INNOVATIVE TECHNIQUES FOR DATA SECURITY: A VIEW OF THE CRYPTOGRAPHIC ALGORITHMS IN A VIRTUAL WORLD

Simona Bibic, Carmina Georgescu, Emil Simion, Antonela Toma

UNIVERSITY POLITEHNICA of BUCHAREST
FACULTY OF APPLIED SCIENCES
Department of Methods and Mathematical Models
Center for Research and Training in Innovative Techniques of Applied Mathematics in Engineering (CITI)

Emails: simona.bibic@upb.ro; emil.simion@upb.ro; carmina.georgescu@upb.ro; antonela.toma@upb.ro

This course is composed of two theoretical parts that present the notions of cryptographic algorithm and standardization of cryptographic modules, as well as a module of cryptographic algorithms designed to ensure information security (including available software resources).

Contents

MODULE 1. CRYPTOGRAPHIC ALGORITHMS IN A VIRTUAL WORLD

INTRODUCTION

Cryptography is used to protect data from eavesdropping and also the integrity of the data. There are two types of cryptographic techniques: symmetric and asymmetric. The symmetric techniques involve the usage of the same key in encryption and decryption process or in integrity verification process. The asymmetric techniques are base on the computational difficulty of number theory problems such as factoring the product of two large numbers or discrete logarithms problem.

Communication systems and storage device of data are not reliable in practice because of noise or other forms of introduced interference. The purpose of coding theory [4] is to detect and even correct errors. Usually, coding is defined in terms of source coding and channel coding. Source coding involves changing the message source to a suitable code to be transmitted through the communication channel. The idea of channel coding is to encode the message again after the source coding by introducing some form of redundancy so that errors can be detected or even corrected. Cryptography techniques are used to protect the integrity (including authentication and non-repudiation) and confidentiality of the stored and/or transmitted data. The general scheme of using message source, source encoding and encryption is presented in Figure 1. Source encoding performs, using error detection and correction methods, the reliability of communication through the communication channel.

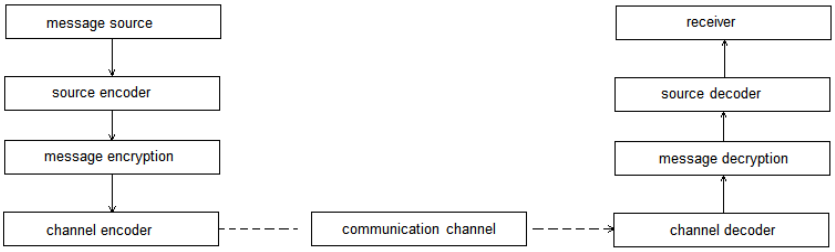

Figure 1 Encoding and encryption data thus a communication channel.

In this lecture we present a synthesis of cryptographic algorithms used for data protection and integrity through a communication (inclusive storage) channel.

SYMMETRIC ALGORITHMS

In the cathegory of symmetric algoritms we find algoritms which ensures condidentiality and/or integrity.

Let's talk about cryptographic primitives which ensures confidentiality: symmetric ciphers. Symmetric ciphers use the same key for encryption and decryption. The information which we desire to protect can be stored information on different types of media and transmitted information on different types of communication chanells. Therefore we need to design two types of ciphers to ensure protection of stored data and protection of transmitted data. Block ciphers are ideal for off-line encryption (storage protection) while stream ciphers are used for encrypting communication. The main difference between this two types of ciphers is that block ciphers have no internal state while stream ciphers have a internal state. Usually the encryption operation in case of usage of stream ciphers is xor-ing the plain text between data and the of stream cipher output. There are methods for transformation a block cipher in a stream cipher and viceversa.

For example, there are several characteristics of block ciphers [5], [2]:

- estimated level of security versus unconditioned security;
- length of effective key K (e.g. in case of AES standard the key size is 128, 192 or 256 bits);
- complexity of output function;
- structure type (e.g. Feistel type in case of DES standard);
- operations used (e.g. AES use subtitutions (S-BOX operation), traspositions (shift row operation), galois field multiplication (mix columns operation) and xor (add round key operation));
- lengh of plaintext block;
- number of rounds (usually depends of the key size and plain text size);
- text inflation: difference between length of plaintext length of cipher text;
- key schedule algorithm;
- balance between encryption and decryption functions;
- complexity of implementation in hardware and software;

- error propagation;
- mode of operations (e.g. Electronic Code Book (ECB), Cipher Block Chaining (CBC), Output Feedback Block (OFB) etc.).

All the above aspect were taken in discussions in AES selection process (1997-2001).
There are several stream ciphers but non of them is standardized at this time. The european project eSTREAM (2004-2008) fail to do this standardization.

ASYMMETRIC ALGORITHMS

Asymmetric cryptography allows two or more parties to develop a cryptographic protocol for secure key exchange over an insecure communication channel. There are several asymmetric primitives, based on integer factorizations (IF) of large numbers (e.g. RSA used for encryption and/or signature), discrete log (DL) computing (e.g. El Gamal used for encryption and/or signature, Digital Signature Algorithm (DSA), Diffie-Hellman (DH) protocol that allows two parties that have no prior knowledge of each other to jointly establish a shared secret key etc.) or elliptic curves theory (EC).

In public key cryptography there are two keys for each user: a public key, which is used to encrypt the message and a private key, which is used to decrypt the message. The security of a public key encryption scheme is based on the fact that it is computationally difficult to derive the private key form the public key.

Integer factorization

RSA gets its security from the difficulty of factoring large numbers. The public and private keys are functions of a pair of large (*100* or *200* digits or larger) prime numbers. Recovering the plain text from the public key and the ciphertext is conjectured to be equivalent to factoring the product of two primes.

Discrete log theory

ElGamal algorithm is both use for confidentiality and authentication processes. The strength of tha algorithm is based on the difficulty of solving the discrete log problem in a finite field. To generate a public and private key we chose at the beginning a prime number p and two numbers g and x smaller then p. We compute $y=g^x \bmod p$. The public key is $\{y,g,p\}$ and the private key is x. Both numbers g and p can be used in common by a group of users. To

encrypt a message *M* we chose first a random number *k* relative prime with *p-1*. We compute after

$$\begin{cases} a = g^k \bmod p \\ b = y^k M \bmod p \end{cases}$$

The pair (*a,b*) is the cipher text which length is twice then the length of plain text. For decrypting a cipher text (*a,b*) we compute: $M=ba^{-x} \bmod p$.

