METHOD FOR SECURITY ENHANCEMENT OF AUDIO INFORMATION IN COMMUNICATION MULTIMEDIA SYSTEMS AND NETWORKS APPLYING ENCRYPTION ALGORITHM WITH PUBLIC KEY

S. Pleshkova-Bekiarska, D. Kinanev

Department of Telecommunications,
Technical University - Sofia, Kliment Ohridski 8, Sofia

snegpl@tu-sofia.bg

Abstract

In this article it will be analyzed the characteristics of the RSA algorithms for generation of public and private keys used by the cryptosystems in order to ensure the confidentiality, authenticity and integrity of the audio information exchanged between different communication points. For that purpose it will be used the resources provided by the Java JDK 1.8.0_71 programming language with which practically will be shown the advantages and disadvantages of this algorithm.

Keywords: *public, private, key, encryption, java, audio*

1. INTRODUCTION

The security of the information [1] in modern days is extremely important in different aspects, governmental information, business documents, physical security, etc. This is multidisciplinary area of study and professional activities which are concerned with the development and implementation of security mechanisms based on the four principle of the information security: Confidentiality, Integrity, Availability, Non-repudiation.

Confidentiality is the property, that information is not made available or disclosed to unauthorized individuals.

Data integrity means maintaining and assuring the accuracy and completeness of data over its entire life-cycle. This means that data cannot be modified in an unauthorized or undetected manner.

For any information system to serve its purpose, the information must be available when it is needed. This means that the computing systems used to store and process the information, the security controls used to protect it, and the communication channels used to access it must be functioning correctly.

In law, non-repudiation implies one's intention to fulfill their obligations to a contract. It also implies that one party of a transaction cannot deny having received a transaction nor can the other party deny having sent a transaction. It is important to note that while technology such as cryptographic systems can assist in non-repudiation efforts, the concept is at its core a legal concept transcending the realm of technology.

In general the algorithms used for encryption of information are symmetric [2] and asymmetric [3]. Symmetric algorithms are algorithms for cryptography that use the same cryptographic keys for both encryption and decryption. Examples are AES, DES, Blowfish. The main disadvantage of this type of encryption algorithms is the same key is used from the both parties which provides low security of the encrypted information.

In asymmetric algorithms the cryptographic keys are paired and encryption performed with one key, called public, can be decrypted only by the other member, called private key of the pair. Examples are RSA, DSA, Diffie–Hellman. As most common used is the RSA algorithm which unlike the other similar algorithms, it provides all the four key principles of the information security.

Considering the fact that the RSA algorithm is most commonly used, the subject of this article will be developing method to analyze what would be the effect of applying the RSA algorithm in the encryption of audio information and respectively analyzing what would be the resulted key parameters which are important in the implementing such feature in communication multimedia system.

2. BRIEF OVERVIEW OF METHODS OF ENCRYPTION WITH PUBLIC KEY

The RSA algorithm [4] has been invented by Ron Rivest, Adi Shamir, and Leonard Adleman in 1978. It implements a public-key cryptosystem, as well as digital signatures. In RSA, encryption keys are public, while the decryption keys are kept in secret. The RSA algorithm involves four steps: key generation, key distribution, encryption and decryption.

By default the RSA algorithm uses PKCS#1 standard [6] which defines the mathematical definitions and properties that RSA public and private keys must have.

The standard defines several basic primitives. The primitive operations provide the fundamental instructions for turning the raw mathematical formulas into computable algorithms.

- I2OSP, OS2IP: conversion between the potentially large nonnegative integers encountered in the mathematical formulas and their computer data representation as a sequence of bytes (an octet string).
- RSAEP, RSADP: basic encryption and decryption algorithms.
- RSASP1, RSAVP1: algorithms for producing and verifying signatures.

