Introduction to cryptography

Jorge Estrada Sarlabous

Introduction to cryptography.

J. Estrada Sarlabous

January 9, 2001

<<... La inteligencia se encuentra en muchos lugares
Intelligence is found in many places
L'intelligence se trouve dans beaucoup endroits
Intelligenz findet sich an vielen Plaetzen ...>>

- Jorgito

1 Introduction

Cryptography has a long a fascinating history. The most complete non-technical account is D. Kahn's book, The Codebreakers, which starts with the ancient Egyptians, 4000 years ago and ends with the XX century and its crucial role in the World Wars. Cryptography applications were mainly associated to the military and the diplomatic service, where it was used as a tool to protect secrets.

Until some years ago, sophisticated secure systems were mainly restricted to the military world. ENIGMA, was a famous machine used by the Germans during WW II, M-209 was its American counterpart. The English were not only able to break the ENIGMA code, but also successful in keeping the Germans unaware of this fact during the war.

With the proliferation of computers and communications systems in the 1960s arised a demand from the private, civil sector for means to protect information in digital form and to provide security services:

- Phone calls via satellite are easy to listen in. Similarly, unauthorized viewers arise with respect to payed TV

- In multiuser computer systems, each user must be identified. Today passwords are used, in the future, smart cards should provide higher security

- Electronic banking requieres a susbtitute for handwritten signature

- Data authentification may be considered as a means of defense against computer viruses

The works of Feistel at IBM in the early 1970s and their adoption in 1977 as an U.S. Federal Information Processing Standard for enciphering unclassified information the Data Encryption Standard (DES), is the most well known cryptosystem in history (it remains still the standard for securing electronic commerce in many places).

The famous paper New directions in Cryptography of Diffie and Hellman in 1976 began a revolution in cryptology, with the introduction of the public key cryptosystems. In 1978, Rivest, Shamir and Adleman discovered the PKC , which is nowdays refered to as RSA.

One of the most significant contributions provided by PKC is the digital signature. In 1991, the first international standard for digital signatures (ISO/IEC 9796) was adopted (it is based on RSA PKC). In 1994, the U.S. Government adopted the Digital Signature Standard (based on ElGamal PKC).

Aren't there any other methods to achieve security? Besides bodygards, special papers and inks, water marks, silver wires, etc., Mathematics provides the theoretical justification of the strength of an algorithm or a protocol. Once security is mathematically proved, there is no doubt that an algorithm is secure and there is no need on (often contradictory) experts opinions!

2 Some information security objectives

Privacy:	keeping information secret from unauthorized entities
Data integrity:	ensuring information unaltered
Entity authentication:	corroboration of the identity of an entity
Message authentication:	corroborating of source of information
	(also known as data origin authentication)
Signature:	a means to bind information to an entity
Authorization:	conveyance (to another entity) of
	official sanction to do or to be something
Validation:	a means to provide timeliness of authorization
	to use or to manipulate information
Access control:	restricting access to resources to privileged entities
Certification:	endorsement of information by a trusted entity
Timestamping:	recording the time of creation of information
Witnessing:	verifying the existence or creation of information
	by an entity other than the creator
Receipt:	acknowledgement that information has been received
Confirmation:	acknowledgement that services have been received
Ownership:	a means to provide an entity with the legal right
	to transfer or use a resource to others
Anonymity:	concealing the identity of an entity
Non-repudiation:	preventing the denial of previous commitments
Revocation:	retraction of certification of authorization

2.1 Crytographic objectives

Confidentiality (Secrecy):	service to keep the information from all but those
	authorized to have it
Data integrity:	service to address unauthorized alteration of data
Authentication:	service related to identification of entities
	and/or of information
Non-repudiation:	service to prevent an entity from denying previous
	commitment

2.2 Main problems of cryptography

- Enciphering

- Authentication

- Digital Signature

- Key Establishment Protocols

- Key Management

2.3 Crytographic primitives

Basic cryptographic tools to provide information security, such as enciphering schemes, hash functions, generators of pseudorandom sequences and digital signature schemes.

They should be evaluated with respect to several criteria:

- Level of security (work factor): Upper bound on the amount of work needed to defeat the objective

- Functionality: Primitives may be combined to achieve several information security objectives. Which are most effective for a given objective?

- Methods of operation: When applied with different ways and with different outputs, primitives have different characteristics, hence depending on the context, one primitive could provide different functionality

- Performance: Efficiency of a primitive in a particular mode of operation (for instance, rating of the number of bits per second which an enciphering scheme can encipher)

- Ease of implementation: Difficulty of realizing the primitive in a practical situation (complexity of implementing a primitive in a software or a hardware environment).

