A short tutorial on classical cryptography.

From ancient methods, to public key.

Marco Bodrato

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

ICQBIC - TUS - March 16th, 2008

1. From symmetric to asymmetric
   - What do we mean by cryptography
   - Symmetric cryptography
   - How do we share a secret?
2. Towards public key
   - DH Key Agreement Protocol
   - Can the protocol be attacked?
   - Pros and Cons
3. Public key cryptosystems
   - Rivest, Shamir, Adleman: a first paradigm
   - ElGamal: a different approach
   - Comparisons

What do we mean by cryptography?

Original meaning

It's an ancient world: *hidden writing*.

It concerns two persons, sharing a *secret way of writing*, so that nobody else can understand.

. . . but nowadays. . . digital world requires digital security

We need all the *security features* we have in the real world.

What do we mean by cryptography?

Original meaning

It's an ancient world: *hidden writing*.
It concerns two persons, sharing a *secret way of writing*, so that nobody else can understand.

. . . but nowadays. . . digital world requires digital security

We need all the *security features* we have in the real world.

- Private communication
- Digital signature
- Identity proof
- Electronic money
- Anonymity
- Internet vote. . .

. . . and much more!

The *ancient* way

Symmetric cryptography...

The simplest model

clear	Encrypt	ciphered	Decrypt	clear
m	$\rightarrow \boxed{\mathcal{E}} \rightarrow$	x	$\rightarrow \boxed{\mathcal{D}} \rightarrow$	m

The *ancient* way

Symmetric cryptography...

The simplest model

clear	Encrypt	ciphered	Decrypt	clear
m	$\rightarrow \boxed{\mathcal{E}} \rightarrow$	x	$\rightarrow \boxed{\mathcal{D}} \rightarrow$	m

A more general (keyed) model

$$m \rightarrow \boxed{\mathcal{E}_K^s} \rightarrow \quad x \quad \rightarrow \boxed{\mathcal{D}_K^s} \rightarrow \quad m$$

$\mathcal{E}$ and $\mathcal{D}$ are the algorithms to Encrypt and Decrypt, respectively. This scheme is called *symmetric* because the key K is the same for both algorithms. If I know how to encrypt, I know how to decrypt.

The *ancient* way

. . . is still in wide use today!

It may be easier to consider functions within bit-strings

$$\mathcal{E}^s : \mathbb{K} \times \{0,1\}^* \to \{0,1\}^*, \qquad \mathcal{E}^s(K,m) = x$$
$$\mathcal{D}^s : \mathbb{K} \times \{0,1\}^* \to \{0,1\}^*, \qquad \mathcal{D}^s(K,x) = m$$
$$\forall k \in \mathbb{K},\ \mathcal{E}^s|_{K=k} \circ \mathcal{D}^s|_{K=k} = Id \quad \mathcal{D}^s(K, \mathcal{E}^s(K,m)) = m$$

There are many famous and standard algorithms in use today:

- Block: 3DES, AES, IDEA, Blowfish. . .
- Stream: RC4, CBC-AES, Snow2.0. . .

The key of encryption strength is . . . the *Key*.

The best way to break a symmetric encryption should be the *brute force* attack.
We can try all the possible keys $k \in \mathbb{K}$, and decrypt $m_k = \mathcal{D}_k^s(x)$ till we find a meaningful message.
If the key is longer than the message m, brute force fails.
Strong encryption needs strong keys.

Do we need to transmit the key?

If we are far from one another, how do we communicate the key? Will we encrypt the key itself? With another key. . .

There is the need of a Key Agreement Protocol.

W. Diffie and M. Hellman (1976)

How to share a secret, in public: the first Key Agreement Protocol

The protocol

Two participants choose a prime p, and a suitable number $g < p$.

$\mathcal{A}$ chooses a secret number a and sends $g^a \pmod p$ to $\mathcal{B}$.
$\mathcal{B}$ chooses a secret number b and sends $g^b \pmod p$ to $\mathcal{A}$.

$\mathcal{A}$ computes $(g^b)^a \pmod p$, $\mathcal{B}$ computes $(g^a)^b \pmod p$.

Both share $g^{a \cdot b} \pmod p$.

An attacker knows

All transmitted data: the prime p and $g, g^a, g^b \in \mathbb{Z}_p$.
But she does not know either a or b, and *cannot* compute g^{ab}.

W. Diffie and M. Hellman (1976)

A numerical example

Participants publicly choose $p = 1234567891$, and generator $g = 3$.

$\mathcal{A}$ chooses $a = 104218932$, and publishes $g^a \equiv 915345118 \pmod{p}$.

$\mathcal{B}$ chooses $b = 482273880$, and publishes $g^b \equiv 346320197 \pmod{p}$.

Now both have g^{ab}
$915345118^{482273880} \equiv 346320197^{104218932} \equiv 1090646172 \pmod{p}$

Is DH breakable?

Any observer knows...

$p = 1234567891,\ g = 3,\ g^a \equiv 915345118,\ g^b \equiv 346320197,$
can she compute g^{ab}?

There isn't any theoretical proof, obstruction is computational only.

Is DH breakable?

Any observer knows...