Elliptic curves theory

The advantage of using EC [3] versus factorization of large numbers is the key size: the same security margins are achieved using a smaller amount of key bits. EC, used in cryptography, are defined over a general field and the standardized ones are defined over a primes field and over a Galois fields (binary fields). Generally speaking EC, over binary fields are easy to implement in then EC over prime fields. RSA-based protocols have been adapted to EC, replacing the group $\boldsymbol{Z}_{pq}$ with an EC: the ECDH key agreement scheme is based on the DH scheme, the ECDSA is based on the DSA and the ECMQV key agreement scheme is based on the MQV key agreement scheme. As we know there are several cryptographic primitives like symmetric (block and stream ciphers, hash functions etc.) and asymmetric one (RSA, (EC) DSA etc.). EC needs a smaller amount of bits than DL or IF. Some of the application that involve elliptic curve computations are secure communications, e-commerce and digital money.

HYBRID SYSTEMS

Hybrid systems are composed by combination of asymmetric and symmetric primitives to provide confidentiality and integrity. Usually confidentiality of information is achieved using a symmetric primitive and the authenticity and non-repudiation are obtained using asymmetric primitives such as RSA, DSA or ECDSA algorithms.

In the case of ensuring confidentiality, the data is usually encrypted by the sender *A* with a symmetric algorithm, the encryption key is encrypted with the public key of the receiver *B*.

Another cryptographic primitive is hash function, which computes a digest of the data. This hash value is signed with the private key of the sender *A*, the receiver *B* will check the

signature using the public key or *A*. In Figure 2 is presented the process of signing a document using a PKI infrastructure based on RSA signing function.

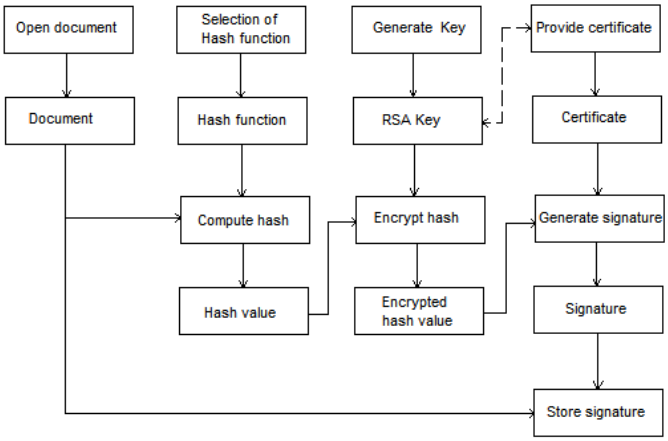

Figure 2 Flowchart of the signing process of an electronic document

REFERENCES

[1] Alexander W. D., Chris J. M. (2006). *User's guide to Cryptography and Standards*, Artech House.

[2] Daemen J. and Rijmen V. (2002), *The design of RIJNDAEL: AES - the Advanced Encryption Standard* (Information Security and Cryptography), Springer-Verlag.

[3] Hankerson D., Menezes A., Vanstone S. (2004), *Guide to Elliptic Curve Cryptography*, Springer-Verlag New York, Inc.

[4] Ling S. and Xing C. (2004). *Coding Theory*, Cambridge University Press.

[5] Nechvatal J. et. al. (2000). *Report on the Development of the Advanced Encryption Standard (AES),* National Institute of Standards and Technology.

[6] Schneier B., Applied Cryptography, Protocols, Algorithms, and Source Code in C, 20th Anniversary Hardcover: ISBN 978-1-119-09672-6, John Wiley&Sons, 2006.

MODULE 2. CRYPTOGRAPHIC STANDARDS

INTRODUCTION

An actual problem in the field of information security, that has no unitary point of view, is that of the methods and techniques used in the evaluation of the cryptographic algorithms proposed as public cryptographic standards. This problem appeared due to the high level of heterogeneity of the cryptographic field and because of the interdisciplinary character of this domain.

The problem is systematically taken into account by the International Association of Cryptographic Research - IACR (http://www. iacr.org/) in more then 60 annual international conferences (http://www.iacr.org/ events/), such as CRYPTO, EUROCRYPT, ASIACRYPT, AFRICA CRYPT with various audience representing governmental institutions, research centers and prestigious universities all over the world.

Considering those facts this chapter will present some ideas, untouched yet in our country, concerning the evaluation process and cryptographic standards, subject with significance impact in most countries and for Romania as well.

THE CHARACTER OF INTERDISCIPLINARY FUNDAMENTAL RESEARCH

The overall objective of this research is the analysis of the main results and of a number of present international standards in the field of information security, explicitly concerning the applied cryptography. As a starting point, the study will employ the mathematical models used for every known standard, developed as a result of national and international projects. By using the comparative analysis of these standards, it is intended to develop a common model that will lead to a national standard in the field. This method was also used by a number of EU and NATO Member States. Such a research can be carried out, from our point of view, by a team consisting of mathematicians, specialists in IT&C, automatics and electronics, with expertise in fundamental fields, that have the capability to develop this standard that is compulsory for the unitary implementation, at a national level, of the main

theoretical INFOSEC concepts, possibly in the form of a reference model. The team of mathematicians will study the optimization of the existing models with a view to adapt them to the requirements needed for the implementation in a real system. At this level, a link will be established between the fundamental research and a model suitable for implementation in real systems. In this phase, the team of researchers specialized in optimized implementation will be involved. This team will restrict the model in order to create hardware / software products to comply with the commercial and governmental requirements. Following the testing phase and the analysis by the team specialized in standards and minimum security levels required for each communications level in the national inter-relation data system, there will be a phase of development proposals for models to ensure the confidentiality of information. The models shall be complete from the structural point of view, meaning that it will have a mathematical model, implementation system comprising of analysis schemes, general schemes for adjustment to different systems and analysis of the risk level, as well as the usable API for real applications. This phase will be ensured by the team having vast experience in hardware and software implementation.

