

Public key cryptography.

ElGamal, hints on implementation.

Marco Bodrato

Centro "Vito Volterra" – University of Roma "Tor Vergata" – Italy

Tokyo University of Science - March 18th, 2008

A note!

The slides I showed during ICQBIC are available on the web:

<http://bodrato.it/presentazioni/#ICQBIC08>,

Released with a Creative Commons BY-NC-SA licence. 

\LaTeX source is available too,
and it's signed with my ElGamal GPG/PGP key.

Recall on ElGamal key generation

Key generation $\mathcal{K} \rightarrow (K_e, K_d)$

Choose a **prime** p , a **generator** g so that $\langle g \rangle = \mathbb{Z}_p^*$, and a random number a , so that it is **coprime to $p - 1$** ;

compute $k \equiv g^a \pmod{p}$.

Return the public $K_e = (p, g, k)$, and the secret $K_d = (p, g, a)$.

To fully understand all the steps needed to implement ElGamal, we need to understand all the terms in the definitions.

Definitions: primality

prime A **prime number** (or a **prime**) is a natural number (greater than one) which has exactly two distinct natural number divisors: 1 and itself.

Examples: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, ...

relatively Two numbers a, b are relatively primes iff $\text{GCD}(a, b) = 1$

Examples: $\text{GCD}(64, 65) = 1$, $\text{GCD}(78, 65) = 13$

\mathbb{Z}_n^* The multiplicative group of invertibles. Is the set of numbers relatively prime with n . When n is prime, it is the set of x s $0 < x < n$, it's cyclic: it has a generator ($\phi(n - 1)$ of them).

Examples: $\mathbb{Z}_{12}^* = \{1, 5, 7, 11\}$, $\mathbb{Z}_{11}^* = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$

Definitions: primitive root

p is prime $\Rightarrow \mathbb{Z}_p^*$ is a cyclic group **and** \mathbb{Z}_p is a field

The operation internally defined on \mathbb{Z}_p^* is multiplication.

By the sentence “is a cyclic group” we mean there is a generator.
An element g is a generator iff

- $\forall x \in \mathbb{Z}_p^*, \exists n \in \mathbb{N} : g^n \equiv x \pmod{p}$
- $g^n \equiv 1 \Leftrightarrow p - 1 | n$

There are $\phi(p - 1)$ generators

If g is a generator, g^n is a generator iff $\text{GCD}(n, p - 1) = 1$.

Definitions: generator

Example: 2 generates \mathbb{Z}_{11}^*

$$\triangleright 2^1 = 2 \equiv 2 \pmod{11} \qquad 2^6 = 2 \cdot 2^5 \equiv 9 \pmod{11}$$

$$2^2 = 2 \cdot 2^1 \equiv 4 \pmod{11} \qquad \triangleright 2^7 = 2 \cdot 2^6 \equiv 7 \pmod{11}$$

$$\triangleright 2^3 = 2 \cdot 2^2 \equiv 8 \pmod{11} \qquad 2^8 = 2 \cdot 2^7 \equiv 3 \pmod{11}$$

$$2^4 = 2 \cdot 2^3 \equiv 5 \pmod{11} \qquad \triangleright 2^9 = 2 \cdot 2^8 \equiv 6 \pmod{11}$$

$$2^5 = 2 \cdot 2^4 \equiv 10 \pmod{11} \qquad 2^{10} = 2 \cdot 2^9 \equiv 1 \pmod{11}$$

\triangleright tags the $\phi(10) = 4$ generators.

Exercise:

Try to find, by hand, both generators of \mathbb{Z}_7^* .

Compute the Greatest Common Divisor ($\text{GCD} \in P$)

Euclidean algorithm 1

$$\text{GCD}(a, a) = a$$

$$\text{GCD}(a, b) = \text{GCD}(\max(a, b) - \min(a, b), \min(a, b))$$

Euclidean algorithm 2

To compute $\text{GCD}(a, b)$, first find q, r so that

$$a = qb + r, 0 \leq r < b$$

$$r \neq 0 \Rightarrow \text{GCD}(a, b) = \text{GCD}(b, r)$$

$$r = 0 \Rightarrow \text{GCD}(a, b) = b$$

Extended Euclidean algorithm

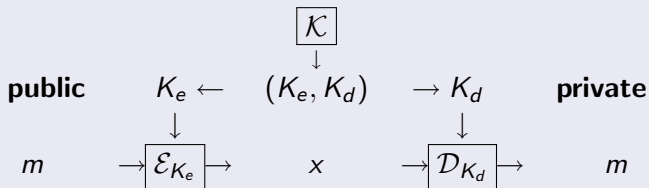
▶ look at algorithm

Keep track of operations, to find $x, y \in \mathbb{Z}$ so that Bézout's identity is verified: $\text{GCD}(a, b) = x \cdot a + y \cdot b$

Recall on public key encryption

Asymmetric cryptosystems typically consist of a triplet of polynomial-time algorithms:

- \mathcal{K} , the algorithm for **key generation**, generates couples (K_e, K_d)
- \mathcal{E} , the algorithm for **encryption**, taking the key K_e and the clear message m , generates the ciphered message x .
- \mathcal{D} , the algorithm for **decryption**, taking the key K_d and the ciphered message x , recovers the clear message m .



ElGamal: key generation ($\mathcal{K} \in P$)

Key generation $\mathcal{K} \rightarrow (K_e, K_d)$

Choose a prime p , a generator g , and a random number a , so that

$$\text{GCD}(a, p - 1) = 1$$

compute $k \equiv g^a \pmod{p}$.

