIE KAUFMAN, RADIA PERLMAN, MIKE SPECINER: Network Security: Private Communication in a Public World. Prentice Hall; First edition (Marzo, 1995). ISBN: 0130614661** ***[L/S 004.056 KAU]***
  - *Chapter 2: 2.5, Chapter 5: 5.2, Chapter 5: 5.3 & Chapter 6*

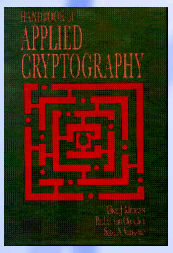


- **[2] STALLINGS, WILLIAMS: Cryptography and network security. Principles and practice. Fourth edition. Prentice Hall 2006. ISBN:0131873164 http://williamstallings.com/** ***[L/S 004.7 TAN]***
  - *Chapters 4, 8 & 9*

Complementary bibliography



- **[3] LUCENA, J. MANUEL: Criptografía y seguridad en computadores. Cuarta edición. http://wwwdi.ujaen.es/~mlucena/bin/cysec4.zip**
  - Chapters 5, 6 & 12



- **[4] MENESES, ALFRED: Handbook of applied cryptography. CRC Press. 1996. http://www.cacr.math.uwaterloo.ca/hac/**
  - Chapters 2 & 8

Specific bibliography on Elliptic Curves

- *Implementing Elliptic Curve Cryptography* by Michael Rosing
- *Guide to Elliptic Curve Cryptography* by Darrel Hankerson
- *Elliptic Curves in Cryptography* by I. Blake

Chapter Index

- **Introduction into encryption using asymmetric keys**
- **Necessary mathematical basis**
  - Number theory for modular arithmetic
- **Algorithms based on exponentiation:**
  - RSA
  - ElGamal
- **Elliptic curves**
- **Cryptography based on elliptic curves**
  - ElGamal
  - Diffie-Hellman

Introduction: Public-Key Encryption

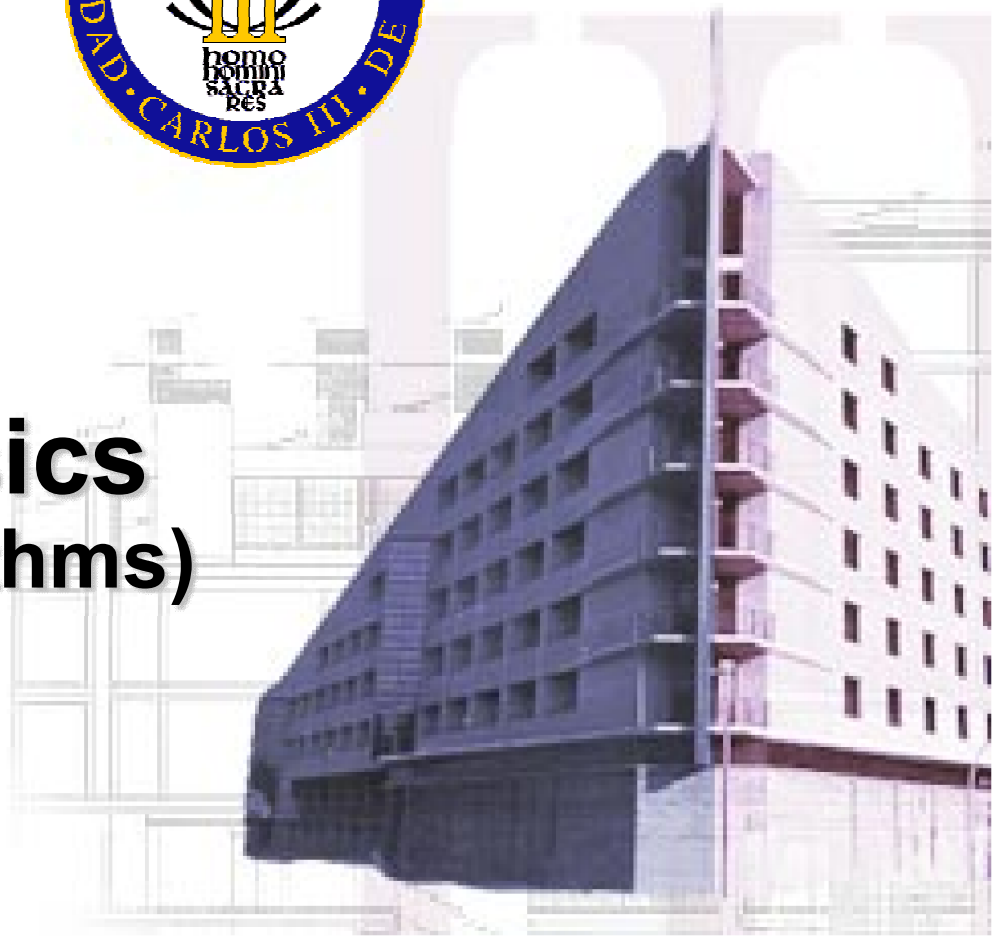
- **Introduced by Whitfield Diffie and Martin Hellman in the 70's**
- **Fundamental novelty:**
  - No symmetric keys, but asymmetric key pairs
- **Larger key length than in symmetric encryption**
  - E.g., 1024 bits
- **Two different keys instead of a single one:**
  - K_{pr}: private key
  - K_{PU}: public key
- **One for decryption and one for encryption**

Public-Key Encryption: Security

- **In order to be secure it must be (computationally) extremely difficult to calculate one key of the key pair if the second key is known**
  - The calculation of the private key must be computationally infeasible based on the public key
- **The two main applications of such algorithms are:**
  - Encryption (confidentiality).
  - Authentication (digital signatures).