THE OBJECTIVES AND THE METHODOLOGY OF RESEARCH

Regardless their category, standards are important because they define practices, methods and measures to be followed by the developer, evaluator and customer. For this reason, standards enhance the products reliability and efficiency that results in an improvement of the quality. Standards provide solutions that are accepted by the specific community they address to, and are used by experts in that field. By using standards, organizations are able to reduce costs and to protect their technological investments. For the IT&C field, standards provide interoperability, security and integrity.

Interoperability. Products developed in compliance with a standard are comparable and can be used in conjunction with other products developed in compliances with the same category of standards. By ensuring interoperability between different equipments, standards enable organizations and companies to choose a certain product in order to minimize expenses.

Security. Standards may be used to ensure a certain common level of security. Most managers have no expertise in security issues but, by using a standardized algorithm (e.g.

Federal Information Processing Standards) they are certain that this was analyzed by the standardization organization (in this case National Institute of Standards and Technologies, http://www.nist.gov).

Integrity. Standards may be used to ensure integrity of a certain product.

Starting from the way standards are developed, some steps can be made by developing a system to analyze the mathematical models used for the present international standards, the development of an optimized system to be implemented in a source to ensure information confidentiality in commercial and governmental systems. Some comparative studies of technique and methods used in the evaluation of cryptographic public standards can be made, as follows:

- the NESSIE projects (project developed by the European Commission during 2000-2003 in the framework of the IST Information Society Technologies program that proposed the drawing up and evaluation of a number of cryptographic primitives),
- ECRYPT - European Network of Excellence for Cryptology, project under development within the framework of IST-2002-507932 program, both projects being under the coordination of the Katholieke Universiteit Leuven (partners list published at http://www.ecrypt.eu.org/ partners.html,
- CRYPTREC (project regarding the cryptographic evaluation financed by the Japan Government),
- standardization bodies National Institute of Standards and Technologies (NIST – standardization body under the US Government), American National Standards Institute (ANSI – industrial standardization body from the US), GOST (governmental standardization organism within the Community of Independent States, http://www.gost-r.info/).

We also have to take into account the NATO, EU and national standards in the field, whenever possible. The study has to take into consideration the means of approaching the side domains, determined by the implementation of cryptographic algorithms in cryptographic modules and of the modules in cryptographic systems. This approach provides a constructive mediation of the ongoing conflict of the cryptographic policies between the commercial, research and governmental dominions. This research has a multidisciplinary approach, including comparative aspects related to the interrelationship between cryptographic standards and IT&C security standards. The comparative approach

will be layered on comparative studies of the standards regarding the cryptographic algorithms, the cryptographic module and IT&C security.

EXPECTED RESULTS

The research is intended to formulate the general framework for the establishment of the requirements (e.g. data format, test vectors, reference implementation) to be met by evaluated and standardized cryptographic algorithms (e.g. by NIST - Cryptographic Toolkit, IEEE, NESSIE, ECRYPT) as follows:

- block ciphering standards;
- stream ciphering standards;
- asymmetric ciphering standards;
- digital signature standards;
- hash function standards;
- message authentication standards;
- entity authentication standards.

The aspects required for accomplish the protection profile of the cryptographic algorithm are distinguished on the Security evaluation, implementation Costs and its Characteristics. In fact, our approach will be structured on the following directions stipulated by NIST regulations:

- development of attached mathematic models, their optimization through adaptation depending on analyses obtained for individual standards existing at global level, evaluation of computational security limits, taking into account the overlapping of the cryptographic algorithms to general attacks types (statistical evaluation, exhaustive research attack), and to particular attacks types (e.g. block cipher type differential cryptanalysis SPN). For a consistent approach we'll take into account the definition of the following concepts: time complexity, space complexity, data complexity, and also space-time compromise. A special attention will be paid to "hard resolvable" problems with applications in cryptographic type standards (factoring matter and discrete algorithm matter, both in Galois field, and elliptic curves);

- software implementation, taking into account the size of the processor word, implementation language, variation of the processing rate in regard to key size, rate variation in regard to algorithm mode of operation, data processing rate and a comparison of them, in face of particular types of restraints regarding RAM and ROM memory space;
- restrains regarding RAM and ROM, taking into account the memory resources required for cryptographic algorithms implementation. Small resources of implementation space may allow the implementation of the algorithms on specific platforms such as SMART CARDS;
- hardware implementation, taking into account architectural model, implementation methodologies and objectives, implementation within FPGA and ASIC, data processing rate, hardware characteristics, the possibility of distributed computation or collimation of the computations made within specific processing;
- attacks against implementation. Shall be taken into account the following types of attacks assuming cryptanalysis simulation depending on implementation (software or hardware):

FA (fault analysis) – during an arithmetic operation, hardware equipments can produce errors (transient, latent or induced), and through a national exploitation of these errors the private key can be recovered for a series of asymmetric algorithms;

TA (timing attacks) – by calculating execution time we can assume which bytes are zero and which are one;

DPA (differential power analysis) – power consumption depending on the executed instruction, and being monitored we can deduce the source code; in case the instruction sequence depends on key length, than the power consumption can provide information regarding that;

SPA (simple power analysis) – consist in assessing the power spent by the device during cryptographic operation, this type of attack being applied, as a rule, to the devices having exterior voltage source (like SMART CARDS);

EMA (electromagnetic attacks)

- encryption vs. decryption, taking into account the complexity (space - time) of the encryption procedure against the decryption procedure;
- key generation algorithm which, in case of symmetric algorithms, has a special importance from the view of security and of processing rate;
- capabilities and flexibility;

- possibility of implementations collimation at infrastructures level;
- copyrights, taking into account the interoperability character of those.

This research intends to be updatable, in sense that during its development or after its completion, considering international results in the field, if new methods and technologies are developed, these shall be included into a updated version of this project. In fact, this is a standard procedure applied in most international organizations and developed states for motivating both theoretical progress in the field, and the technologic one in implementation and manufacture.

LESSON LEARNED

This lecture tried to formulate, a series of requirements for the future development of cryptographic standards in UE. Having these requirements (relatively) standardized, the evaluation itself becomes more standardized – hence more efficient. Consequently, a metric easily and convincingly can be established for the space of the cryptographic primitives, and a hierarchy based on alternative criteria, other than those concerning the cryptographic strength, can be derived and used subsequently.

REFERENCES

[1]. Alexander W. D., Chris J. M. (2006). *User's guide to Cryptography and Standards*, Artech House.