By default in PKCS#1 it is used RSAES-PKCS1-v1_5 scheme for encryption and decryption. RSAES-PKCS1-v1_5 combines RSAEP and RSADP primitives and can operate on messages of length up to k-11 octets (k is the octet length of the RSA modulus), preventing the attacks on low-exponent RSA which is the reason why the block size is kept small. These attacks works when similar messages are encrypted with the same RSA public key. When the block size of information is too large, it is possible to be recovered the key parameters of the private key.

The Digital Signature Algorithm (DSA) is a Federal Information Processing Standard for digital signatures. It was proposed by the National Institute of Standards and Technology (NIST) in August 1991 [5]. Key generation has two phases. The first phase is a choice of algorithm parameters which may be shared between different users of the system, while the second phase computes public and private keys for a single user.

3. DEVELOPMENT OF PRACTICAL ANALYZE METHOD OF RSA ENCRYPTION ALGORITHM APPLIED FOR SECURITY OF AUDIO INFORMATION

The following diagram illustrates the analyze method of the RSA algorithm used for encryption of audio file with .mp3 extension.

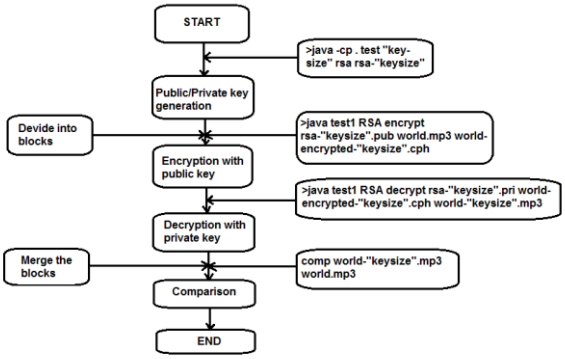

Fig. 1. Workflow diagram

With the command >java -cp . test "key-size" rsa rsa-"keysize" it is triggered the generation of the key pair as it is pointed what to be the key size, what is the used algorithm and what will be the name of the files where the public and the private keys will be stored.

With the command >java test1 RSA encrypt rsa-"keysize".pub world.mp3 world-encrypted-"keysize".cph the program will encrypt file "world.mp3" with RSA algorithm and publik key "rsa-keysize.pub". If the file is longer than 53 bytes then the program will divide the file into blocks, each 53 bytes long, it will encrypt the blocks and then will produces the encrypted file "world-encrypted-keysize.cph".

In order to decrypt the produces file it is used the command >java test1 RSA decrypt rsa-"keysize".pri world-encrypted-"keysize".cph world-"keysize.mp3", where the program is pointed to use "rsa key-size.pri" to decrypt file "world-encrypted-keysize.cph" and produce a result file called "world-keysize.mp3".

At the next step the program will compare whether the original file "world.mp3" matches the produces decrypted file "world-keysize.mp3".

Public/Private key generation

- Choose two distinct prime numbers p and q.

For security purposes, the integers p and q should be chosen at random, and should be similar in magnitude but 'differ in length by a few digits to make factoring harder. Prime integers can be efficiently found using a primality test.

- Compute

$$n = p * q\,, \tag{1}$$

where n is used as the modulus for both the public and private keys. Its length, usually expressed in bits, is the key length.

- Compute

$$\varphi(n) = \varphi(p)\varphi(q) = (p-1)(q-1) = n - (p+q-1), \tag{2}$$

where φ is Euler's totient function. This value is kept private.

- Choose an integer e such that

$$1 < e < \varphi(n) \text{ and } \gcd(e, \varphi(n)) = 1\,, \tag{3}$$

where e and φ(n) are coprime.

- Determine d as

$$d \equiv e - 1 (\mathrm{mod}\, \varphi(n))\,, \tag{4}$$

where d is the modular multiplicative inverse of e (modulo φ(n))

This is more clearly stated as: solve for d given

$$d * e \equiv 1 (\mathrm{mod}\, \varphi(n))\,, \tag{5}$$

e having a short bit-length and small Hamming weight results in more efficient encryption – most commonly 216 + 1 = 65,537. However, much smaller values of e (such as 3) have been shown to be less secure in some settings.

e is released as the public key exponent.

d is kept as the private key exponent.