Return the public $K_e = (p, g, k)$, and the secret $K_d = (p, g, a)$.

- Generation of a prime is in P .(unessential) [Primes is in P, AKS]
- Finding a generator is in P (LV).(unessential) [▶ see appendices](#)
- Finding a random number, coprime to p is in P (LV). [GCD]
- Modular exponentiation is in P . [▶ we will see](#)

Encryption $\mathcal{E}_{K_e} : m \rightarrow (w, x)$, uses $K_e = (p, g, k)$

Take a message $m \in \mathbb{Z}_p$, and a **random** r ;
 compute $w = g^r \pmod{p}$ and $x \equiv m + k^r \pmod{p}$,
 send the couple (w, x) .

The couple (w, x) can be sent on the net.

Decryption $\mathcal{D}_{K_d} : (w, x) \rightarrow m$, uses $K_d = (p, g, a)$

Recover the message by

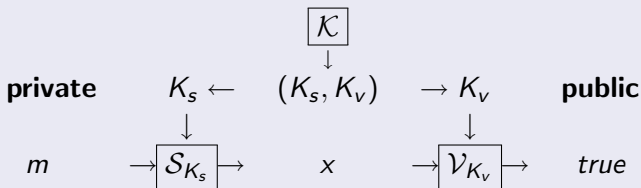
$$x - w^a \equiv m + k^r - (g^r)^a \equiv m + (g^a)^r - (g^a)^r \equiv m \pmod{p}.$$

For both procedures we recycle pieces of code, those used for key generation. Everything can be computed in polynomial time.

Recall on public key signature

Asymmetric digital signature systems typically consist of a triplet of polynomial-time algorithms:

- \mathcal{K} , the algorithm for **key generation**, generates couples (K_s, K_v)
- \mathcal{S} , the algorithm for **signing**, taking the key K_s and the clear message m , generates the signed message x .
- \mathcal{V} , the algorithm for **verifying**, taking the key K_v and the signed message x , answers **true** or **false**.



ElGamal: signature

ElGamal signature uses the same keys: $K_s = K_d$ and $K_v = K_e$.

Signature \mathcal{S}_{K_d} , uses $K_s = K_d = (p, g, a)$

Take a message $m \in \mathbb{Z}_p$, and a **random** number $r \in \mathbb{Z}_{p-1}^*$.

Compute $x = g^r$ and solve in y the equation

$$m \equiv xa + ry \pmod{p-1}.$$

Publish the signed message (m, x, y) .

The only new procedure we need is modular inversion. [▶ look at algorithm](#)

Verify \mathcal{V}_{K_e} , uses $K_v = K_e = (p, g, k)$

Verify if $k^x x^y \equiv g^m \pmod{p}$, return **true** if true, **false** otherwise.

Verify gives correct results because of Fermat's Little Theorem.

$$g^{ax} g^{ry} \equiv g^{ax+ry} \equiv g^m \pmod{p} \Leftrightarrow ax + ry \equiv m \pmod{p-1}.$$

Exercise

Implement ElGamal protocol (encryption-decryption and/or signature-verify), for small numbers.

Use one of this three primes:

- $p = 1073741827$, or
- $p = 2147483658$, or
- $p = 9223372036854775907$

(depending on language/architecture).

Use the generator $g = 2$.

Thank you!

Questions?

Thank you very much for your kind attention!

Questions?

This presentation will be available on the web:
<http://bodrato.it/presentazioni/#ICQBIC08-1>,

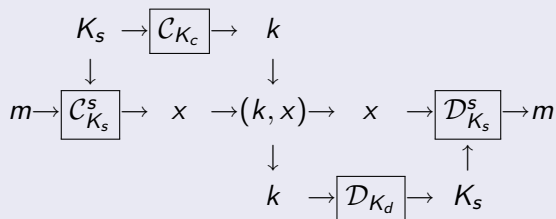
Released with a Creative Commons BY-NC-SA licence.



How public-key protocols are used in practice

Public-key protocols are **slow**, so they are always used in conjunction with a *symmetric* one.

Hybrid protocol



A session key K_s is randomly generated and used once for the symmetric algorithm, it is also attached, ciphered by the asymmetric algorithm, to the encrypted message.

Finding a primitive root

Fermat's Little Theorem

$$\forall a, p \in \mathbb{N}, \text{GCD}(a, p) = 1 \Rightarrow a^{p-1} \equiv 1 \pmod{p}$$

The order of $a \in \mathbb{Z}_p^*$ divides $p - 1$. Once we have the factorization

$$p - 1 = \prod_{i=1}^n p_i^{\alpha_i}$$

We just have to check whether

$$\forall i \in \{1, \dots, n\} a^{(p-1)/p_i} \not\equiv 1 \pmod{p}$$

Extended Euclidean algorithm

Semigraphic representation of the Extended Euclidean algorithm.

a	b	d	<i>expression</i>
1	0	a	
0	1	b	$r_0 = a - qb$
1	$-q$	r_0	$r_1 = b - q_0r_0$
\vdots	\vdots	\vdots	\vdots
y_{n-1}	x_{n-1}	r_{n-1}	\vdots
y_n	x_n	r_n	$r_{n+1} = r_{n-1} - q_nr_n$
$y_{n-1} - q_ny_n$	$x_{n-1} - q_nx_n$	r_{n+1}	\vdots
y	x	$\text{GCD}(a, b)$	$\text{GCD}(a, b) = ya + xb$

If $\text{GCD}(a, b) = 1$, then $x \equiv b^{-1} \pmod{a}$, and $y \equiv a^{-1} \pmod{b}$.