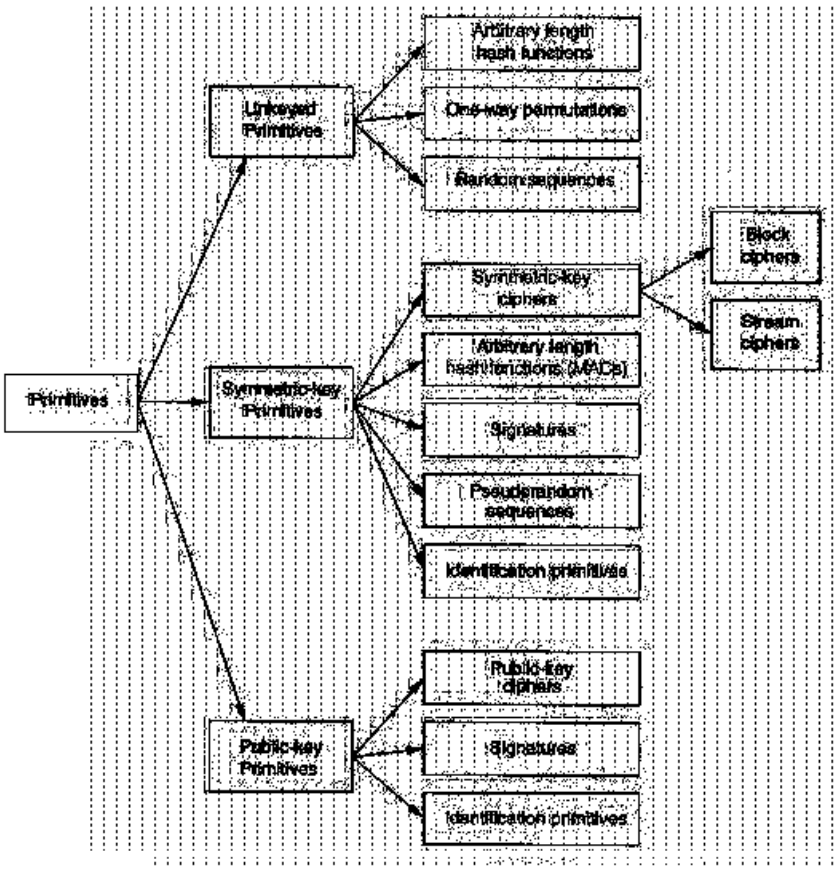

Figure 1: Security Primitives

3 Enciphering

3.1 Terminology.

A	finite set, called the alphabet of definition
M	the set of strings of symbols from A,
	called message space
$m \in M$	element of M, called plaintext
C	the set of strings of symbols from an
	alphabet of definition $A^{'}$
$c \in C$	element of $A^{'}$, called ciphertext
K	a set, called the key space
$k \in K$	element of K, called key

Each element $e \in K$ determines a 1-1 transformation $E_e : M \longrightarrow C$, which is called enciphering function. The process of applying E_e to $m \in M$ is referred to as enciphering m

For each $d \in K$, D_d denotes a 1-1 transformation $D_d : C \longrightarrow M$, which is called deciphering function. The process of applying D_d to $c \in C$ is referred to as deciphering c

Definition 3.1 *An enciphering scheme consists of a set $\{E_e : e \in K\}$ of enciphering transformations and a set $\{D_d : d \in K\}$ of deciphering transformations, such that for each $e \in K$, there is a unique key $d \in K$, with $D_d(E_e(m)) = m$ for all plaintexts $m \in M$*

Example 3.1 *For a fixed N-letter alphabet (with numerical equivalence also fixed), consider the following affine family of cryptosystems:*

For each $a \in (\mathbb{Z}/N\mathbb{Z})^*$ and $b \in \mathbb{Z}/N\mathbb{Z}$, the key is $e = (a, b)$ and the enciphering function is the map from $M := \mathbb{Z}/N\mathbb{Z}$ to $C := \mathbb{Z}/N\mathbb{Z}$ defined by $E_e(m) = am + b \ (mod\ N)$, for any $m \in M$.

For each $e = (a, b) \in K$, the unique key $d \in K$ such that $D_d = E_e^{-1}$ is $d = (a^{-1}, -a^{-1}b)$, where a^{-1} denotes the multiplicative inverse of the element a in $\mathbb{Z}/N\mathbb{Z}$.

3.2 Symmetric key enciphering

<<...Five or six weeks later, she [Mme. d'Urfé] asked me if I had deciphered the manuscript which had the transmutation procedure. I told her I had.
"Without the key, sir, excuse me if I believe the thing impossible"
"Do you wish me to name your key, madame?"
"If you please"
I then told her the key-word, which belonged to no language, and I saw her surprise. She told me that it was impossible, for she believed herself the only possessor of that word which she kept in her memory and which she had never written down.
I could had told her the truth - that the same calculation which had served me for deciphering the manuscript had enabled me to learn the word- but on a caprice it struck me to tell her that a genie had revealed it to me. This false disclosure fettered Mme. d'Urfé to me. That day I became the master of her soul, and I abused my power...>>

- Casanova, 1757, quoted in D. Kahn's The Codebreakers

Definition 3.2 *Consider an enciphering scheme consisting of a key space K, a set $\{E_e : e \in K\}$ of enciphering transformations and a set $\{D_d : d \in K\}$ of deciphering transformations. The enciphering scheme is called Symmetric key enciphering scheme, if for each associated enciphering/deciphering pair (e, d), it is computationally "easy" to determine d knowing only e and to determine e knowing only d*

Since usually $e = d$, the term symmetric is appropriate.

The are two classes of symmetric key enciphering schemes: the block ciphers and the stream ciphers.

Definition 3.3 *A block cipher is an enciphering scheme which breaks up the plaintext to be transmited into strings (called blocks) of a fixed length t (blocklength) over an alphabet A and enciphers one block at a time.*

Definition 3.4 *(Monoalphabetic enciphering scheme) Suppose the ciphertext and plaintext are the same. Let $m = m_1 m_2 m_3 ....$ be a plaintext message consisting of juxtaposed characters $m_i \in A$, where A is some fixed character alphabet. A monoalphabetic enciphering scheme uses a permutation e over A, with enchipering mapping $E_e(m) = e(m_1)e(m_2)e(m_3)....$ (concatenation of the characters $e(m_i)$).*

Example 3.2 *(Caesar enciphering scheme) If* $|A| = N$ *and* m_i *is associated with the integer value* i, $0 \leq i \leq N-1$, *then the key is* $k \in [0, N-1]$, *the enciphering mapping* e *is defined as* $c_i = e(m_i) = m_i + k \ (mod N)$ *and the deciphering mapping is defined as* $d(c_i) = c_i - k \ (mod N)$. *According to folklore, Caesar used the key* $k = 3$.