Classification of Asymmetric Algorithms

- **Two main families of asymmetric algorithms applied to cryptography depending on the mathematical functions used**
  - Exponentiation algorithms
  - Elliptic curve algorithms



Mathematical Basics
(exponentiation algorithms)

Some Number Theory for Modular Arithmetic

Departamento de INGENIERIA TELEMATICA
www.it.uc3m.es

Keep Focus

- **Target: What do we want to do?**
  - A Cryptographic operation like coding or decoding
  - What do we want to cipher? A message or document
- **In asymmetric cryptography based on exponentiation :**
  - A message is a concatenation of numbers (truncated to a block size)
  - Cryptography consists of performing exponentiation operations on each of this numbers

Finite set of cryptograms needed → Congruence concept

- **Modular arithmetic (discrete mathematics):**
  - Mathematical basis for the operations of public-key encryption
- **Concept of congruence ("≡") :**
  - Consider two integers **a** and **b**
  - **a** is congruent to **b** in the modulo or body **n** in **Zn** if and only if:
    - ✓ An integer **k** exists that divides **(a-b)** without remainder
    - ✓ **n**: "modulus of the congruence"

$$a - b = k * n$$

$$a \equiv_n b$$

$$a \equiv b \bmod n$$

Complete Set of Residues (CSR)

- **For any positive integer number n, the CSR is {0, 1, 2, ..., n-1}. That means:**
  - $\forall\ a \in Z \quad \exists!\ r_i \in CSR\ /\ a \equiv r_i \bmod n$
    - ✓ Only one representation per residue
- **Examples:**
  - CSR (11) = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
  - CSR (6) = {0, 1, 2, 3, 4, 5} = {12, 7, 20, 9, 16, 35}
  - The second set is equivalent:
    - ✓ $12 \rightarrow 0$, $7 \rightarrow 1$...
    - ✓ Only different "representation" of the same number

Divisibility of Numbers

- **Important in cryptography:**
  - Find **gcd(a,b)**
    - ✓ Greatest common divisor for two integers **a** and **b**
- **Example in cryptography:**
  - Used to ensure the existence of the inverse in Z_n
  - Condition: base **a** and modulus **n** are coprime:
    - ✓ I.e.: **gcd(a,n) = 1**
    - ✓ Coprime = relatively prime

Euclid's Algorithm

- Used to calculate the gcd of a and b
- If x divides both a and b, x also divides:
  - a mod b and b mod a
    - ✓ Because of commutativity property
- Partial proof:
  - x,a,b,a′,b′,k: integer numbers

  a) If x divides both a and b ⇒ a = x*a′ and b = x*b′

  b) Therefore: a-k*b = x*a′-k*x*b′

  ⇒ a-k*b = x(a′-k*b′)

  c) Conclusion: x also divides (a-k*b)

Euclid's Algorithm

- If **x** divides **a** and **b** $\Rightarrow$
  - **x** divides "**a mod b**" (**a-k*b**)
  - **x** divides "**b mod a**" (**b-k*a**)
- **Algorithm:**
  - If **a > b** → $r_0 = a$ and $r_1 = b$
  - Calculate $r_2 = r_0 \bmod r_1$
  - Iterate $r_n = r_{n-2} \bmod r_{n-1}$
  - The last remainder before reaching 0 is the **gcd(a, b)**

Divisibility with Euclid's Algorithm: Example

gcd (148, 40)

$148 = 3 * 40 + 28$

$40 = 1 * 28 + 12$

$28 = 2 * 12 + 4$

$12 = 3 * 4 + 0$

gcd (148, 40) = 4

$148 = 2^2 * 37$

$40 = 2^3 * 5$

Common factor: $2^2 = 4$

No common factor

$385 = 5 * 7 * 11$

$78 = 2 * 3 * 13$

This is important for cryptography

gcd (385, 78)

$385 = 4 * 78 + 73$

$78 = 1 * 73 + 5$

$73 = 14 * 5 + 3$

$5 = 1 * 3 + 2$

$3 = 1 * 2 + 1$

$2 = 2 * 1 + 0$

gcd (385, 78) = 1

Inverses in Z_n (I)

- **For cryptography, inverting an operation must be possible in order to recover the plaintext from the ciphertext (decryption)**
- **Terminology problem:**
  - In mathematical terms encryption is a function,
  - In colloquial speech, we use the following "analogies"
    - ✓ The encryption operation is "multiplication"
    - ✓ The decryption operation is "division"
  - This analogy is true also for the inverses in Z_n
- **Thus, if in an encryption operation the function returns a value a in Z_n, we have to find an inverse $a^{-1} \bmod n$ for the decryption operation.**
- **In other words:**

Inverses in Z_n (II)

- If `a*x mod n = 1` then `x` is the multiplicative inverse (a^{-1}) of `a` in the modulo `n`
- Problem:
  - Inverses do not always exist (rarely in reality)
- Next slides show when the inverse exists

Existence of the Inverse for Coprime Numbers

- **There is an inverse a^{-1} in $\mathbf{mod\ n}$ iff $\mathbf{gcd(a,n) = 1}$**
  - (iff means "if and only if")
- **Only when distinct numbers result from**

  $\mathbf{a*i\ mod\ n}$ (for all $\mathbf{i \in Z_n}$)
- **Example (Brute-force test):**
  - **Reminder: gcd (a, n) = 1 $\Rightarrow$ $\exists$ x ! 0<x<n / a $*$ x mod n = 1**
  - $\mathbf{a = 4}$ and $\mathbf{n = 9}$. Values for i = {1, 2, 3, 4, 5, 6, 7, 8}
  - 4∗1 mod 9 = 4 4∗2 mod 9 = 8 4∗3 mod 9 = 3
  - 4∗4 mod 9 = 7 4∗5 mod 9 = 2 4∗6 mod 9 = 6
  - 4∗7 mod 9 = 1 4∗8 mod 9 = 5
  - Inverse for 4 is 7
- **Multiplication can be seen as 'encryption' here**
  - Simple substitution scheme
- **Can be reversed for 'decryption'**

Non-Existence of an Inverse (for Non-Coprime Numbers)

- **What if a and n are not coprime?**
  - I.e.: $\gcd(a,n) \neq 1$
  - No x exists such that $a*x \bmod n = 1$ for $0<x<n$
- **Example (Brute-force test):**
  - a = 3; n = 6, values for i = {1, 2, 3, 4, 5}

    $3*1 \bmod 6 = 3$ $3*2 \bmod 6 = 0$ $3*3 \bmod 6 = 3$

    $3*4 \bmod 6 = 0$ $3*5 \bmod 6 = 3$
- **There is no inverse for a=3 in the set!**

If n=10

- How many numbers will have inverse?