The public key consists of the modulus n and the public (or encryption) exponent e. The private key consists of the modulus n and the private (or decryption) exponent d, which must be kept secret. p, q, and φ(n) must also be kept secret because they can be used to calculate d.

Encryption with public key

The following function is used in order to produce the encrypted information:

$$c \equiv m^e \bmod n\,, \tag{6}$$

where c is the resulted encrypted information, e is the public key, m is the messaged.

Decryption with private key. Result file world-"key".mp3

$$c^d \equiv (m^e)^d \equiv m \bmod n\,, \tag{7}$$

where is the exponent of the private key.

Comparison between resulted and original file – At this steps the decrypted file world-"key".mp3 will be compared with the original file.

4. EXPERIMENTAL RESULTS FROM TESTING ENCRYPTION ALGORITHM WITH THE PUBLIC KEY FOR SECURITY OF AUDIO INFORMATION

The tests has been completed on computer with the following parameters:

Processor: Intel(R) Core(TM) i5-4300U CPU @ 1.90 Ghz 2.50Ghz; Installed memory (RAM): 8 GB; System type: 64-bit; Operating System: Windows 7

Java library: JDK 1.8.0_71

Generating of public and private keys.

The question about what should be the key length is very important considering the high security these keys needs to provide.

The main assumption which reflects directly on the choice of key length is the so called factoring of the private key. In general this is the revers process of the key generation in which from the known parameters and known mathematical operations is trying to determine the secret kept parameters involved in the key generation. This method is commonly used from the hackers in order to found out the private key and compromise the information security.

In this test will be generated keys which are 512, 1024, 2048 and 4096 bits long. For that purpose it will be used the predefined functions and classes provided from Java and the source code in the Attachment 1. The public key will be stored in file with X.509 encoded format and name "rsa-size.pub". The private key will be saved in file with PKCS#8 encoded format with name "rsa-size.pri".

Encryption of a test file using the generated public key.

As a start of this test file "test.mp3" with size 3.34 MB will be used. The program returns an error "Data must not be longer than 53 bytes". This is because the RSA algorithm uses PKCS1 padding schema, which can only encrypt 53 bytes at a time (This is the size of one block of data) if your key size is 64 bytes (512 bits). It`s visible that for the different size of the key, this amount changes. For key 1024 bits (128bytes) long, it is 117 bytes, for 2048 bits (256 bytes) it is 245 bytes, for 4096 bits (512 bytes) it is 501 bytes.

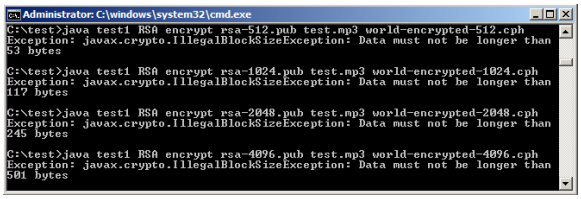

Fig. 2. Encryption of a large test file with the public key

For the purpose of this analyze an audio file called "world.mp3" with size 53 bytes will be used.

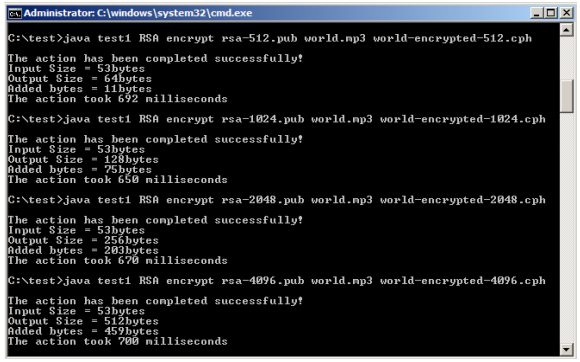

Fig. 3. Encryption of a test file which is 53 bytes long

From the test it could be seen that with 512 bit (64 bytes) key length, the encrypted file is 64 bytes, 1024 bits (128 bytes), it is 128 bytes, 2048 bits (256 bytes), it is 256 bytes, 4096 bits (512 bytes), it is 512 bytes.