[2]. ISO standards: http://www.iso.ch/

[3]. IEC standards: http://www.iec.ch/

[4]. ITU standards: http://www.itu.int/ITU-T/

[5]. ANSI standards: http://webstore.ansi.org/

[6]. BSI standards: http://www.bsi-global.com/

[7]. NIST standards: http://www.nist.gov/ , http://www.csrc.nist.gov/

[8]. 3GPP standards: http://www.3gpp.org/

[9]. ETSI standards: http://www.etsi.org/

[10]. IEEE standards: http://standards.ieee.org/

[11]. IETF standards: http://www.ietf.org/

[12]. SECG standards: http://www.secg.org/
[13]. PKCS standards : http://www.rsasecurity.com/rsalabs/pkcs/index.html
[14]. NESSIE evaluation project: http://www.cryptonessie.org/
[15]. ECRYPT evaluation project: http://www.ecrypt.eu.org/
[16]. CRYPTREC evaluation project: http://www.ipa.go.jp/security/enc/CRYPTREC/index-e.html

MODULE 3. SOME EXAMPLES OF RELEVANT CRYPTOGRAPHIC ALGORITHMS & TECHNIQUES

INTRODUCTION

Next we consider it useful to define some of the main notions used in the framework this collection of problems. **Cryptology** is the science of secret writing, with the aim of defending the secrecy of data and confidential information, using cryptographic systems. **Cryptography** is the defensive side of cryptology, having as object of activity the elaboration (designing) cryptographic systems and rules used. **Cryptanalysis** is the offensive side of cryptology, having as object of activity the study of its own cryptographic systems in order to offer them the necessary characteristics, so that they can perform the function for which they were designed. At the same time, cryptanalysis can analyse systems, cryptographic data of the third parties (through the cryptograms made with them) breaking them to obtain useful information for the institution it serves. By **cryptographic algorithm** we mean a set of oneway transformations by which the set clear messages (texts) in a language turn into the set M of cryptograms. **The encryption key** is a particular convention, materialized, in a word, phrase number etc. and which directs (regulates) the encryption operation. **A cryptographic** protocol is a set of rules, between two or more partners, through to whom an authentication and / or transfer of key or message takes place. **A cryptographic system** is composed of three elements: encryption algorithm, keys and encryption key distribution protocol. **Decryption** is the inverse operation of encryption and it consists in applying the known encryption system. (In the presence of the correct key) on the cryptograms to find the clear message. Decryption is the operation by which, only based on the analysis of cryptograms made with a system of unknown cipher, highlights the clear message that was

encrypted and determines the characteristics of the cryptographic system used for encryption.

CHINESE REMAINDERS THEOREM-CRT

Theoretic statement

Let m_1, ..., m_k integers with (m_i, m_j) = 1 for every i ≠ j.

Then the system x ≡ a_i mod m_i has a unique solution modulo $\prod_{i=1}^{k} m_i$

Proof:

We will use the following notations:

M = $\prod_{i=1}^{k} m_i$ and

$M_i = \frac{M}{m_i}$

Since (m_i, m_j) = 1 for every i ≠ j we obtain that (M_j, m_j) = 1 for every value of j. This means that there exists N_j that satisfies the relation $M_j N_j$ = 1 mod m_j.

Then if we substitute x = $\sum_{i=0}^{k} a M_i N_i$ and reduce modulo m_i, we obtain :

x = $\sum_{i=0}^{k} a_j M_j N_j$ mod m_i for every value of i.

Using the fact that (M_i, m_j) ≠ 1 for i ≠ j we obtain:

x = $a_i M_i N_i$ mod m_i

x=a_i mod m_i for every value of i

The Uniqueness of the Solution : Consider x' and x'' two solutions of the equation then we obtain :

X = $x' - x''$ = 0 mod m_i for every value of i and thus x = 0 mod M.

Numerical Example

Solve the following equation system:

$x \equiv 3 \; mod \; 13$
$x \equiv 34 \; mod \; 47$
$x \equiv 2 \; mod \; 51$

BLOCK CIPHER CRYPTANALYSIS

Theoretical Summary

Since there is no universal mathematic formula which can be applied in cryptanalysis operations, we have chosen as a series of exercises for this chapter alterations of some typical algorithms of block ciphering. There are certain indications given, all of them preceded by a short description of the algorithm itself.

Numerical Examples

Study the following simplifications of the RC5 algorithm :

a. RC5 with 8 iterations and with no rotations
b. RC5 with 8 iterations and the number of total rotations is equal to the number of iterations

Study the following simplifications of DES algorithm :

a. DES with 12 iterations, but with no S applications
b. DES with 4 iterations
c. DES with 6 iterations

MERKLE-HELLMAN CIPHERING SYSTEM

Theoretical Summary

The Merkle-Hellman ciphering algorithm consists of coding a message under the form of the solution to a Knapsack problem of for which the multipliers $\{M_1, M_2, ..., M_n\}$ represent the cipher's key and the ciphered text $\sum_{i=1}^{n} b_i$ represents the clear text $\{b_1, b_2, ..., b_n\}$.

Definition. *A sequence of multipliers $\{M_1, M_2, ..., M_n\}$ can be defined as superincreasing if:*

$$M_k > \sum_{i=1}^{k-1} M_i$$

The superincreasing knapsack problem can be easily solved using the following schematic applied for k = n, ..., 1 *(it uses forms of the greedy algorithm)* :

If $M_k < S$ then $b_k = 1$ and $S = S - M_k$

Else $b_k = 0$

Knapsack type algorithms that are not superincreasing constitute difficult problems which do not have an optimal solution *(meaning fast solving algorithm)* . The only known

method of determining if $b_i = 1$ consists of verifying all of the possible solutions. All optimal algorithms have exponential complexity.

The Merkle-Hellman algorithm is based upon this propriety : the private key represents the sequence of multipliers to a superincreasing while the public key represents the sequence of multipliers to a knapsack with the same solution, but that is not superincreasing. Merkle and Hellman discovered a method of conversion from a superincreasing knapsack problem to a classic knapsack problem. The conversion technique employs the use of modular arithmetic.