Example 3.3 *(Affine enciphering scheme) For a fixed* N*-letter alphabet (with numerical equivalence also fixed), consider the following affine family of cryptosystems:*

For each $a \in (\mathbb{Z}/N\mathbb{Z})^*$ *and* $b \in \mathbb{Z}/N\mathbb{Z}$, *the key is* $e = (a, b)$ *and the enciphering function is the map from* $M := \mathbb{Z}/N\mathbb{Z}$ *to* $C := \mathbb{Z}/N\mathbb{Z}$ *defined by* $E_e(m) = am + b \ (mod\ N)$, *for any* $m \in M$.

For each $e = (a, b) \in K$, *the unique key* $d \in K$ *such that* $D_d = E_e^{-1}$ *is* $d = (a^{-1}, -a^{-1}b)$, *where* a^{-1} *denotes the multiplicative inverse of the element* a *in* $\mathbb{Z}/N\mathbb{Z}$.

The statistical study of the frecuence of occuring symbols of a given alphabet for messages written in a fixed language may make feasible to break monoalphabetic enciphering schemes, if suffiently large messages are analysed by a cryptoanalist. It gave rise to the introduction of another family of enciphering schemes.

Definition 3.5 *(Polygram substitution enciphering scheme) Groups of* t *characters of an alphabet* A, *with* $|A| = N$ *are substituted by groups of* t *characters of* A. *The key may be any of the* N^t *possible replacements of groups of* t *characters of* A *by groups of* t *characters of* A.

Example 3.4 *(Hill encihering scheme) The key is an invertible (in* $\mathbb{Z}/N\mathbb{Z}$ *)* $t \times t$ *matrix* $G = (a_{ij})$. *Given a plaintext* $m = m_1 m_2 m_3 ... m_t$, *the enciphering mapping is* $c = c_1 c_2 ... c_t$, *with* $c_i = \sum_{j=1}^{t} a_{ij} m_j$, *for* $i = 1, .., t$

3.3 DES

DES (Data Encryption Standard) is a Feistel enciphering scheme which processes plaintext blocks of $n = 64$ bits, producing $64-$bits ciphertext blocks using a secret ket K of $56-$bits (more precisely, the input key is a $64-$bits key, $8-$bits of which (bits 8,16,...,64) may be used as parity bits.

An overview of the algorithm is the following. Enciphering proceeds in 16 stages, called rounds. From the input key K, sixteen $48-$bits subkeys K_i are generated, one for each round. Within each round, 8 fixed $6-to-4$ bit substitution mappings (called $S-$boxes) S_i, collectivelly denoted S are used. The $64-$bits plaintext is divided into $32-$bits halves L_0 and R_0. Each round

takes $32-$bit inputs L_{i-1} and R_{i-1} from the previous round and produces $32-$bits outputs L_i and R_i for $1 \leq i \leq 16$ after the formulae:

$L_i = R_{i-1}$

$R_i = L_{i-1} \bigoplus f(R_{i-1}, K_i)$

where $f(R_{i-1}, K_i) = P(S(E(R_{i-1}) \bigoplus K_i)))$, $\bigoplus$ denotes addition *mod* 2^n, E is a fixed expansion permutation mapping from 32 to 48 bits and P is another fixed permutation on 32 bits.

An initial bit permutation IP precedes the first round, following the last round the left and right halves are exchanged and the resulting string is bit-permuted by the inverse of IP. Deciphering involves the same key and algorithm, but with subkeys in the reverse order.

A novel generalization of the Feistel structure is the enciphering scheme IDEA (International Data Encryption Algorithm) , which mixes operations from three different groups of 2^n elements. This symmetric enciphering scheme processes plaintext blocks of $n = 64$ bits, producing $64-$bits ciphertext blocks using a secret ket K of $128-$bits

3.4 Stream enciphering schemes

Stream enciphering schemes are an important class of enciphering algorithms, which encipher individual characters of the plain text one at a time, using an enciphering transformation that changes with time. Generally are faster than block enciphering schemes in hardware implementations and they are sometimes more appropriate when buffereing is limited, when characters must be individually processed as they are received and when transmission errors are highly probable. There are relatively few fully-specified stream enciphering schemes in the open literature.

Definition 3.6 *(One-time-pad enciphering scheme) Denote by $m_1, m_2, m_3, ...$ the plaintext characters over the binary alphabet, by $k_1, k_2, k_3, ...$ the key binary digits (key stream) and by $c_1, c_2, c_3, ...$ the ciphertext digits.*

The enciphering transformation is defined by:

$c_i = m_i \bigoplus k_i$ *for* $i = 1, 2, 3, ...$

where $\bigoplus$ *denotes the bitwise addition mod 2.*

The deciphering transformation is defined by:

$m_i = c_i \bigoplus k_i$ *for* $i = 1, 2, 3, ...$

When the keystream digits are generated independently and randomly, the one-time-pad enciphering scheme is "unconditionally secure" against ciphertext-only attack, i.e., the ciphertext contributes no information about the plaintext. A drawback is that the key should be as long as the plaintext (difficult key distribution and key management). It motivates the use of keystreams pseudorandomly generated from a smaller secret key, which are no longer "unconditionally secure", but one hopes to be computationally secure.