Decryption of the test file using the generated private key.

In order the encrypted files to be decrypted, the corresponding private key needs to be used.

The same program will be used for the decryption of the files, but with the following command:

>java test1 RSA decrypt rsa-512.pri world-encrypted-512.cph world-512.mp3

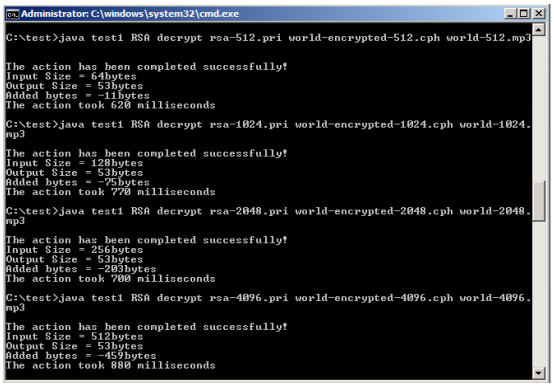

Fig. 4. Decryption of the encrypted file with the private key.

The decrypted files and original file will be compared whether they match:

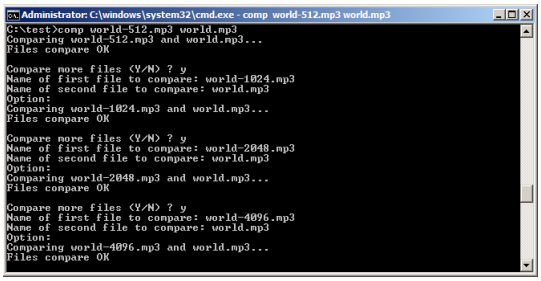

Fig. 5. Comparison between the decrypted and original files

After the check the program returns positive comparison.

5. CONCLUSION

From the tests it could be seen that the RSA algorithm can encrypt only one block information at a time which is 53 bytes long. This is because by default RSA using PKCS#1 padding scheme. So if there is a need to encrypt larger files it should be developed additional feature which will divide the file into 53 bytes long packages because by default the RSA algorithm doesn`t include such a possibility.

From the carried out tests it is visible that by the RSA algorithm when one block information is encrypted, the produced encrypted file is long as the used public key.

For the estimated times of the operations it could be concluded that

$$T_{ENC} = T_c \ , \qquad (8)$$

where T_c is the time for which it will be calculated expression (6).

$$T_{DEC} = T_{C^d}, \quad (9)$$

where T_{C^d} is the time for which it will be calculated expression (7).

Usually the time for decryption T_{C^d} is greater than the time for encryption T_c which also can be seen from the measured times.

During the tests the average measured times is 700 milliseconds.

As future development of this analyze method, the test could be carried out on a device which has smaller processor or memory, like mobile device or minicomputer where the time of encryption and decryption could be isolated and more precisely and objectively analyzed.

Also as development of the method, it could be proposed approach to be avoided the RSA limitation of the block size of information which could be encrypted at a time as well as functions in the program code which could divide the larger files into smaller packages.

Acknowlegment

This paper was supported by Technical University – Sofia inner program to research projects under 161ZF0003-07: "Methods and tools for the study of audio signals".

References

[1] "NIST Special Publication 800-27 Rev A". csrc.nist.gov.

[2] Pelzl & Paar (2010). Understanding Cryptography. Berlin: Springer-Verlag. p. 30

[3] Stallings, William (1999-01-01). Cryptography and Network Security: Principles and Practice. Prentice Hall. p. 165. ISBN 9780138690175.

[4] Rivest, R.; Shamir, A.; Adleman, L. (February 1978). "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems". Communications of the ACM 21 (2): 120–126. doi:10.1145/359340.359342.

[5] "FIPS PUB 186]: Digital Signature Standard (DSS), 1994-05-19". csrc.nist.gov.

[6] PKCS#1 v2.2: RSA Cryptography standard, RSA Laboratories, October 27, 2012.