Having available a superincreasing knapsack problem with $\{M_1, M_2, ..., M_n\}$ multipliers, it can be transformed to a classic knapsack problem with the help of the multiplier sequence :

$$\{m M_1 \bmod p,\ m M_2 \bmod p,\ ...,\ m M_n \bmod p\}$$

Where m and p are coprime natural numbers from the private key with the property that $p > \sum_{i=1}^{n} M_i$.

In order to cipher a binary message, it must be split into blocks of equal size of the multipliers sequence cardinal. The encryption of a block will be the natural number :

$$\sum_{i=1}^{n} b_i (m M_i \bmod p)$$

The receiver of the message has the information of the private key : the original multipliers and the values of m and p. The first step is to calculate $m^{-1} \bmod p$, then the encrypted text will be multiplied with the result and in order to finalize the decryption, the receiver will resolve the superincreasing knapsack problem.

Numerical example

a) Construct the public key of the Merkle-Hellman algorithm composed of the private key {2, 3, 6, 13, 27, 52}, the moduli p=105, and the multiplier m = 31. Encrypt the message 101110.

b) Decipher the message C = 4608 encrypted by the Merkle-Hellman algorithm using the following parameters: n=9, private key {1,2,5,10,19,40,98,179,355} , p=1717, m = 507.

PRIMES IN CRYPTOGRAPHY

The RSA algorithm

The RSA encryption algorithm is based n prime numbers. To generate the two keys, choose two random large numbers, p and q. For maximum security, choose p and q of equal length. Compute the product $n=pq$. Then randomly choose the encryption key, e, such that e and $(p-1)(q-1)$ are relatively prime. Finally, use the extended Euclidian algorithm to compute the decryption key, d, such that

$$ed=1 \bmod (p-1)(q-1).$$

In other words

$$d=e^{-1} \bmod ((p-1)(q-1)).$$

Note that d and n are also relative prime. The numbers e and n are the public key; the number d is the private key. The two primes, p and q, are no longer needed. They should be discarded, but never revealed.

To encrypt a message m, first divide it into numerical blocks smaller than n (with binary data, choose the largest power of 2 less than n). That is, if both p and q are 100 –digit primes, then n will have just fewer than 200 digits and each message block, m_i, should be just under 200 digits long. The encrypted message, c, will be made up of similarly sized message blocks, c_i, of about the same length. The *encryption formula* is simply: $c_i=m_i^e \bmod n$.

To *decrypt* a message, take each encrypted block c_i and compute: $m_i=c_i^e \bmod n$.

Since

$$(c_i)^d = (m_i^e)^d = m_i^{e\,d} = m_i^{k(p-1)(q-1)+1} = m_i m_i^{k(p-1)(q-1)} = m_i \bmod n.$$

In figure 3 we have a graphical view of using RSA for signing:

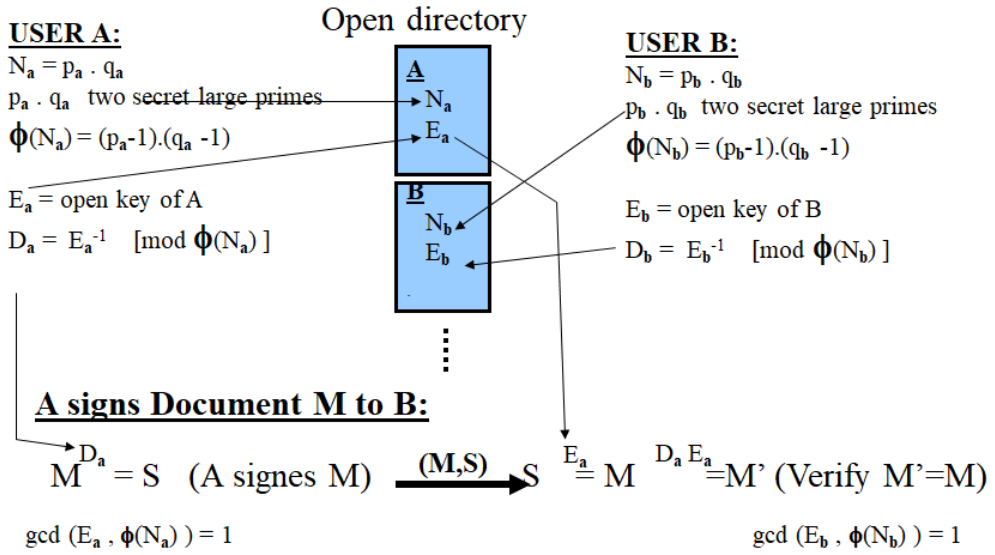

Figure 3

Remark. In order to avoid the classic methods of factorization p and q must be strong prime numbers. They must have properties:

i) p-1 has a large factor r

ii) p+1 has a large factor s

iii) r-1 has a large factor t

The signing operation of a message M is realized through the exponentiation of the print H(M) with the help of a private key s = H $(M)^d$ mod n. The verification of the signature is obtained through the comparison of H(M) with s^e mod n.

In practice, the value of e is relatively small, as a result d is a large number. This materializes into different running times between private operations and public ones.

The optimization of the verification of the signature can be deducted by using employing the Chinese Remainder Theorem (Lemma), yet this may introduce vulnerabilities within the system's implementation.

Thus, if p > q, we can precalculate the values.

dP = (e^{-1}mod n) mod (p – 1)

dQ = (e^{-1}mod n) mod (p – 1)

q*Inv*= q^{-1} mod p

In the calculation phase the following process takes place:

m_1=c^{dP} mod p

m_2=c^{dQ} mod q

h= q/nv (m_1 - m_2) mod p

m=m_2 + hq

The **Private key** is (p, q, dP, dQ, q/nv)

Numerical examples

1. The decryption of the message 01154 05746 04357 01154 using RSA (p=211 and q=167), using the public exponent e=2^8+1. The elements form the plain text are decoded using ACSII. N=35237. Phi(N)=34860, d=23873, the plain text message is TEST.
2. Given the number n = 36187829 with the property that it is the product of 2 numbers with the value of the product φ(n)= 36175776, factorize n.
3. Cipher the message M = 3 , utilizing the RSA algorithm with the following parameters: N=187 (cipher moduli), e =7 (cipher exponent).
4. Decipher the message C = 130 employing the RSA system with the following parameters: N= 187 = 17 * 11 (cipher moduli) and e = 7 (cipher exponent).
5. Decipher using CRT (Chinese Remainder Theorem/Lemma), the encrypted message c = 8363 knowing that p = 137, q = 131, n = p · q = 17947, e = 3, d = 11787.