(A∗B) mod 10

	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9
2	2	4	6	8	0	2	4	6	8
3	3	6	9	2	5	8	1	4	7
4	4	8	2	6	0	4	8	2	6
5	5	0	5	0	5	0	5	0	5
6	6	2	8	4	0	6	2	8	4
7	7	4	1	8	5	2	9	6	3
8	8	6	4	2	0	8	6	4	2
9	9	8	7	6	5	4	3	2	1

10=2*5

=(2-1)*(5-1)

Reduced Set of Residues RSR (1)

- **Reduced Set of Residues (RSR):**
  - Subset {0, 1, ... n_i, ... $n-1$} of numbers in Z_n
    - ✓ Being coprime to n
  - If n prime: All elements in RSR coprime to n
- **"zero" is no solution, thus:**
  - RSR = {1, ..., n_i, ... n-1} / gcd (n_i, n) = 1
- **Examples:**
  - RSR mod 8 = {1, 3, 5, 7}
  - RSR mod 5 = {1, 2, 3, 4}

Reduced Set of Residues RSR (2)

- Why is the RSR useful in cryptography?
- The knowledge of the RSR allows application of an algorithm to find out the multiplicative inverse of x in Z_n
- The Euler phi function $\phi(n)$ helps in this task

Euler Phi Function $\phi(n)$

- **Returns the number of elements in a RSR**
- **Different cases for `n` (four forms):**
  - a) `n` is a prime number
  - b) `n` can be represented as $n = p^k$
    - ✓ **`p`: prime, `k`: integer**
  - c) `n` is a product $n = p*q$, `p` and `q` prime
  - d) `n` is any number (generic case).

$$n = p_1^{e1} * p_2^{e2} * \ldots * p_t^{et} = \prod_{i=1}^{t} p_i^{ei}$$

➢ **Let us explain the four cases**

Euler Phi Function $\phi(n)$ (n prime)

- **Case 1: `n` is a prime number**
  - The RSR is the CSR without the number 0
  - So $\phi(n)$ is the number of elements in CSR **- 1**
    - ✓ $\phi(n) = n-1$ (*Used in ElGamal and DSS*)
  - Explanation:
    - ✓ `n` is prime ➔ all elements in CSR are coprime
    - ✓ Except for the zero
- **Examples:**
  - RSR(7) = {1,2,3,4,5,6} (six elements)
    - ✓ $\phi(7) = n-1 = 7-1 = 6$
  - Similar: $\phi(11) = 11-1 = 10;$ $\phi(23) = 23-1 = 22$

Euler Phi Function $\phi(n)$ ($n=p^k$)

- **Case 2: $n = p^k$, p: prime, k: integer**
  - $\phi(n) = \phi(p^k) = p^k - p^{k-1} = p^{k-1}(p-1)$
  - From the p^k elements in CSR:
    - ✓ Remove the multiples of p
      - ➢ $1*p, 2*p, 3*p, \ldots, (p^{k-1}-1)*p$
    - ✓ And the zero
  - $\phi(p^k) = n-1-(p^{k-1}-1) = p^k - p^{k-1} = p^{k-1}(p-1)$
- **Examples:**
  - RSR(16) = {1,3,5,7,9,11,13,15} (eight elements)
    - ✓ $\phi(16) = \phi(2^4) = 2^{4-1}(2-1) = 2^3*1 = 8$
  - Similar: n = 12^5
    - ✓ $\phi(125) = \phi(5^3) = 5^{3-1}*(5-1) = 5^2*4 = 25*4 = 100$

Euler Phi Function $\phi(n)$ ($n=p*q$)

- **Case 3: $n = p*q$, p and q prime numbers**
  - $\phi(n) = \phi(p*q) = \phi(p)*\phi(q) = (p-1)(q-1)$
  - From the $p*q$ elements in CSR:
    - ✓ Remove all multiples of p: $1*p,\ 2*p,\ \ldots\ (q-1)*p$,
    - ✓ Remove all multiples of q: $1*q,\ 2*q,\ \ldots\ (p-1)*q$
    - ✓ And the zero.
  - $\phi(p*q) = p*q - [(q-1)+(p-1)+1]$
    $= p*q - q - p + 1$
    $= (p-1)(q-1)$
- **Examples:**
  - RSR(15) = {1,2,4,7,8,11,13,14} (eight elements)
    - ✓ $\phi(15) = \phi(3*5) = (3-1)(5-1) = 2*4 = 8$
  - $\phi(143) = \phi(11*13) = (11-1)(13-1) = 10*12=120$
- **Case 3 is used a lot:**
  - Basis for RSA encryption (de-facto standard)

Euler Phi Function $\phi(n)$ (n=generic)

- **Case 4: $n = p_1^{e1} * p_2^{e2} * \ldots * p_t^{et}$**
  - p_i prime

$$\Phi(n) = \prod_{i=1}^{n} p_i^{e_i - 1}(p_i - 1)$$

  - Not demonstrated easily...
- **Examples:**
  - RSR(20) = {1, 3, 7, 9, 11, 13, 17, 19} (8 elem.)
    - ✓ $\phi(20) = \phi(2^2 * 5) = 2^{2-1}(2-1) * 5^{1-1}(5-1)$

$$= 2^1 * 1 * 1 * 4 = 8$$

  - $\phi(360) = \phi(2^3 * 3^2 * 5)$

$$= 2^{3-1}(2-1) * 3^{2-1}(3-1) * 5^{1-1}(5-1) = 96$$

Euler's Theorem

- If $\gcd(a,n) = 1 \Rightarrow a^{\phi(n)} \bmod n = 1$
- Now we combine the following into one equation:
  - $a^{\phi(n)} \bmod n = 1$
  - $a*x \bmod n = 1$
- Multiplying by a^{-1}
  - The operation is possible since $\gcd(a,n) = 1$
    - ✓ A unique value for an inverse exists in Z_n
  - $a^{\phi(n)} * a^{-1} \bmod n = x \bmod n$
  - $x = a^{\phi(n)-1} \bmod n$
- The value x is the inverse of a in Z_n
  - $a^{-1} = a^{\phi(n)-1} \bmod n$

Calculation of the Inverse using Euler's Theorem

- **Example:**
  - Get the inverse **x** to 4 in modulo 9 $\Rightarrow$ **x = inv(4,9)**
- **First problem: Existence of the inverse:**
  - Does an **x** exist such that
    - ✓ **a*x mod n = 4*x mod 9 = 1**?
    - ✓ Check: **gcd(4,9) = 1**
  - ➢ Yes, although 4 and 9 are not prime.
- **Calculate the inverse using Euler's Theorem:**
  - $\phi(9) = 6 \Rightarrow x = 4^{6-1} \bmod 9 = 7$

    $\Rightarrow 7*4 = 28 \bmod 9 = 1$
- **Therefore: inv(4,9) = 7 and inv(7,9) = 4**

Euler's Theorem for n = p*q

- If a is coprime with respect to n and n=p*q (p and q primes), Euler's Theorem verifies also with respect p and q.
- For example
  - $n = p*q \Rightarrow \phi(n) = (p-1)(q-1)$
  - $\forall\ a\ /\ gcd(a,\{p,q\}) = 1$
- Then:
  - $a^{\phi(n)} \bmod p = 1$
  - $a^{\phi(n)} \bmod q = 1$