$p = 1234567891,\ g = 3,\ g^a \equiv 915345118,\ g^b \equiv 346320197,$
can she compute g^{ab}?

There isn't any theoretical proof, obstruction is computational only.

The attacker knows all the informations she needs.
A number $\tilde{a}$ so that $g^{\tilde{a}} \equiv g^a \pmod{p}$ **can** be found (guessed, or by exhaustive search $\tilde{a} = 1, 2, 3 \ldots$).
Then $(g^b)^{\tilde{a}} = (g^{\tilde{a}})^b \equiv g^{ab} \pmod{p}$ is the shared secret.

There isn't any (known) fast classical algorithm

Discrete logarithm is commonly believed hard.

Advantages and disadvantages

What we gain

We can securely *communicate* keys, without meeting in person.

What we loose

Protocols like DH usually require far slower computations than symmetric algorithms.
Interaction is needed.

What else we need

Authentication (against *Man-in-the-middle* attacks).
Off-line, or non-interactive protocols.
...

Public key cryptosystems definitions

The model again

clear	Encrypt	ciphered	Decrypt	clear
m	$\rightarrow \boxed{\mathcal{E}_{K_e}} \rightarrow$	x	$\rightarrow \boxed{\mathcal{D}_{K_d}} \rightarrow$	m

Public key cryptosystems definitions

The model again

clear	Encrypt	ciphered	Decrypt	clear
m	$\rightarrow \boxed{\mathcal{E}_{K_e}} \rightarrow$	x	$\rightarrow \boxed{\mathcal{D}_{K_d}} \rightarrow$	m

Public-key systems are called *asymmetric* because the two keys $K_e \neq K_d$ are different. Encryption key can be made public.
Not any more "one key for each couple", but "one public key for each person".

Asymmetric encryption systems are triplets

$\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$: three algorithms.
$\mathcal{K}$ generates key couples (K_e, K_d) so that the diagram above is correct $\forall m$.

RSA: encryption

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

Choose a numbers n product of two primes $n = p \cdot q$, and two numbers e, d so that $e \cdot d \equiv 1 \pmod{(p-1) \cdot (q-1)}$.
Return $K_e = (n, e)$, $K_d = (n, d)$.

RSA: encryption

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

Choose a numbers n product of two primes $n = p \cdot q$, and two numbers e, d so that $e \cdot d \equiv 1 \pmod{(p-1)\cdot(q-1)}$.
Return $K_e = (n, e)$, $K_d = (n, d)$.

Encryption $\mathcal{E}_{K_e}$

Take a message $m \in \mathbb{Z}_n$, compute $x \equiv m^e \pmod{n}$.

Decryption $\mathcal{D}_{K_d} \equiv \mathcal{E}_{K_e}$

Recover the message by $x^d \equiv m^{ed} \equiv m \pmod{n}$.

The algorithm works because of the Chinese Remainder Theorem and Fermat's little theorem.

RSA: signature

Digital signature systems are triplets

$\Sigma = (\mathcal{K}, \mathcal{S}, \mathcal{V})$, three algorithms.

RSA uses the same keys, and reverts encryption.

RSA: signature

Digital signature systems are triplets

$\Sigma = (\mathcal{K}, \mathcal{S}, \mathcal{V})$, three algorithms.

RSA uses the same keys, and reverts encryption.

Signature $\mathcal{S}_{K_d} \equiv \mathcal{D}_{K_d}$

Take a message $m \in \mathbb{Z}_n$, compute signature $s \equiv m^d \pmod{n}$.

Verify $\mathcal{V}_{K_e} \equiv \mathcal{E}_{K_e}$

Recover the message by $s^e \equiv m \pmod{n}$.

Only the owner of the key K_d could "decrypt" the message m to the signature s.

ElGamal: encryption

ElGamal encryption trivially uses DH protocol, a half static version.

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

Choose a prime p, a generator g and a random number a; compute $k = g^a \pmod{p}$.
Return $K_e = (p, g, k)$, $K_d = (p, g, a)$.

Encryption $\mathcal{E}_{K_e}$

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

Decryption $\mathcal{D}_{K_d}$

Recover the message by $x - (g^r)^a \equiv x - (g^a)^r \equiv m \pmod{p}$.

ElGamal: signature

ElGamal signature uses the same key, but is a little bit more complicated.

Signature $\mathcal{S}_{K_d}$

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).

Verify $\mathcal{V}_{K_e}$

Verify if $k^x x^y \equiv g^m \pmod{p}$. True if
$g^{ax} g^{ry} \equiv g^{ax+ry} \equiv g^m \pmod{p} \Leftrightarrow ax + ry \equiv m \pmod{p-1}$.

We are using Fermat's little theorem again.

Main differences between RSA and ElGamal

RSA	ElGamal
deterministic encryption no message expansion	randomised encryption 2× message expansion can be generalised
deterministic signature no message expansion reverses encryption	randomised signature 3× message expansion zero knowledge proof
based on factorisation hardness	based on Discrete Log hardness

Thank you!

Questions?

Thank you very much for your kind attention!

Questions?

This presentation is available on the web:
http://bodrato.it/presentazioni/#ICQBIC08,
Relesed with a CreativeCommons BY-NC-SA licence.