GALOIS COMPUTATION

Theoretic statement

To meet current security requirements, cryptographic algorithms are based on new algebraic structures, including Galois calculations and elliptic curves. In the following we will present the notion of Galois body, used in the algebraic operations of the AES cryptographic standard. The Galois field is defined by a polynomial f(X)$\in Z_2[X]$ of degree n. The operations between two polynomials a(X)=a_0+a_1X+…+$a_n X^n$ and b(X)=b_0+b_1X+…+$b_n X^n$ from GF(2^n) are defined in the following way:

i) a(X)⊕b(X)= c(X) where c_i=a_i+b_i mod 2

ii) a(X)⊗b(X)= a(X)b(X) mod f(X)

A element from GF(2^n) can be represented in binary form thus his coefficients. The inverse of an element from GF(2^n) is determined by Euclid algorithm.

Numerical examples

1. What is the inverse of the element {45} represented in hexadecimal format defined within GF(2^8) polynomial f(X) = $1 + X + X^3 + X^4 + X^8$.

2. Sum the {53} and {83} elements from the GF(2^n) Galois Field defined by the polynomial $1 + X + X^3 + X^4 + X^8$.

3 Multiply the {53} and {83} elements from the GF(2^n) Galois Field defined by the polynomial $1 + X + X^3 + X^4 + X^8$.

ELGAMAL ENCRYPTION

The encryption algorithm

The El Gamal encryption algorithm is defined by a prime number p and $g \in Z^*_p$ a primitive element called generator. For the private key $x \in Z^*_p$ is computed $y=g^x \bmod p$, the public key is $\{y,g,p\}$. For encrypt a message $M \in Z^*_p$ we choose a random $k \in Z^*_{p-1}$, the encrypted text is $(y_1y_2)=(g^k \bmod p, My^k \bmod p)$.

For decrypting the message (y_1,y_2) is computed $y_2(y^{x_1})^{-1} \bmod p$.

A numerical example

The encryption of the message $M=4$ using the ElGamal algorithm with the parameters $p=17$, $g=14$ and $x=2$ is $\{6,8\}$.

ELGAMAL ENCRYPTION SYSTEM BASED ON ELLIPTIC CURVES

The encryption algorithm

The El Gamal algorithm can be extended to any finite group $(G, *)$ for which the discreet logarithm problem is difficult, for the points from an elliptic curve.

Thus, let us consider $\alpha \in G$ for which the discreet log problem is difficult in the subgroup $H=\{\alpha^i|, i \geq 0\}$.

Based on a private key $x \in Z$, we construct $\beta=\alpha^x$, and the public key is $\{G,\alpha,\beta\}$.

To encrypt a message M we choose randomly $k \in Z_{|H|}$:

$$E(M,k)=(\alpha^k, M^* \beta^k)$$

The plain text message M is recovering form the encrypted message (y_1,y_2) using the formula:

$$y_2*(y^{x}{}_1)^{-1}=M*\beta^k*((\alpha^k)^x)^{-1}=M*(\alpha^{kx})^{-1}=M.$$

A numerical example

The El Gamal encryption of the plain text message *(10,9)* using the elliptic curve E*:* $y^2=x^3+x+6 \bmod 11$ is *((10,2),(3,5)).*

MENEZES-VANSTONE

The encryption algorithm

In this encryption system -in fact a variant of El Gamal- the elliptic curve is used for masking, the plain texts and the encrypted texts can be formed from every nonzero elements (it is not necessary to be points from *E*).

Let E be an elliptic curve over Z_p, p>3 a prime number which contains a cyclic subgroup of G in which the problem of the discreet logarithm is difficult. Based on the private key $d \in Z$, we construct $\beta=d\alpha$, the public key being *{E,α,β}.*

For encrypting a message $m=(m_1,m_2) \in Z^*_p \times Z^*_p$ we select random value *k* and construct the encrypted text (y_0, y_1, y_2) after the rules:

$$y_0=k\alpha,\ (c_1,c_2)=k\beta,\ y_i=c_i m_i,\ i=1,2.$$

For decrypting, if we know (y_0, y_1, y_2) and the private key *d* we find the plain text:

$$(m_1,m_2)=(y_1 c^{-1}{}_1 \bmod p,\ y_2 c^{-1}{}_2 \bmod p),$$ where $dy_0=(c_1, c_2).$

A numerical example

Let us consider the Menezez-Vanstone algorithm specified by the following parameters: the elliptic curve E: $y^2=x^3+x+6 \bmod 13$. Show that $\alpha=(4,3)$ is a generator of the group E. Encrypt the message (3,7) with the random value k=4.

After computations we obtain the encrypted text *((9,4),12,5).*

EL GAMAL SIGNATURE

The signature algorithm

Let *p* be a prime number for which the discreet problem in Z_p is difficult and $\alpha \in Z^*_p$ a primitive element. The public key is derived from the private key*:* $\beta=\alpha^a \bmod p.$ The signature of a message x is computed using the random value (secret) $k \in Z_{p-1}$, is defined by (γ, δ) where:

$$\gamma = \alpha^k \bmod p \text{ and } \delta=((H(x)-a\gamma)k^{-1} \bmod (p-1),$$

H(.) is a hash function *(H(x)=x if is not specified the hash function).*

The signature (γ, δ) of the message *x* is valid if:

$$\beta^{\gamma}\gamma^{\delta}=\alpha^{H(x)} \bmod p$$

A numerical example

Sign the message x=101 with the El Gamal algorithm specified by the following parameters: $p=467$, $\alpha=2$, the private key $a=127$ and the random variable $k=213$. After the computation we obtain the signature: $(29,16)$.

DIFFIE-HELLMAN KEY EXCHANGE

The protocol

Let p be a prime number, q a prime divisor of $p-1$ and $\alpha \in Z^*_p$ an element of order q. The Diffie-Hellman protocol returns a session key K:

Step 1. A randomly generates $a \in Z^*_q$ and sends to B the value $R_A=\alpha^a \bmod p$.

Step 2. B randomly generates $b \in Z^*_q$ and sends to A the value $R_B=\alpha^b \bmod p$.