Example: Euler's Theorem for n = p*q

- **Assume: n = p*q = 7*11 = 77**
  - $\phi(n) = (p-1)(q-1) = (7-1)(11-1) = 6*10 = 60$
- **If k = 1,2,3,...**
  - $a = k*7$: $a^{\phi(n)} \bmod n = k*7^{60} \bmod 77 = 56$
  - $a = k*11$: $a^{\phi(n)} \bmod n = k*11^{60} \bmod 77 = 22$
  - $\forall a \neq k*7,11$: $a^{\phi(n)} \bmod n = a^{60} \bmod 77 = 1$
- **Therefore:**
  - $\forall a \neq k*7,11$: $a^{\phi(n)} \bmod p = a^{60} \bmod 7 = 1$
    $a^{\phi(n)} \bmod q = a^{60} \bmod 11 = 1$
- **Else:**
  - $a^{\phi(n)} \bmod p = 0$
  - $a^{\phi(n)} \bmod q = 0$

Reducibility

- **If we want to compute 81^{30} mod 91**
  - How to do that?
- **Ordinary calculator:**
  - 1,797010299914431210413179829509 6e+57 mod 91 [losing accuracy]
- **Applying reducibility:**
  - $81^{30} \bmod 91 = (81^5 \bmod 91)^6 \bmod 91 = 9^6 \bmod 91 = 1$
- **Result:**
  - $81^6 \bmod 91 = 1$

What to do if $\phi(n)$ is not known?

- **When the values of '$\phi(n)$' and / or 'a' are large computing "$a^{\phi(n)-1} \bmod n$" to obtain the inverse requires a lot of effort**
  - Using the reducibility property repeatedly is not the best way.
- **If $\phi(n)$ is not known or Euler's Theorem is not convenient:**
  - Use the **Extended Euclid's Algorithm** to find the inverse of a in Z_n
  - Very fast and feasible method

Extended Euclid's Algorithm

- Based on Euclid's algorithm but besides the remainders it also takes into account the **quotients** in each iteration:
  - Both `a mod n` and `n div a`
- The algorithm performs a series of iterations in the form:

$$g_i = nu_i + av_i$$

- The last g_i is the `gcd(a,n)`. If `a` and `n` are coprimes this equals to:

$$1 = nu_i + av_i$$

Computing inverses with the EEA

Find `inv(9,25)` by Euclid's algorithm.

a) 25 = 2*9 + 7

b) 9 = 1*7 + 2

c) 7 = 3*2 + 1

d) 2 = 2*1 + 0

$7 = 25 - 2*9$

$2 = 9 - 1*7$

$1 = 7 - 3*2$

remainders

$7 = 25 - 2*9$

$2 = 9 - 1*(25 - 2*9) = 3*9 - 1*25$

$1 = (25 - 2*9) - 3*(3*9 - 1*25)$

$1 = 4*25 - 11*9 \mod 25$

Remainders Table

	2	1	3	2	
25	**9**	**7**	**2**	**1**	**0**

Inv (9,25) = -11

-11 + 25 = 14

inv (9, 25) = 14

Algorithm for inverse calculation

To find `x = inv (A,B)`

Do $(g_0, g_1, u_0, u_1, v_0, v_1, i) = (B, A, 1, 0, 0, 1, 1)$

While $g_i \neq 0$ **do**

y_{i+1} = integer part of (g_{i-1}/g_i)

$g_{i+1} = g_{i-1} - y_{i+1} * g_i$

$u_{i+1} = u_{i-1} - y_{i+1} * u_i$

$v_{i+1} = v_{i-1} - y_{i+1} * v_i$

$i = i+1$

If $(v_{i-1} < 0)$ **do**

$v_{i-1} = v_{i-1} + B$

$x = v_{i-1}$

x = inv (A, B)

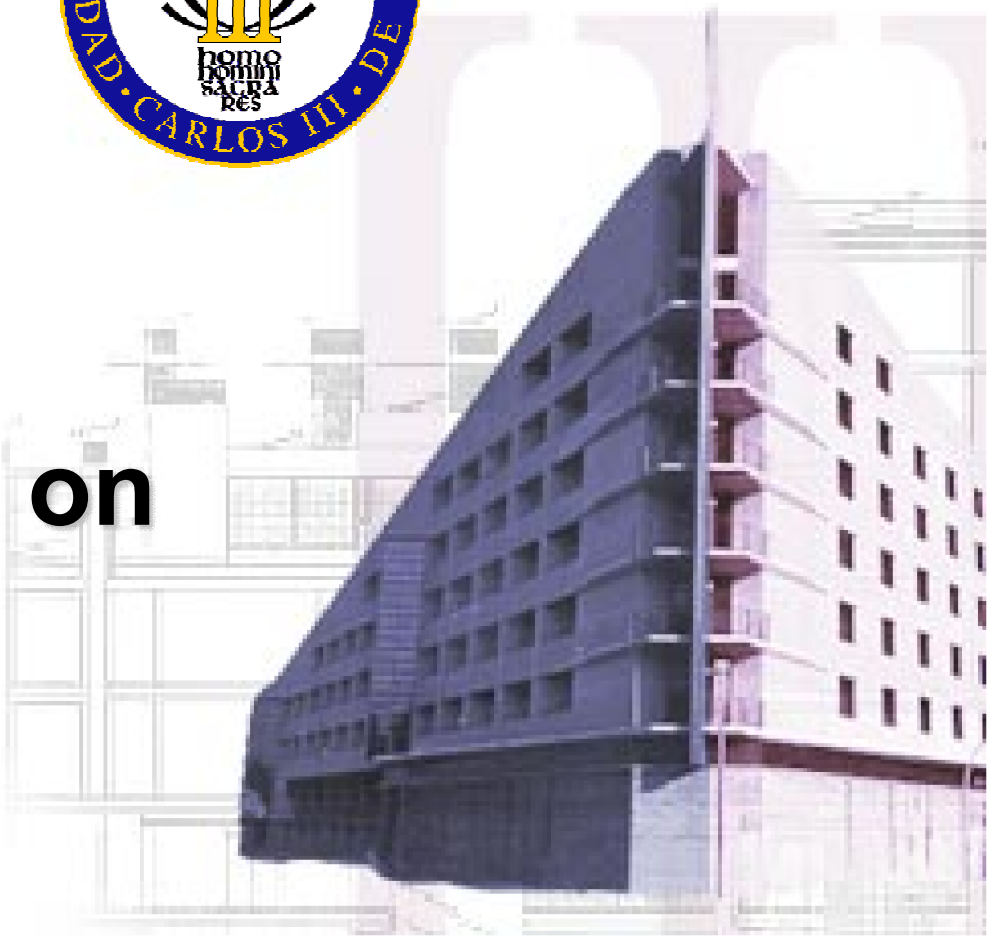
x = inv (9, 25)

