CRYPTOGRAPHIC RISK IN THE USAGE OF RSA ALGORITHM

Emil SIMION
e-mail: esimion@fmi.unibuc.ro

Abstract. *The paper presents some vulnerability that can appears in the improper usage of public key encryption algorithm RSA (Rivest, Shamir, Adellman).*

1. INTRODUCTION

The concept of public-key cryptography was invented by Whitfield Diffie and Martin Hellman, and independently by Raplhy Merkle. 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. Usually there is a connection between the public and the private key. This paper will present in section 2 the RSA public key encryption algorithm and in section 3 same "bad implementations" of the encryption method, which can conduct to derive the private key from the public key. We also present, in section 4, some tricks, used in generation of a public/private key, to enable a smart brute force attack.

2. RSA ALGORITHM

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.

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:

$$c_i=m_i^e \mod n.$$

To *decrypt* a message, take each encrypted block c_i and compute:

$$m_i=c_i^e \mod n.$$

Since

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

In hardware, RSA is about 1000 times slower then DES. In software, DES is about 100 faster than RSA. RSA encryption goes much faster if we choose smart values of the encryption exponent e. The three most common choices are *3* (recommended by standards PEM and PKCS#1), *17* and *65537* (recommended by standards X.509 and PKCS#1). There are no security problems with using any of these values for e (assuming you pad messages with random values), even if a whole group of users uses the same value for e.

Private key operations can be speeded up with the Chinese remainder theorem if you save the values of p and q, and additional values such as d mod ($p-1$), d mod ($q-1$), and q^{-1} mod p. The additional numbers can easily be calculated from the private and public keys.

3. ATTACKS ON RSA ALGORITHM

3.1 Chosen ciphertext attack

Some attacks work against the implementation of RSA. These are not attacks against the basic algorithm, but against the protocol. It's important to realize that it's not enough to use RSA. This attack is presented in Schneier [8] and can be avoided if we use a one-way hash function before signing a document.

3.2 Common Modulus Attack on RSA

A possible RSA implementation gives everyone the same *n*, but different values for the exponents *e* and *d*. Unfortunately, this doesn't work. The most obvious problem is that if the same message is never encrypted with two different exponents (both having the same modulus), and those two exponents are relative prime (which they generally would be), then the plain text can be recovered without either of the decryption exponents. This attack is presented in Schneier [8] and is feasible if we use a common *n* among a group of users.

3.3 Low encryption exponent attack against RSA

RSA encryption and signature verification are faster if we use a low value for e, but that can also be insecure. If you encrypt $e(e+1)/2$ linearly dependent messages with different public keys having the same value of e, there is an attack against the system. If there are fewer than that many messages, or if the messages are unrelated, there is no problem. If the messages are identical, then *e* messages are enough. The easiest solution is to pad messages with independent random values. Most real-world RSA implementations –PEM and PGP for example –do this.

To avoid this kind of attack the messages must be padded with random values before encrypting them; make sure that m is about the same size as n.

3.4 Low decryption exponent attack against RSA

Another attack, this one by Michael Wiener, will recover d, when d is up to one quarter the size of n and e is less than n. This rarely occurs if e and d are chosen at random, and cannot occur if e has small value. This attack can be avoided if we chose a large value for d.

3.5 Attack on encryption and signing with RSA

It makes sense to sign a message before encryption it, but not everyone follows this practice. With RSA, there is an attack against protocols that encrypt before signing. This attack is presented in Schneier [8].

3.6 Attack in case of small difference between prime numbers p and q

In the situation that the prime numbers p and q are close one to each other the following remark allows us to develop an efficient searching algorithm:

$$\left(\frac{p+q}{2}\right)^2 - n = \left(\frac{p-q}{2}\right)^2.$$

For the factorization of n we must test all integers $x > \sqrt{n}$ for which $x^2 - n$ is perfect square; let be this integer denoted by y. We can write

$$\begin{cases} p = x + y \\ q = x - y. \end{cases}$$

Thus the numbers p and q must be of large enough.

3.7 Hardware attack

Hardware attacks exploits some hardware parameters such running time, simple power analysis (SPA), differential power analysis (DFA) and fault analysis (FA).

Examples of hardware attacks are:

-*Timings attacks*: depending the running time we can predict which of the bits from the key are zero and which of the bits of the key are one;

-*SPA attack*: depending the consume power of the cryptographic engine and using adequate math we can derive the key bits. This attack is suitable for crypto devices with external power supply such as smart cards, using the consumed power we can recover the code source from the inside of the smart card;

-*DPA attack*: in the most cases of microprocessors the consumed power depends on the value of the operand (for example erasing a bit requires a less power then setting a bit), measuring different inputs we can deduce the value of the operand;

-*Fault analysis*: we can induce same faults in the processor computations and using some math we can derive the key bits.

4. INCLUDING TRAP INFORMATION INTO RSA EXPONENTS

The developer of an RSA cryptosystem may include traps at the key generation level. Obviously the reveal of this trap information may compromise the image of the cryptographic provider. Will present some methods of including such trap information:

- hiding the low private exponent *d* using a permutation function of the odd numbers smaller then *n* (the encryption modulus);
- hiding the low public exponent *e* and some information about the private exponent *d* using a permutation function of the odd numbers smaller then *n* (the encryption modulus);
- hiding the prime number *p* in the product *n=pq*;

For each of the hiding methods mentioned above there are detection test.

5. REFERENCES

[1] IEEE P1363/D13, *Standard Specification for Public Key Cryptography*, Draft Version 13, November 1999.

[2] Koblitz N., *Elliptic Curve Cryptosystems*, Mathematics of Computations, vol. 48, nr.177, 1987, pag. 203-209.

[3] Koblitz N., *Introduction to Elliptic Curves and Modular Forms*, Springer-Verlag, 1984.

[4] Koblitz N., *A Course in Number Theory and Cryptography*, Springer-Verlag, 1988.

[5] Koblitz N., *Algebric aspects of Cryptography*, Springer-Verlag, 1999.

[6] Salomaa A., *Public Key Cryptography*, Springer-Verlag, 1989.

[7] Schroeder M.R., *Number Theory in Science and Communications*, Third Edition, Springer Verlag, 1997.

[8] Schneier B., *Applied Cryptography*, John Wiley&Sons, 1996.

[9] Simion E., *Teoria deciziilor, Cercetări Operaținale şi Criptologie*, Ed. Politehnica Press, 2001.

[10] Simion E., *Criptanaliza. Rezultate şi Tehnici Matematice*, Ed. Univ. Bucureşti, 2004.

[11] Welschenbach M., *Cryptography in C and C++*, APress, 2001.