Step 3. A computes $K=K_{A,B}=R^a_B = \alpha^{ab}$

Step 4. A computes $K=K_{B,A}=R^b_A = \alpha^{ab}$

A numerical example

Find the key derived from Diffie-Hellman protocol with the parameters: $p=25307$, $\alpha=2$, $a=2009$, $b=2010$

$K=21554$.

SUBLIMINAL CHANNELS

The El Gamal subliminal channel

In the El Gamal authentication system A choose a prime number q and a primitive element $\alpha \in Z_q$. The values q and α are public. Using a secure channel, A and B, establish a prime $p \in Z_q$ The protocol used by A to send a subliminal message $y \in Z_q$ to B using the plain text x is the following:

Step 1 . A computes $\beta=\alpha^y \bmod q$.

Step 2. Find γ like a solution of the equation $x=p\beta+y\gamma \bmod (q-1)$.

Step 3. A sends to B the triplet (x, β, γ).

Step 4. B computes $a=(\alpha^p)^{\beta}\beta^{\gamma} \bmod q$.

Step 5. *If* $a=\alpha^x \bmod q$ then B decides that the message is authentic.

Step 6. B recovers the subliminal message: $y=(x-p\beta)\gamma^{-1} \bmod (q-1)$.

A numerical example

Consider the subliminal channel El Gamal specified by *q=11 and α=2.* Suppose that we want to transmit the message *y=9* using the secret key *k=0* and cipher text *x=5.* The message send thus the communication channel will be *(5,6,5).*

SIDE CHANELL ATTACKS

Attacks in the deployment environment involve a series of hardware measurements on cryptographic mode:

Attacks by measuring execution time. By measuring the time required to perform private key operations, the attacker can determine the exponents used in the Diffie-Hellman protocol, the RSA factor (especially on the RSA algorithm used for Chinese motto of CRT scraps), as well as a number of other cryptographic systems such as the DSS digital signature algorithm.

Attacks by measuring the power consumed. The attack with the help of simple power analysis

(SPA) is the measurement of the power consumed by the device during the cryptographic operation.

This type of attack usually applies to devices with an external voltage source (such as example smart cards). Power consumption depends on the instruction executed. So, by monitoring the power consumption, the sequence of instructions (source code) can be deduced. If the instruction sequence depends on the length of the key, then power consumption can give key information. In most processors, the power pattern consumed by a instruction also depends on the value of the operands (for example, setting a bit in a register consumes more energy than deleting it). Measurements performed on many inputs can deduce the value of the operand. The technique is called **differential analysis a power (DPA)**.

Attacks with hardware failures (errors). Hardware can cause errors (transient, latent, or induced) during arithmetic operations. Throughthe rational exploitation of these errors can recover the private key for the algorithms of RSA sign and Rabin. A number of cryptographic protocols such as Fiat-Schamir and Schnorr can be broken by judicious use of the results of these errors.

Differential Failure Analysis (DFA) is a the scheme used to recover the secret keys of a cryptographic system from a physically secure Hardware Security Module (HSM) device. The defect model is that of transient (random) defects ¸and of induced defects. The method is used for identification keys in the case of the use of known ciphers (eg DES) and / or ciphers with unknown algorithm or to the reconstruction of the algorithm (with a known structure).

SOFTWARE RESOURCES

CRYPTOOL

CrypTool is a software package dedicated to the simulation and analysis of cryptological mechanisms In an illustrative way. From the initial role of training in the field of personnel security to various private companies, CrypTool has evolved into an open educational project source with applications in the field of cryptography and most related fields. The product primarily aimed at students of the faculties of mathematics and computer science, of the companies that active in the field of information security as well as application developers or computer users in general who wish to acquire the minimum baggage of cryptographic knowledge.

The product is currently free and available in several versions, the first of which being CrypTool 1.4.x fully developed in the C ++ environment. It later expanded into two other versions, still in beta, that use the latest development standards generation being in a continuous update. Thus, in July 2008, CryptTool was launched 2.0 developed in the C # environment, a version that provides a wider range of features combined with a graphical interface with drag-and-drop facilities. At the beginning of 2010 The JCrypTool version developed in the Java environment has been launched, the advantages of this version being that it is independent of the platform on which it runs (Windows, Linux, Mac) and that it uses the powerful FlexiProvider tool that makes it easy to load modules cryptographic in any application built over JCA (Java Cryptography Architecture).

CrypTool was developed in collaboration with educational institutions thus becoming a software educational and a good introductory tool in the field of cryptology, currently being used successfully in many prestigious universities. Due to the easy handling of the mechanisms cryptographic as well as visualization and presentation in an easy and unique way of the results, CrypTool can represent the practical component of the theoretical

courses in the field of cryptology as well as a quick method of getting acquainted with its essential components field.

The product covers both branches of cryptology, namely cryptography and cryptanalysis. There are implemented facilities within each subdomain as follows:

• classical cryptography: Caesar ciphers, monoalphabetic substitution, homophonic substitution, Vigenere, Hill, Playfair, ADFGVX, Addition, XOR, Vernam, Solitaire etc;

• modern symmetric cryptography: IDEA, RC2, RC4, DES, 3DES, DESX ciphers as well as all finalists of the AES figure, namely MARS, RC6, Rijndael, Serpent and Twofish;

• asymmetric cryptography: RSA;

• hybrid cryptography: data encryption performed with symmetric algorithms (AES), protection the encryption key being secured by asymmetric methods (RSA);

• digital signatures: RSA, DSA, ECDSA (Elliptic Curve Digital Signature Algorithm), Nyberg-Rueppel;

• hash functions: MD2, MD4, MD5, SHA, SHA-1, SHA-2, RIPEMD-160;

• random generators: secude, x^2 mod n, LCG (linear congruence generator), ICG (inverse congruence generator).

In cryptanalysis, most standard attacks are implemented afterwards as follows:

• attack with ciphertext: Caesar, Vigen`ere, Addition, XOR, Substitution, Playfair;

• attack with clear text: Hill, Single-column transposition;

• manual attack: mono alphabetic substitution, Playfair, ADFGVX, Solitaire;

• brute force attack: for all algorithms; either the entropy of the clear text is assumed is small or the key is partially known or the alphabet of the clear text is known;

• attacks on RSA: based on factorization or techniques that use algebraic structures;

• attacks on hybrid systems: attacks on RSA or AES (side channels attacks);

• attacks on digital signatures: RSA by factoring; viable to the length of 250 bits (i.e. 75 digits);

• attacks on hash functions: generating collisions ASCII texts with the birthday paradox;

• randomization analysis: FIPS-PUB-140-1 test battery, periodicity, Vitany, entropy, histograms, autocorrelations, ZIP compression test, etc.