i	y_i	g_i	u_i	v_i
0	-	25	1	0
1	-	9	0	1
2	2	7	1	-2
3	1	2	-1	3
4	3	1	4	-11
5	2	0	-9	25

x = inv (9, 25) = -11+25 = 14

Example

Algorithms based on Exponentiation



Departamento de INGENIERIA TELEMATICA
www.it.uc3m.es

Algorithms based on Exponentiation

- **Basic assumption of all these algorithms:**
  - Calculation of exponents is sufficiently fast
  - Solving a discrete logarithm or factorizing large numbers is slow
- **Some well-known algorithms:**
  - RSA
  - ElGamal (and its generalized version)
  - Rabin
  - McEliece
  - Merkle-Hellman
  - Chor-Rivest
  - Goldwasser-Micali
  - Blum-Goldwasser

RSA

- **Invented in 1978 by:**
  - Ron Rivest, Adi Shamir, and Leonard Adleman
- **Very easy to understand**
- **Believed one of the most secure asymmetric encryption schemes**
  - No one has proven there is no mechanism to break it…
  - Security based on the difficulty to factorize large numbers
- **Pair of keys can be used for encryption and authentication**
  - Public-key and private-key
    - ✓ Obtained from a number **n** which is the product of two large prime numbers **p** and **q**
- **The key length is variable (e.g., 512 / 1024 bits)**
- **The plaintext must be smaller than the key length**
  - RSA mainly intended for encryption of symmetric keys

RSA: The Basic Mathematics I

- **The mathematics of RSA can be expressed as:**
  1. Find two large prime numbers **p** and **q**, keep them secret
     - Example: Two 1024-bit numbers
  2. Calculate the product **n = p*q**
  3. Choose **e** such that:
     1. **1 < e < Φ(n)**
     2. **e** and **Φ(n)=(p-1)(q-1)** are coprime (*case 3 of Euler phi*)
        That is: they have no common divisor other than 1
     3. **e** is not needed to be prime, but must be odd
     4. **(p-1)(q-1)** cannot be prime:
        Its two factors **(p-1)** and **(q-1)** are even numbers
     - Then: **e** will have an inverse modulo **Φ(n)=(p-1)(q-1)**
  4. The public key is **(e,n)**

RSA: The Basic Mathematics II

5. Calculate **d** as the multiplicative inverse to **e** modulo $\Phi(n) = (p-1)(q-1)$
   - That is: $d*e = 1 \pmod{\Phi(n)}$
6. The private key is **(d,n)**
7. Encryption is performed according to:
   - $c = m^e \pmod{n}$
8. Decryption is performed according to:
   - $m = c^d \pmod{n}$

RSA

- **Decryption obtains the following message:**

$$c^d = (m^e)^d = m^{ed} = m^{k(p-1)(q-1)+1} = (m^k)^{(p-1)(q-1)} m$$

- **Considerations:**
  - **p** and **q** must have a large number of bits
  - If an attacker wants to recover the private key from the public key he/she must know **p** and **q** factors of **n** (computationally expensive).

Example: RSA

n = 91 = 7∗13; $\phi(n) = \phi(7*13) = (7-1)(13-1) = 72$ M = 48

Choose: e = 5 , gcd (5,72) = 1 and d = inv(5,72) = 29

Encryption:

$C = M^e \bmod n = 48^5 \bmod 91 = 5245803968 \bmod 91 = 55$

Decryption:

$M = C^d \bmod n = 55^{29} \bmod 91 = 48$ ... **but 55^{29} is a *large number!***

55^{29} is a number with 51 digits...
55^{29} = 295473131755644748809642476009391248226165771484375
How to accelerate this operation of decryption?

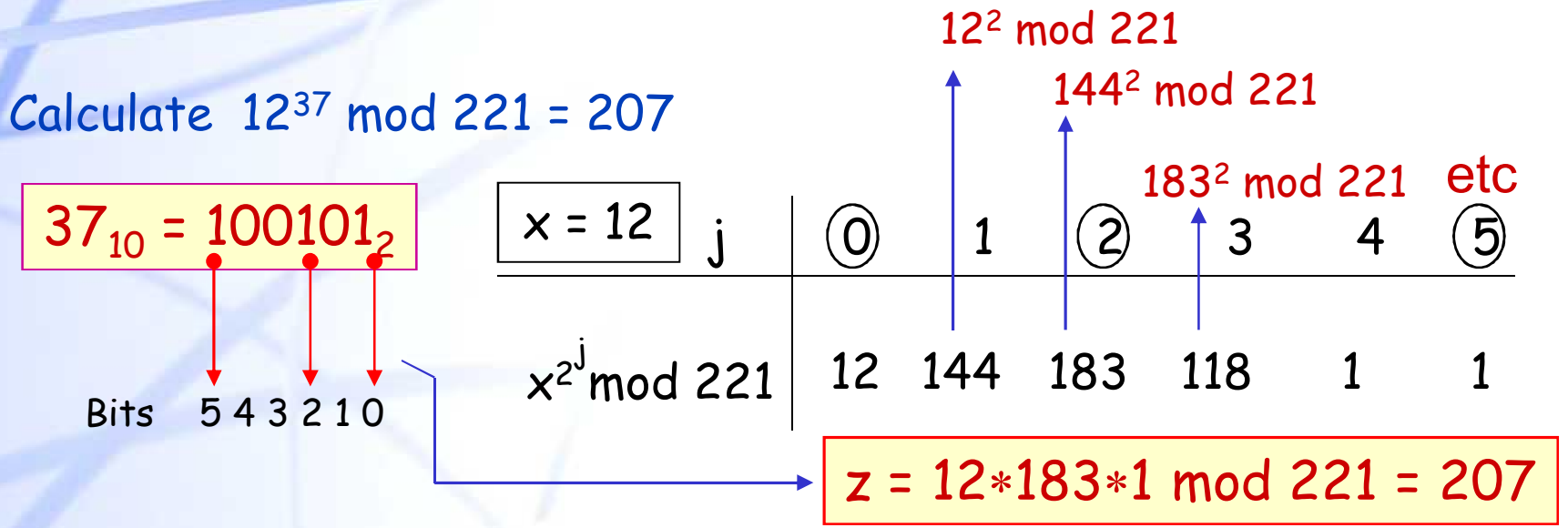
1st option: *use reducibility* bad

More optimization?
Fast exponentiation ok

One Method of Fast Exponentiation

- **Consider $x^y \bmod n$**
  - The y exponent represented in **binary**
- **Compute $x^y = x^{2^j}$, $j = 0 \ldots n-1$**
  - n being the number of bits of y
  - Consider only the products where the bit at position j is 1
- **Example:**
  - Calculate $12^{37} \bmod 221 = 207$
  - 12^{37} is a number with 40 digits!