4 Public key cryptosystems (PKC)

<< ... The security of a cryptosystem must not depend on keeping secret the crypto algorithm. The security depends on keeping secret the key...>>

- Kerckhoff von Nieuwenhof, La cryptographie militaire

4.1 The idea of PKCs

Assume that any participant X has one pair of keys, $e(X)$ and $d(X)$, such that the enciphering key is publicly known, but the deciphering key $d(X)$ is kept secret, and assume further that knowing $e(X)$, is it not computationally feasible, given a random ciphertext $c = E_{e(X)}(m)$, to determine the message m. This property implies that given $e(X)$ it is infeasible to determine $d(X)$.

The public keys $e(X)$ are publicly available (for instance, they may be stored in a public file, such as a telephone book) but the private deciphering keys $d(X)$ are known only to their owners.

How the PKC works:

1.- If A wants to send a message m to B, A looks up the public key $e(B)$ of B, A enciphers m using $e(B)$ and sends the enciphered text $c = E_{e(B)}(m)$ to B

2.- B is able to decipher the received ciphertext c, since only B knows the key $d(B)$, $D_{d(B)}(c) = m$

No other participant can decipher c, since by hypothesis, no one can deduce $d(B)$ from $e(B)$ (and $c = E_{e(B)}(m)$). In this process only keys related to B are used.

4.2 Advantages of PKCs

- No key exchange among participants is necessary. In comparison to symmetric cryptosystems, spontaneous communication is possible, without previous exchange of keys

- Relatively small number of keys. In symmetric cryptosystems, every pair of participants must have an secret key, in an asymmetric PKCs each participant needs only two keys (and only one of them is kept secret)

- New participants can joint the PKC without new problems for the older participants. In symmetric cryptosystems, if a new NP participant joints the system, then all older participants have to exchange a secret key with NP, in the asymmetric case, the older participants only have to update their database with the new public key (if they wish) of NP.

- The pairs private key/public key may remain unchanged for considerable periods of time.

4.3 Disadvantages of PKCs

- Symmetric cryptosystems are faster. Some hardware implementations achieve much better enciphering rates than efficient software implementations.

- Keys for symmetric cryptosystems are relatively short.

- Danger of impersonation. PKCs need some key management: If some unauthorized person I uses the name B of an authorized participant and sends to the old participants a public key $e(I)$ as the public key of B, then any participant X that assumes (incorrectly) that $e(I)$ is actually the public key of B will send messages to B using $e(I)$ and I may intercept the messages from X to B, decipher them with his own private key $d(I)$, re-encipher the message using the public key $e(B)$ and send it to B. That forces to check that the public keys are authentic, i.e. the sender of a message must obtain

by some means an "authentic" copy of the intended receiver's public key. There are elegant PK techniques to solve this problem.

4.4 The PKC RSA

Named after the names of its inventors R. Rivest, A. Shamir and L. Adleman, is the most widely used PKC. It security relays on the intractability of the integer factorization problem.

1.- Key generation for RSA: Any participant X has a public key $e(X)$ and a secret key $d(X)$, which are generated as follows:

- generate two large random (and different) numbers p and q, both approximately of the same size

- compute $n = pq$ and $\phi = (p-1)(q-1)$

- select a random integer ex, $1 < ex < \phi$, such that $gcd(ex, \phi) = 1$

- use the Euclidean algorithm to compute the unique integer dx, with $1 < dx < \phi$, such that $dxex = 1 \ (mod \ \phi)$

- set $e(X) = (n, ex)$ and $d(X) = dx$

The integer ex is called the enciphering exponent, the integer dx is called the deciphering exponent, n is called the modulus.

2.- RSA public key encryption scheme (A enciphers a message m for B, B deciphers)

- Enciphering:

a) A obtains B's authentic public key $e(B) = (n, ex)$

b) A represents the message as an integer $m \in [0, n-1]$

c) A computes $c = m^{ex} \ (mod \ n)$

d) A sends the ciphertext c to B

- Deciphering:

a) B uses the private key $d(B) = dx$, to recover $m = c^{dx} \ (mod \ n)$

Proof of correctness of the deciphering algorithm.-

Since $exdx = 1 \ (mod \ \phi)$, there exist an integer k, such that

$exdx = 1 + k\phi$. If $gcd(m, p) = 1$, then by Fermat's theorem holds

$m^{p-1} = 1 \ (mod \ p)$

Hence, raising both sides to the power $k(q-1)$ and multipliying by m, holds

$m^{exdx} = m \ (mod \ p)$

and analogously,

$m^{exdx} = m \ (mod \ q)$

Since p and q are different primes, after the Chinese Remainders Theorem

$m^{exdx} = m \ (mod \ n)$

and finally holds,

$c^{dx} = (m^{cx})^{dx} = m \ (mod \ n)$

Some comments.-

- RSA is substantially slower than commonly used symmetric key cryptosystems, such as DES. In practice, it is used for the transport of symmetric cryptsystems keys and for enciphering of small data items

- RSA PKC has been patented in the U.S. and in Canada, and some organizations have written or are in process of writing standars that address the use of RSA

- Security of RSA: The problem of computing the deciphering exponent dx from the public key (n, ex) is computationally equivalent to the problem of factoring n