In support of users, CrypTool has implemented a series of demos and animations which exemplifies various facilities that the product offers using cryptographic primitives supported and implemented in the application such as Caesar, Vigilance, Nihilist, DES (all four with ANIMAL), Enigma (Flash), Rijdael / AES (Flash and Java), hybrid encryption and decryption (AES-RSA and AES-ECC), signature generation and verification digital, Diffie-Hellman key exchange protocol, secret sharing (CRT or Shamir), challenge-response method, side-channel attacks, e-mail security via the S / MIME protocol (Java and Flash), 3D graphical presentations for (pseudo) random data, sensitivity of hash functions to changes in clear text, number theory and crypto RSA system (Authorware).

CrypTool also contains an interactive educational module dedicated to cryptographic applications which requires elementary aspects of number theory called "NT". This module introduce the user in elementary problems of number theory such as Euclid's algorithm for finding the greatest common divisor, the Fermat primality test, the Fermat factorization, factoring Pollard Rho and others. Another advantage of the CrypTool product is the existence of a documentation menu and an online extension of it containing in addition explanations on general notions of cryptography, a chronology on the development of the field, examples of the use of application facilities, index sorted by cryptographic topics and the list of references.

The fact that the software package is open source, that it covers aspects related to both classical and modern cryptography, the multiple ways of original simulation and visualization, as well as the easy way of application and analysis of cryptographic mechanisms lead us to the conclusion that the CrypTool package is both a quick way to get started in the field of cryptography and a powerful tool for specialists to study and the application in the same environment of various concrete problems that may occur in cryptography and cryptanalysis.

OPENSSL

OpenSSL is a suite of applications that implement Secure Sockets Layer protocols (SSL v2 / v3) and Transport Layer Security (TLS v1) as well as a dedicated library covering a wide range of cryptographic primitives. The project is managed by a community of volunteers from around the world who communicate, using the Internet, for planning and development of the OpenSSL toolkit as well as the related documentation.

OpenSSL is based on the SSLeavy library developed by Eric A. Young and Tim J. Hudson, project completed at the end of 1998. The product has a double licensing, both the OpenSSL library as well as the original SSLeavy library. Both types of licenses are of the type Open-source BSD, so the toolkit can be used for both commercial and non-commercial purposes. The software package uses powerful cryptographic tools, being developed continuously distributed legally by several European countries, but subject to restrictions on import / export and use in some countries of the world.

OpenSSL is available in many versions being in a continuous development, bugs being often signaled and corrected. The current stable version is OpenSSL 0.9.8m being available from February 2010; in addition, users benefit from permanent online access to study developments subsequent to the latest stable version. The versions are available for most UNIX operating systems (including Solaris, Linux, Mac OS X and the four open source BSD operating systems), Open VMS and Microsoft Windows.

OpenSSL implements the SSL and TSL protocols. Transport Layer Security (TLS) si its predecessor Secure Sockets Layer (SSL), are cryptographic protocols that provide security of communications over networks similar to the Internet. The two protocols allow client / server applications to communicate securely. TLS provides endpoint authentication as well as the confidentiality of communications over the Internet using RSA security supporting keystroke lengths at 2048 bits. Protocols are used for navigation on the Internet, e-mail, voice-over-IP (VoIP), etc.

The OpenSSL cryptographic library implements a wide range of algorithms used in various standards used on the Internet. The facilities provided by this library are used to implement SSL, TLS, and S / MIME, as well as SSH, OpenPGP, and other cryptographic standards. The library has implemented a variety of cryptographic and other primitives facilitated as follows:

• Symmetric encryption algorithms: Blowfish, CAST, DES, IDEA, RC2, RC4, RC5;

• Asymmetric encryption algorithms: RSA (based on large number factorization), DSA (based on the discrete logarithm problem), EC (elliptic curves) Diffie-Hellman key exchange;

• Digital certificates: X509, X509v3;

• Hash functions and authentication codes: HMAC, MD2, MD4, MD5, MDC2, RIPEMD, SHA;

• Input and output control functions, data encoding functions: PKCS7, PKCS12, ASN1, BIO, EVP, PEM.

The OpenSSL utility is a command line tool used to manage various functions cryptographic from the OpenSSL library. It can be used for:

• Creation and management of private keys, public keys and parameters;

• Operations involving cryptography with public keys;

• Create X.509 certificates, CSRs and CRLs;

• Calculation of message summaries;

• Encryption and decryption using various ciphers;

• client / server testing (SSL / TLS);

• Mail signatures and encryption (S / MIME);

• Temporary trademark applications, generations and checks.

OpenSSL is one of the few open source projects subject to compliance validation with the FIPS 140-2 standard, used in computer security, developed by National Institute of Standards and Technology (NIST). The software package itself is not validated, being developed a software component of it called OpenSSL FIPS Object Module, this being compatible with OpenSSL being created to offer the possibility of products that use OpenSSL APIs to be subject to FIPS 140-2 compliance validation. in January 2006 this component was certified, but it was revoked in July 2006 due to some doubts regarding the validity of the interaction of the module with external software. In February 2007 the product was recertified.

FIPS Module OpenSSL validation is unique among all FIPS 140-2 validations by the fact that the manufacturer provides the entire source code. Therefore, used without no modification and built on any platform according to the documentation provided a validated cryptographic module is obtained directly. Any minor change to the code involves the need for revalidation, an expensive process (approximately $ 50,000) and a long 6 and 12 months). The latest open source validation is OpenSSL FIPS Object Module (Software Version: 1.2), FIPS 140-2 certificate # 1051. There is currently no other product open source subject to FIPS 140-2 validation due to funding gap. The previous ones were funded by the

commercial sector and government sponsors, some of them they prefer to remain anonymous.

“The European Commission support for the production of this publication does not constitute an endorsement of the contents which reflects the views only of the authors, and the National Agency and Commission cannot be held responsible for any use which may be made of the information contained therein”.