One Method of Fast Exponentiation: Example



- Original version: 36 multiplications and reductions modulo 211
  - **72 operations**
- Fast version: 5 multiplications (for `j=0` the value is `x`) plus 5 reductions modulo 221 plus two final multiplications
  - **10 operations**
- Saves more than 80% ☺

The other history of RSA

- Rivest, Shamir & Adleman are the authors of RSA but an asymmetric encryption algorithm based on the complexity of factoring big numbers as the unidirectional function was discovered previously...
- In 1969 the Government Communications Headquarters (GCHQ) in Great Britain begins to work in the idea of distributing keys by means of asymmetric encryption. In 1973, the mathematic Clifford Cocks reachs the same conclusion than the creators of RSA.
- Unfortunately, this work was considered top secret and its content is not publicized nor patented, something that Diffie and Hellman made in 1976 with their key exchange algorithms and in 1978 the creators of RSA algorithm.

Choosing the prime numbers

- **Prime numbers `p` and `q` must be chosen appropriately. Follow these guidelines:**
  - `p` and `q` size **must** difer in a few digits.
  - `p` and `q` **must not** be close primes.
  - Minimum length of `p` and `q` : **250 bits**.
  - Values of `p-1` and `q-1` of the Euler function **must have** large prime factors.
  - The gcd between `p-1` and `q-1` **must be** small.

Secure prime numbers

Secure primes: r large prime is chosen so that:

p = 2***r**+1 and **q** = 2***p**+1 are also primes

EXAMPLE: If r is the 4 digits prime 1019:

p = 2∗1019 + 1 = 2039 Prime ok

q = 2∗2039 + 1 = 4079 Prime ok

p-1 = 2038; q-1 = 4078

p-1 = 2∗1019; q-1 = 2∗2.039 gcd (p, q) = 2

The modulo would be n = p∗q = 8.317.081

Attack to the key: Factoring n

- What is the algorithm strength?
- An intruder who wants to know the secret key from the values of n and e must solve the problem of factoring large numbers, given that to know the private key the value of the Euler phi function must be known previously

$$\phi(n)=(p-1)(q-1)$$

…in order to find…

$$d = inv\ [e,\phi(n)]$$

…but first the value of the primes p and q is needed.

Factorization time needed

Processor: $2x10^8$ instructions per second (90s).
Source: *Criptografía Digital, José Pastor. Prensas Univ. de Zaragoza, 1998*.

No bits (n)	No digits	Days	Years
60	18	$1,7 \times 10^{-8}$	-
120	36	$1,5 \times 10^{-5}$	-
256	77	1,0	-
363	109	$9,0 \times 10^{2}$	2,5
442	133	$9,4 \times 10^{4}$	$2,5 \times 10^{2}$
665	200	$3,8 \times 10^{8}$	$1,0 \times 10^{6}$

- **RSA640 Challenge (193 digits)** broken in november 2005 Bonn University.
- What in 1998 was estimated about a million years, today can be broken in 30 years with a 2,2 GHz PC.
- Challenges for larger numbers **will be solved**… so we must be careful.

Equivalent Private Key in RSA

- An equivalent private key d_P, allows decryption of a cryptogram C (previously encrypted with a public key e) without d_P needing to be the inverse of e. Any RSA system has at least one equivalent key of private key d.
- Reason: e and d are inverse in $\phi(n)$, but encryption is carried out in n.
- If $p=13$; $q=19$; $n=247$, $\phi(n)=216$ and $e=41$ is chosen, then $d=inv(41,216)=137$, which is unique.
- Encrypting $N=87$ with the public key produces $C=87^{41} \bmod 247= 159$.
- Therefore we know that $N=C^d \bmod n=159^{137} \bmod 247=87$
- But also decrypt it with $d_P=29,65,101,173,209$ and 245.

Number of equivalent Private Keys

- If $\gamma = \text{lcm}[(p-1),(q-1)]$
  and $d\gamma = e^{-1} \bmod \gamma = \text{inv}(e,\gamma)$
- Public key e will have λ equivalent keys d_i of the form:
  $d_i = d\gamma + i\,\gamma \qquad 1 < d_i < n$
  $i = 0, 1, \ldots \lambda \qquad \lambda = \lfloor (n - d\gamma)/\gamma \rfloor$
- In the previous example:
- $\gamma = \text{lcm}[(p-1),(q-1)] = \text{lcm}(12,18) = 36$
- Then: $d\gamma = \text{inv}(41, 36) = 29,$
  so $d_i = d\gamma + i\gamma = 29+i*36$
- That is $di = 29,65,101,137,173,209,245$. Note that (137) is the private key d and we verify that:
- $\lambda = \lfloor (n - d\gamma)/\gamma \rfloor = \lfloor (247 - 29)/36) \rfloor = 6{,}05 = 6$

Numbers that cannot be encrypted in RSA

- If $N^e \bmod n = N$ then N is a number that cannot be encrypted. Although key e is valid, N will be sent as plaintext ☹.
- In RSA there will be at least 9 of such numbers.
- In a critical case, all numbers of n could not be encrypted.
- To know such values, a brute force attack is needed in p and q, that is we need to check that $X^e \bmod p = X$ and $X^e \bmod q = X$ with $1 < X < n-1$ ☹.
- **Example:**
  - n = 35 (p = 5, q = 7), with $\phi(n) = 24$ and e = 11.
  - Among the possible numbers {0, 34}, the following {6, 14, 15, 20, 21, 29, 34} cannot be encrypted. Additionally, {0, 1} and n-1 (in this case 34) are always non-encryptable.

Birthday problem

- We can design an attack to the private key based on this problem.
- Question: the probability of at least two people sharing a birthday amongst a certain group of **k** people.
- Solution: In a group of 23 (or more) randomly chosen people, there is more than 50% probability (0,507) that some pair of them will have the same birthday.

$$p_{NB} = n!/(n-k)!n^k$$

n = number days of the year

Birthday problem (attack on the key)

- **Algorithm proposed by Merkle and Hellman in 1981:**
  - The attacker chooses two different random numbers **i, j** within **n**. Additionally, chooses any message or number **N**.
  - For **i=i+1** and **j=j+1** calculates $N^i \bmod n$ and $N^j \bmod n$.
  - When a ciphertext match for a give pair **(i,j)** is found, the attacker is able to find **d**.
  - An example in the following slides:
    - ✓ Let p = 7; q = 13, n = 91, e = 11, d = 59. The attacker only knows n and e. Start with N = 20 and choose i = 10 and j = 50.