- Recommended size of modulus: After the latest algorithms for factoring integers (1996), such as quadratic sieve and number field sieve, a modulus n of at least 768 bits is recommended, for long term security, 1024-bit or larger moduli should be used

- Selecting the primes p, q: In order to avoid the elliptic curve factoring algorithm, both primes should be of the same bitlength and sufficiently large, further their difference should not be too small (otherwise, $p^2 \backsimeq n$ and n could be factored by trial division by all primes close to $\sqrt{n}$

4.5 The discrete logarithm problem DLP

Let G be a (multiplicatively written) finite cyclic group of order n with generator α, i.e.,

$G = \{\alpha^j, j \in \mathbb{Z}\}$

Definition 4.1 *Discrete logarithm problem: given a finite cyclic group G of order n, a generator α of G and an element $\beta \in G$, find the integer j, $0 \leq j \leq n-1$, such that $\alpha^j = \beta$.*

Some commets:

- Solving the DLP in a cyclic group G of order n is essentially equivalent to compute an isomorphism between the multiplicative G and the additive group $\mathbb{Z}/n\mathbb{Z}$.

- Some known algorithms for the DLP:

1) algorithms for arbitrary groups (exhaustive search, baby-step giant-step, Pollard's rho algorithm), which are not efficient if the order of the group is large

2) algorithms for arbitrary groups, which are especially efficient if the order of the group has only small prime factors (Pohlig-Hellman algorithm)

3) the index-calculus algorithms, which are efficient only for certain groups, such as Coppersmith's algorithm for $G = \mathbb{F}_{2^m}$

4.6 The PKC ElGamal

Its security is based on the intractability of the discrete logarithm problem (DLP) in a finite, cyclic group G. The group must be chosen such as the following conditions hold:

- Efficency: the group operation in G should be relatively easy to apply,

- Security: the DLP in G should de computationally infeasible.

1.- Key generation for ElGamal PKC: Select a multiplicative cyclic group G of order n, with generator α and a multiplication algorithm in G for all participants. Any participant X has a public key $e(X)$ and a secret key $d(X)$, which are generated as follows:

- X selects a random integer a, $1 \leq a \leq n-1$ and computes the element α^a,

- X's public key is $e(X) = (\alpha,\ \alpha^a)$

- X's private key is $d(X) = a$

2.- ElGamal public key encryption scheme (A enciphers a message m for B, B deciphers)

- Enciphering:

- A obtains B's authentic public key $e(B) = (\alpha,\ \alpha^a)$

- A represents the message as an element m of the group G

- A selects a random integer k, $1 \leq k \leq n-1$ and computes $\gamma = \alpha^k$ and $\delta = m.(\alpha^a)^k$

- A sends to B the ciphertext $c = (\gamma, \delta)$

- Deciphering:

- B uses the private key $d(B) = a$ and computes γ^{-a}

- B recovers the message m by computing $(\gamma^{-a}).\delta$

Proof of correctness of the deciphering algorithm.-

$(\gamma^{-a}).\delta = \alpha^{-ak} m \alpha^{ak} = m$

- Some comments:

- Different random integers k must be used for enciphering different messages (otherwise, if one uses the same integer k to encipher two messages m_1 and m_2 and the resulting ciphertexts are (γ_1, δ_1) and (γ_2, δ_2), then

$\delta_1.\delta_2^{-1} = m_1.m_2^{-1}$ and m_2 could be easily computed if m_1 is known.

- Recommended parameter sizes for $G = \mathbb{Z}_p^*$: As of 1996, a modulus p of at least 768 is recommended, for long term security, 1024-bit or larger. If G and its generator α are public, p must be even larger, since the precomputation of a database of logarithms required in the index-calculus algorithm for any particular p will comprise the secrecy of all private keys using p as modulus.

4.7 The PKC McEliece

Is based on error-correcting codes. One selects first a particular code, for which an efficient decoding algorithm is known, and then one disguises the code as a general linear code.

1.- Key generation for McEliece PKC: The parameters k, n and t are fixed as common parameters. Any participant X has a public key $e(X)$ and a secret key $d(X)$, which are generated as follows:

- X chooses a $k \times n$ generator matrix G for a (n, k)-linear code which can correct t errors and with known efficient decoding algorithm

- X selects a random $k \times k$ non-singular matrix S and also a random $n \times n$ permutation matrix P

- X computes the $k \times n$ matrix $GD = SGP$

- X's public key is $e(X) = (GD, t)$; X's private key is $d(X) = (S, G, P)$

2.- McEliece public key encryption scheme (A enciphers a message m for B, B deciphers)

- Enciphering:

- A obtains B's authentic public key $e(B) = (GD, t)$

- A represents the message m as a string of length k

- A chooses a random error vector z of length n and weight at most t

- A computes the vector $c = mGC + z$ and sends c to B

- Deciphering:

- B computes $c^* = cP^{-1}$

- using the efficient decoding algorithm for the code G, B decodes c^* to m^*

- B recovers m by computing m^*S^{-1}

Proof of correctness of the deciphering algorithm.-

$$c^* = cP^{-1} = (mGD + z)P^{-1} = (mSGP + z)P^{-1} = (mS)G + zP^{-1}$$

Since zP^{-1} is a vector of weight at most t, the decoding algorithm for the code generated by G corrects c^* to $m^* = mS$, hence $m^*S^{-1} = m$ and the deciphering works.

The McEliece PKC has resisted cryptanalysis to date. It is the first PKC to use radomization in the enciphering process. The ciphering and deciphering algorithms are very efficient, but the very large size of the public keys are a practical drawback. Its security is based on the NP-hardness of decoding a general linear error correcting code.