Birthday paradox attack

i	C_i	j	C_i
i = 10	$C_{10} = 20^{10} \bmod 91 = 43$	j = 50	$C_{50} = 20^{50} \bmod 91 = 36$
i = 11	$C_{11} = 20^{11} \bmod 91 = 41$	j = 51	$C_{51} = 20^{51} \bmod 91 = 83$
i = 12	$C_{12} = 20^{12} \bmod 91 = 1$	j = 52	$C_{52} = 20^{52} \bmod 91 = 22$
i = 13	$C_{13} = 20^{13} \bmod 91 = 20$	j = 53	$C_{53} = 20^{53} \bmod 91 = 76$
i = 14	$C_{14} = 20^{14} \bmod 91 = 36$	j = 54	$C_{54} = 20^{54} \bmod 91 = 64$
i = 15	$C_{15} = 20^{15} \bmod 91 = 83$	j = 55	$C_{55} = 20^{55} \bmod 91 = 6$
i = 16	$C_{16} = 20^{16} \bmod 91 = 22$	j = 56	$C_{56} = 20^{56} \bmod 91 = 29$
i = 17	$C_{17} = 20^{17} \bmod 91 = 76$	j = 57	$C_{57} = 20^{57} \bmod 91 = 34$

There is a coincidence/collision with C = 36 for i and j. Note the repetition.

Using i, j and the difference between them when the coincidence is noticed (i = 14), a equation system can be established and if the attack is successful we get either the private key, an equivalent private key, or a specific private key value which only works with a given number (in this case 20). In the last case, we have a false positive.

Results of a birthday paradox attack

- First match for i = 14; j = 50. Attacker uses the public key e = 11, and calculates :

  $w = (14-50) / \gcd(11, |14-50|) = -36 / \gcd(11, 36) = -36$

- Then there will exist values of s and t so that verify the following:

  $w*s + e*t = 1 \quad \Rightarrow \quad -36*s + 11*t = 1$

- Possible solutions to the equation are:

  $w*s \bmod e = 1; \; e*t \bmod w = 1$

  $-36*s = 1 \bmod 11 \quad \Rightarrow \quad s = inv(-36, 11) = inv(8, 11) = 7$

  $11*t = 1 \bmod 36 \quad \Rightarrow \quad t = inv(11, 36) = 23$

- Value t = 23 will be an equivalent private key of d = 59. Check that $w*s + e*t = 1$ is verified and that the equivalent private keys are 11, 23, 35, 47, 71 y 83.

ElGamal

- Designed to produce digital signatures
- Extended for encryption
- Security based on the problem of solving discrete logarithms
- Choose a prime number **p** and two random numbers **g** and **x** smaller than **p**
- Calculate:
  - $y = g^x \pmod p$
- The public key is **(g,y,p)**
- The private key is **x**

ElGamal

- **Encryption of message m:**
  - $a = g^k \pmod p$, k: random integer
  - $b = M \cdot y^k \pmod p$
- **The ciphertext is (a,b)**
  - Twice the length of the plaintext!
- **Decryption (x: private key):**
  - $M = b * inv(a^x,p) \pmod p$
  - $M = M \cdot (g^x)^k \,/\, (g^k)^x \pmod p$

Example: Encryption with ElGamal

- **Alice wants to send a message to Bob**
  - **M = 10**
  - Encrypted using the modulus **p = 13**
- **Encryption:**
  - Public key of Bob:
    - ✓ **p = 13, g = 6, y = (g^x) mod p = 2**
  - Alice chooses, e.g., **k = 4** and calculates:
    - ✓ **a = (g^k) mod p = 6^4 mod 13 = 9**
    - ✓ **y^k mod p = 16 mod 13 = 3**
    - ✓ **b = M*y^k mod p = 10*3 mod 13 = 4**
- **Alice sends to Bob:**
  - **(g^k) mod p, M*y^k mod p => [9,4]**

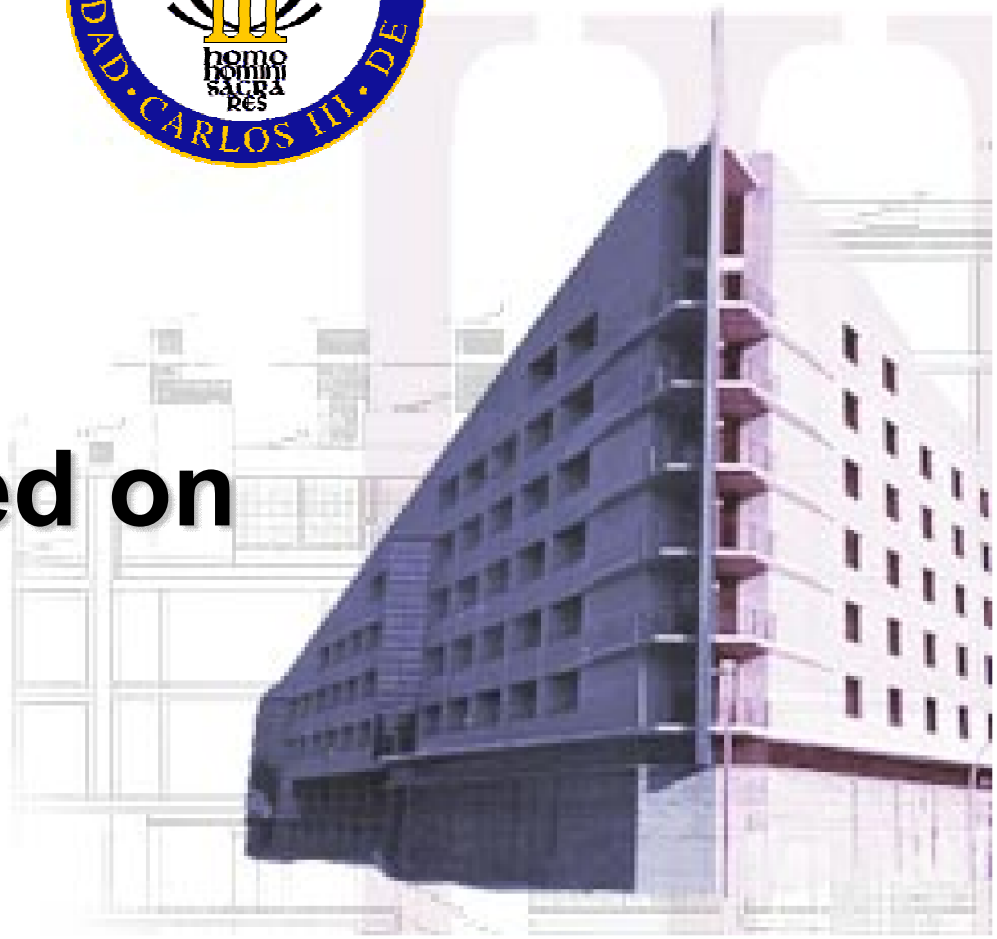
Example: Decryption with ElGamal

- **Private key of Bob: $x = 5$**
  - So: $y = (g^x) \bmod p = (6^5) \bmod 13 = 2$
- **Bob receives: [a,b]**
  - $[(g^k) \bmod p,\ M*y^k \bmod p\ ] = [9,4]$
- **Bob computes:**
  - $a^x = (g^k)^x \bmod p = 9^5 \bmod 13 = 3$
  - $b * inv(a^x,p) = [M*(g^x)^k] * inv[(g^k)^x, p] =$
    $= 4 * inv\ (3,\ 13) = 4 * 9$
- **Finally, to recover the plaintext message:**
  - $M = 4 * 9 \bmod 13 = 10$
  - Note that Bob recovers the message **without knowing** Alice's random number

Summary of public key systems based on exponentiation

- Sender and receiver generate a pair of keys, public and private, both related by a **one-way trap function**.
- Sender and receiver use **different keys** for encryption and decryption operations.
- The system security is associated with the solution of a **difficult and time-consuming** mathematical problem.
- Allow **digital signature**.
- They are **very slow**.
- They are useful for encryption of **short messages** (like session keys) or to sign hashes (message digests).

Algorithms based on Elliptic Curves



ECC

- Elliptic curve-based cryptography (ECC) was introduced by **Victor Miller** and **Neal Koblitz** in 1985.
- Its first **advantage** over exponentiation-based is that it requires **shorter keys** in order to achive similar security (computational cost to break the system)
  - Shorter keys → faster computation time and smaller storage requirements
  - Useful for small devices

Elliptic Curves

- **Equation defining elliptic curves:**
  - $y^2 \; [\; + \; x \cdot y \;] \; = \; x^3 \; + \; a \cdot x^2 \; + \; b$
    - ✓ $[\; + \; x \cdot y \;]$: an optional component of the equation
    - ✓ a, b: constant
    - ✓ $a, \; b, \; x, \; y$: real numbers
- **For application to encryption:**
  - $x, \; y, \; a, \; b$: belong to a finite field
    - ✓ Such as provided by modular arithmetic
- **Using Finite Fields (Galois Fields (GF))**
- **We will not cover GF but a particular case of GF are the ones generated by modulus n operations, n being prime (GF(n))**

Number of points in an elliptic curve

- **Hasse's Theorem:**
  - For a given elliptic curve defined over GF(q), the number of points on the curve will be:

$$q + 1 - t$$

  - Where $|t| \leq 2\sqrt{q}$
    - ✓ ***t*** **is *Frobenius' trace***
    - ✓ ***O*** **is included**

Operations over Elliptic Curves

- **Addition** of points and **product** of a point by an integer are defined
- Addition was explained in the previous slide (in cryptography we will use the same expressions applied to finite bodies).
- Define also the multiplication of a point **P** on the curve with one integer **k**:
  - **2P = P+P**
  - **3P = P+P+P** ...
  - Analogous to 'exponentiation': $x^2=x*x$

Order of a point and order of the curve

- **The order of a point P on the curve is the integer n such that:**
  - nP=O
  - (n+1)P=P

Security of Elliptic Curves

- Security comes from the fact that there is an efficient way of calculating:
  - Q=kP …
- ..if we know **k** and we know **P** (using a technique similar to the method of rapid exponentiation seen in class)
- AND however, if we know **P** and **Q** there is no efficient algorithm to find **k**. We use this to protect **k** when sent to the receiver.

Complexity of Elliptic Curve Cryptography

- **Complexity as known of today:**
  - **If:** the order of **F** is an n-bit prime number
  - **Then:** Computing **k** knowing **k*F** and **F** takes roughly $2^{n/2}$ operations
- **Example:**
  - Order of **F** is a 240-bit prime number
  - Brute-force attack to get **k** would take 2^{120} operations
  - Assume a machine with 1 million operations per second
  - Brute-force attack would last:
    - ✓ 2^{100} seconds or $2^{75} \approx 10^{23}$ years...

ElGamal over ECC

- **Alice and Bob choose:**
  - A finite Galois Field GF
  - An elliptic curve **E**
  - A fixed point **F∈E**
- **Alice chooses:**
  - A random number **a**
    - ✓ Publishes the point **aF∈E** as public key
- **To send a "message" M to Alice, Bob must:**
  - Calculate **M** (point on **E** whose **x**-component is **m**)
  - Chooses a random number **k** and sends **kF** and **M+k(aF)** to Alice
- **To read the message, Alice calculates:**
  - **M+k(aF) – a(kF) = M**

Comparison: ElGamal over ECC vs. ElGamal Based on Exponentiation

- **Comparing ECC and Exponentiation:**
  - Decryption in ECC:
    - ✓ **M+k(aF) – a(kF)**
  - Decryption in Exponentiation:
    - ✓ $M(g^a)^k / (g^k)^a = M(g^a)^k * inv((g^k)^a)$
- **General way to convert a method based on exponentiation to one based on ECC:**
  - Exponentiation → Multiplication
    - The base of the exponentiation → point on the curve
    - Exponent → into a scalar
  - Products → Sum
  - Division → Subtraction

Elliptic Curve Cryptography: Example Diffie-Hellman

- Publish the definition of an elliptic curve and one point **P** on that curve (the *fixed point*)
- Each user creates a private key by choosing a random number **k**
- The public key: multiplication of **k** with **P**
- Example:
  - Alice chooses: $A_p = k_1$ and $A_P = k_1 * P$
  - Carol chooses: $C_p = k_2$ and $C_P = k_2 * P$

Elliptic Curve Cryptography: Example Diffie-Hellman

- **Both can gain a shared secret using**
  - ✓ Alice: $A_p * C_P = k_1 * k_2 * P$
  - ✓ Carol: $C_p * A_P = k_1 * k_2 * P$
- **The security of this scheme is based on a high complexity to calculate k_1 and k_2 from the public keys A_P and C_P**