5 Authentication.

<< ...The King had Allerleirauh brought before him. He spied the white finger and saw the ring which he had put on it during the dance. Then he grasped her by the hand and held her fast, and when she moved to release herself and run away, her fur mantle opened and the star dress shone forth. The King clutched the mantle and tore it off. Then her golden hair shone forth and she stood there in full splendor, and could no longer hide herself ...>>

- Brothers Grimm, "Allerleirauh"

5.1 Some definitions.

Definition 5.1 *(1-1 functions) A function $f : X \longrightarrow Y$ is 1-1 (one-to-one) if each element in the codomain Y is the image is the image of at most one element in the domain X.*

Definition 5.2 *(One-way functions) A function $f : X \longrightarrow Y$ is called a one-way function if for all $x \in X$, $f(x)$ is "easy" to compute, but for "most"*

elements $y \in Im(f)$ is "computationally infeasible" to find an $x \in X$, such that $f(x) = y$.

Example 5.1 *(One-way functions) Given two prime numbers p and q, set $n = pq$ and define the function*

$$f(x) = x^3 \ (mod n)$$

If the factors of n are unknown and large, to compute x from n and $f(x)$ is a difficult problem.

Definition 5.3 *(Trapdoor one-way functions) A trapdoor one-way function is a one-way function $f : X \longrightarrow Y$, such that given some extra information (called the trapdoor information) it becomes feasible to find for any given $y \in Im(f)$ an $x \in X$ with $f(x) = y$.*

Definition 5.4 *A hash function h is a many-to-one function that maps bit-strings of arbitrary finite length to strings of fixed length, say n bits, such that the possibility of collisions (pairs of inputs producing the same output) is "very small".*

Hash functions take a message as input and produce an output referred to as hash code or hash value (or simply hash). Restricting h to a domain of t-bits inputs ($t > n$), if h were "random" (all outputs equiprobable) then about 2^{t-n} inputs would map to each output, hence two randomly chosen inputs would yield the same output with probability 2^{-n}, independently of t.

The main idea of using hash functions in cryptography is that they serve as a compact representative image (imprint, data fingerprint, message digest) of an input string and can be used as if it were uniquely identifiable with the original string. Hash functions are also used in applications where the one-way property is required but not compression.

Definition 5.5 *(MAC) A Message authentication code (MAC) algorithm is a family of functions h_k parametrized by a secret key k, with the properties:*

- for a known function h_k, given a value k and an input x, $h_k(x)$ is easy to compute.

- h_k maps an input x of arbitrary finite bitlength to an output $h_k(x)$ of fixed bitlength.

- given pairs $(x_i, h_k(x_i))$ it is computationally infeasible to compute any pair $(x, h_k(x))$ for any new input $x \neq x_i$, including possibly for $h_k(x) =$

$h_k(x_i)$.

5.2 MAC based on block ciphers

Cipher-block-chaining (CBC) based MAC algorithm . When DES is used as the block enciphering scheme E, $n = 64$ and the MAC key is a 56-bit DES key.

Algorithm CBC-MAC:

- Input: data x, specification of block cipher E; secret MAC key k for E

- Output: n-bit MAC on x (n is the blocklength of E)

1.- Padd (append 0-bits to x untill obtaining a data string x' with length multiple of n) and divide the padded text into n-bits blocks denoted $x_1, ...x_t$

2.- Denoting by E_k the encipering function using E and k, compute the block H_t as follows: $H_1 \leftarrow E_k(x_1)$; $H_i \leftarrow E_k(H_{i-1} \bigoplus x_i), 2 \leq i \leq t$ (this is standard cipher-block chaining, $IV = 0$, discarding ciphertext blocks $C_i = H_i$)

3.- Using a second secret key $k' \neq k$, optionally compute $H'_t \leftarrow E_{k'}^{-1}(H_t)$, $H_t \leftarrow E_k(H'_t)$

4. The MAC is the n-bit block H_t

5.3 MAC and data integrity

If confidentiality and integrity are needed, the following data integrity technique using a MAC may be used. The originator of a message x computes a MAC-value $h_{k'}(x)$, appends it to the data and enciphers the augmented message using a symmetric enciphering algorithm E with shared key k, producing a ciphertext $C' = E_k(x \| h_{k'}(x))$. The recipient determines (using a plaintext identifier field) which keys k and k' were used and separates the recovered data x' from the recovered MAC-value $h_{k'}(x)$ and compares it with $h_{k'}(x')$, if both agree, the recovered data are accepted as being authentic and having integrity.

6 Digital Signature

<< ... Many a man applies his intellect to simplify, many to complicate...>>
- Erich Kaestner

6.1 PKCs and digital signatures

If using the public key of B, A inserts (at the beginning or at the end of the message)the ciphertext $(E_{e(B)}D_{d(A)})(c_A)$, where c_A is a secret, but meaningful message identifying A. Then, when B deciphers the whole message, including this part, using $D_{d(B)}$, he finds that everything makes sense, except a small section. Since the message is claimed to be from A, then B can easily verify it, applying $E_{e(A)}$ to that part of the deciphered message and obtaining c_A. No one other than A could have computed $D_{d(A)}(c_A)$, since $d(A)$ is secret.

- PKS provide efficient digital signature schemes ("public key signature schemes") with much smaller keys than the symmetric counterpart.

6.2 The Digital Signature Algorithm DSA

The first digital signature scheme recognized by any government, the DSS is based on the digital signature algorithm DSA proposed in 1991 by NIST. It is based on ElGamal scheme and is a digital signature scheme with appendix (i.e., it requires the message as input to the verificaion algorithm).

The signature mechanism requires a hash function $h : \{0,1\}^* \to \mathbb{Z}_q$ for some integer q. The DSS explicitly requires use of the hash function SHA-1.

1.- Key generation for the DSA (each entity A creates a public and a private key) :

- Select a prime q such that $2^{159} < q < 2^{160}$

- Choose t so that $0 \leq t \leq 8$ and select a prime number p, where $2^{511+64t} < p < 2^{512+64t}$, with the property that q divides $p-1$.

- Select a generator α of the unique cyclic group of order q in $\mathbb{Z}_p^*$:

(i) Select an element $g \in \mathbb{Z}_p^*$ and compute $\alpha = g^{(p-1)/q} \bmod p$

(ii) If $\alpha = 1$ then go to step (i)

- Select a random integer a, such that $1 \leq a \leq q-1$

- Compute $y = \alpha^a \bmod p$

- A's public key is (p, q, α, y); A's private key is a.

2.- DSA signature generation and verification (entity A creates a binary message m of arbitrary length; any entity B can verify this signature by using A's public key) :

(i) Signature generation. Entity A should do the following:

(a) Select a random secret integer k, $0 < k < q$

(b) Compute $r = (\alpha^k mod\ p) \bmod q$

(c) compute $k^{-1} \bmod q$

(d) compute $s = k^{-1}\{h(m) + ar\} \bmod q$

(e) A's signature for m is the pair (r, s)

(ii) Verification (to verify A's signature (r, s) on m, B should do the following):

(a) Obtain A's authentic public key (p, q, α, y)

(b) Verify that $0 < r < q$ and $0 < s < q$ if not, reject the signature

(c) Compute $w = s^{-1} \bmod q$ and $h(m)$

(d) Compute $u_1 = w.h(m) \bmod q$ and $u_2 = rw \bmod q$

(e) Compute $v = (\alpha^{u_1} y^{u_2} \bmod p) \bmod q$

(f) Accept the signature iff $r = v$

Proof that signature works. If (r, s) is a legitimate signature of A on m, then $h(m) = -ar + ks \pmod q$ must hold. Multiplying both sides by w, we get $w.h(m) + arw = k \bmod q$, hence $u_1 + au_2 = k \bmod q$. Raising α to both sides of this equation yields $(\alpha^{u_1} y^{u_2} \bmod p) \bmod q = (\alpha^k \bmod p) \bmod q$. Thus, $v = r$.

7 Some computational problems of cryptographic relevance

- **Factoring:** Given a positive integer n, find its pairwise prime factorization.
- **RSA inversion:** Given a positive integer $n = pq$, a positive integer e, $gcd(e, (p-1), (q-1)) = 1$, and an integer c, find an integer m such that, $m^e \equiv c \; mod \; n$
- **Square root mod n:** Given a composite integer n and b, a quadratic residue $mod \; n$, find a square root of $b \; mod \; n$
- **DLP**
- **Diffie-Hellman problem:** Given a prime p and a generator $\alpha \in \mathbb{Z}_p^*$ and elements $\alpha^a \; mod \; p$ and $\alpha^b \; mod \; p$ find $\alpha^{ab} \; mod \; p$.

7.1 Comparing asymptotic behaviour of functions

- $f(n) = O(g(n)) \quad \exists \; c > 0, \;\; n_0 \in \mathbb{N}$

$$0 \leqq f(n) \leqq cg(n), \quad n \geqq n_0$$

- $f(n) = \Omega(g(n)) \quad \exists \; c > 0, \;\; n_0 \in \mathbb{N}$

$$0 \leqq cg(n) \leqq f(n), \quad n \geqq n_0$$

- $f(n) = \Theta(g(n)) \quad \exists \; c_1, c_2 > 0, \;\; n_0 \in \mathbb{N}$

$$c_1 g(n) \leqq f(n) \leqq c_2 g(n), \quad n \geqq n_0$$

- $f(n) = o(g(n)) \quad \forall c > 0, \;\; \exists \; n_0 \in \mathbb{N}$

$$0 \leqq f(n) \leqq cg(n), \quad n \geqq n_0$$

- $f(n) = L_n[\alpha, c]$ if for some $c > 0, \quad 0 < \alpha < 1,$

$$f(n) = O(\exp((c + o(1))(\ln n)^{\alpha}(\ln \; \ln n)^{1-\alpha}))$$

Recall that, $\alpha = 0$ implies $f(n)$ is polynomial in $\ln n$ and $\alpha = 1$ implies $f(n)$ is polynomial in n

Examples

- $f(n)$ polynomial of degree $k \quad \Longrightarrow f(n) = \Theta(n^k)$
- $\forall c > 0, \quad \log_c(n) = \Theta(\lg(n))$
- $\sqrt{2\Pi n} \quad (\frac{n}{e})^n \leq n! \leq \sqrt{2\Pi n}(\frac{n}{e})^{n+\frac{1}{12n}}$

  $\Rightarrow \quad n! = o(n^n) \quad and \quad n! = \Omega(2^n)$
- $\lg(n!) = \Theta(n \lg(n))$

Definition 7.1 *A Polynomial-time algorithm is an algorithm whose worst case running time function is of the form $O(n^k)$, where n is the input size and k is a constant. Any algorithm whose running time can not be so bounded is called exponential time algorithm.*

Definition 7.2 *A subexponential-time algorithm is an algorithm whose worse case running time function is of the form $e^{o(n)}$, where n is the input size.*

Definition 7.3 *A computational problem is tractable if it can be solved in polynomial time, at least for a non-negligible fraction of all possible inputs.*

7.2 Some complexity estimates

- **Pollard rho algorithm for factoring integers n:** $O(\sqrt{n})$ space and $O(\sqrt{n})$ time.
- **Elliptic curve factoring algorithm:** It has expected running time $L_p[1/2, \sqrt{2}]$ to find a factor p of n. In the hardest case, when n is the product of two primes of the same size, the expected running time is $L_n[1/2, 1]$.
- **Finding square roots modulo a prime p :** It has expected running time $O((\log p)^3)$.
- **Baby-step-giant-step algorithm to solve DLP:** $O(\sqrt{n})$ group operations, $n =\mid G \mid$.
- **Pollard's rho algorithm for DLP:** $O(\sqrt{n})$ group operations for $G = \mathbb{Z}_n^*$.
- **Pohlig-Hellman algorithm to solve DLP:** if

  $$n = \prod p_i^{\alpha_i}$$

  $O(\sum \alpha_i(\lg n + \sqrt{p_i}))$ group multiplications, hence it is efficient only if each prime divisor p_i of n is small.

8 Jacobian Varieties suitable for DLP based cryptosystems

Pollard rho methods shows that for general abstract groups G, the discrete logarithm problem needs probabilistically $O(\sqrt{(}n))$ group multiplications for $n = |G|$.

Hence, it is important to give methods for the construction of groups (more precisely of presentations of $\mathbb{Z}/p\mathbb{Z}$, with p large). One possible way is to use embedings of $\mathbb{Z}/p\mathbb{Z}$ into the group of rational points of Jacobian Varieties J_C of curves C defined over finite fields.

We have to solve three tasks :

- Construct Jacobian Varieties J_C over finite fields $\mathbb{F}_q$, such that both the points on J_C and the addition law on J_C are explicitly given in a way that needs only $O(log\ p)$ bits of space and such that the addition is done in $O(dim(J_C))$ additions in $\mathbb{F}_q$
- Select Jacobian Varieties J_C satisfying the first condition, such that large primes p divide $|J_C(\mathbb{F}_q)|$. In adition, $log\ p$ should be nearly equal to $log(|J_C(\mathbb{F}_q)|) = (dim\ J_C) log\ q$
- Select Jacobian Varieties J_C satisfying the previous two conditions, such that all known algorithms for the computation of the DLP in $J_C(\mathbb{F}_q)$ have probabilistic complexity $\geq O(\sqrt{p})$

In principle, the addition law on J_C is given effectively by Riemann-Roch Theorem, but in practice, we should restrict ourselves to curves with a cover map to the projective line of fixed degree d, with d small.

The addition on Jacobian Varieties of hyperelliptic curves is based on Cantor algorithm, which can be translated to Gauss theory of binary quadratic forms, and works for any characteristic (including $char = 2$). An efficient implementation and estimates of its performance can be found in U. Krieger Diplome Thesis (Essen, 1997).

References

[1] L.M. Adleman, J. DeMarrais, A subexponential algorithm for discrete logarithms over all finite fields, Math. of Computation, 61 (1993), 1-15.

[2] L.M. Adleman, J. DeMarrais, M.-D. Huang, A subexponential algorithm for discrete logarithms over the rational subgroup of the jacobians of large genus hyperelliptic curves over finite fields, Algorithmic Number Theory, Lect. Notes Comp. Sci. 877, Springer-Verlag (1994), 28-40

[3] A. Beutelspacher, Cryptology, The Mathematical Association of America, Spectrum Series, 1994

[4] D.Cantor, Computing in the Jacobian of a hyperelliptic curve, Math. of Computation, 48 (1987), 95-101.

[5] W. Diffie, M.E. Hellman, New directions in Cryptography, IEEE Trans. Information Theory 22 (1976), 644-654.

[6] T. ElGamal, A public key cryptosystem and signature scheme based on discrete logarithms, IEEE Trans. Information Theory 31 (1985), 469-472.

[7] G. Frey, H. Rueck, A remark concerning m-divisibility and the discrete logarithm problem in the divisor class group of curves, Math. of Computation, 62 (1994),865-874.

[8] G. Frey, Cryptographic Constructions Based on Galois Actions, in Proc. Conference on The Mathematics of Public Key Cryptography, Ed. A. Odlyzko, G. Walsh, H. Williams, Toronto, Ontario, June 12-17, 1999.

[9] D. Kahn, The Code Breakers: The Story of Secret Writing, Macmillan, New York, 1967.

[10] N. Koblitz, A course in number theory and cryptography, Springer-Verlag Graduate Texts in Math. 114, 1994.

[11] R.J. McElliece, A public key cryptosystem based on algebraic coding theory, JPL DSN Progress Report 42-44 (1978), 114-116.

[12] A. Menezes, Elliptic curve Public Key Cryptosystems, Kluwer Acad. Pub., 1993.

[13] A. Menezes, P. van Oorschot, S.A. Vanstone, Handbook of Applied Cryptography, CRC Press, 1996.

Regular publications on cryptology : Journal of Cryptology (Springer-Verlag), Computer and Security (Elsevier) are worth of a special mention. The Springer Lecture Notes in Computer Science publish yearly two volumes called Advances in Cryptology, which are the procedings of the annual conferences Eurocrypt and Crypto.