UDC 681.3.67

C. Marinescu, N. Ţăpuş

AVOIDING PITFALLS IN REAL LIFE RSA IMPLEMENTATIONS

University politehnica of Bucharest, Romania

Introduction

The RSA algorithm is used on a large scale for securing WWW traffic, e-mail, and some wireless devices. The elegance and the apparent simplicity of the RSA algorithm are due to the algebraic framework in which the algorithm was defined. But an improper use of this framework becomes dangerous, threatening the integrity and security of RSA when used either for encryption or for digital signatures. From a software developer's perspective, this is also a reason why a robust and correct implementation of the algorithm that takes all the potential pitfalls into consideration is very important.

Since RSA is based on arithmetic modulo large numbers, it can be slow in constrained environments. For example, on a heavily loaded web server, RSA decryption reduces the number of SSL requests per second that the server is capable to handle. Typically, one improves RSA's performance using special-purpose hardware, but several software implementations have also tried different improvements. In some cases, the speed improvement was invalidated by the pitfalls that where introduced. The speed of the algorithm depends mostly on the key size, the implementation type, the hardware on which the process is running, but it also depends on the protocol that has to be implemented. Many times implementers have tried to speed up the algorithm by using small public/private exponents, taking also great risks for the security of RSA into account. Using, for example, small public exponents increases the speed of the encryption and signature verification process, in comparison to the reversed processes (decryption and signing). The general algorithms that are typically used will take $O(n^2)$ time for public key operations, compared to private key operations that will need $O(n^3)$, where n is the bit number of the modulus N.

The choice of the modulus size is regarded to be one of the most important decisions in practical RSA implementations. It is quite obvious that the standard 512 bits long keys no longer provide sufficient protection, and therefore, the size should be increased. However, the complexity of modular exponentiation will grow rapidly with the size of the modulus, and thus choosing a size which combines efficient operation with long term security will become quite difficult.

The security of the algorithm is based on the assumption that the function $C = M^e \bmod N$ is hard to reverse without knowing the decryption exponent. But there is at least the theoretical possibility to find the factors of the modulus N through brute-force or by using other methods. This would expose the numbers p and q, and after using the Euclidian algorithm the decryption exponent would be revealed. The final result would be the compromised private key.

Since the publishing of the RSA, many researchers have tried to break the algorithm in order to find its weak points. Nonetheless, over the last 25 years of tests and mathematical analysis, it was possible to find different vulnerabilities, but no threat that could break the algorithms security was found. Programming a robust version of RSA is not a trivial task as the resulted methods of attack emphasize once again the importance of a strong implementation. Trying to classify the different aims of an attacker, we can distinguish the following: 1. recovery of the private key; 2. message decryption; 3. signature forgery.

It may be possible that an attack achieving the second or third result can be mounted without the attacker actually recovering the private key. Taking the impact on the security of the algorithm into account, we can establish the following classification over the methods of attack:

- attacks that expose the private key, essentially compromising the security of the whole algorithm;
- attacks that expose just a single encrypted message or forge a signature, but leave the private key intact for further use.

Examples of methods belonging to the first category are the attack against small private exponents (in certain cases), factoring large integers. The second category includes a wide range of subtle attacks that use complicated methods to decrypt the message. We underline the fact that in these cases, there are several constrains that have to be met in order to implement a successful attack (the time has to be less than the one for the brute-force attack as specific attacks use different properties of the algorithm, imposing different restrictions). The properties, which the different methods are trying to use, can lead to a second classification:

- elementary methods of attack;

- methods against the low private exponent;
- methods against the low public exponent;
- attacks against different implementations.

Factoring Large Numbers

For some time, people believed that the security of RSA depends on the problem of factoring prime numbers. If an efficient factoring algorithm would exist, then RSA would be insecure. Recent research has proved that breaking RSA is easier than factoring prime numbers, especially when the private exponent takes small values [1]. This is also the reason why the first attack on the RSA algorithm we consider is factoring the modulus N. We refer to factoring the modulus as the brute-force attack on RSA. Although factoring RSA has improved, the current state of the art is still far from posing a threat to the security of the algorithm (the hardware improvements are not enough), especially when RSA is used properly.

The most well-known factoring algorithms can be classified into two categories: algorithms whose running time depends only on the size of the factored number, and algorithms whose running time depends on the size of the factors. The older factoring algorithms searched for the smallest factors, and were thus of the second type. However, modern algorithms tend to use other approaches, which lead to different time results. The fastest factoring algorithm of the second type is the elliptic curve method. The running time is quite fast, but its basic operations are very slow, so it is unlikely that this algorithm will be able to find 1024 bit long RSA keys in the next few years.

Algorithms of the first type are much faster, since they can use a wider range of mathematical techniques. The fastest factoring algorithm of this type is, at the moment, the General Number Field Sieve, which is running in $\exp((c+O(1))*n^{1/3}*\log^{2/3} n)$, where c < 2. It is believed that this algorithm will be capable of factoring 1024 bit numbers in the next few years. Since the appearance of RSA, all the record breaking factorizations of RSA keys were based on algorithms of this type, and it is reasonable to believe that this trend will continue in the future. The best (and only) way to protect against this attack is to increase the size of the key [2].

Another factoring attack is that of Desmedt and Odlyzko, which applies equally on encryption and signing. This appears to be more practical with forging signatures rather than message decryption, since it is easier to demand the signature on different text messages instead of requesting lots of decryptions. This attack is particularly effective when the signed messages are small. The attacker is trying to factor this particular message into small primes (but the success is not guaranteed) as the signature on the message equals the product of the factors' signatures. The probability of success depends on the size of the message, not on the size of the modulus N. This is why messages should be larger than the RSA modulus and not a multiple of known values.

Elementary Methods of Attack

There are several older methods of attack, which demonstrate that an improper use of the algorithm can compromise the idea of security. The easiest method is trying to guess the message. Given the encrypted message and the public key, the attacker will guess the message and encrypt the text in order to compare the two encrypted parts. The easiest way to defend against this pseudo attack is to include a random sequence of bits at the end of the message.

Another attack method is concentrating on common used modulus. In order to avoid generating big prime numbers, implementers have tried to use common modulus for whole groups of users. A central authority was distributing the pair (e_i, d_i) to every user of the system. On a closer look at this system, one may notice that every user can use his pair of exponents to factorize the modulus. After doing this, it is possible to restore any private key based on the modulus N. The only method to avoid this scheme of attack is to deny the common use of the same modulus N for groups of users.

Another attack against RSA is called super-encryption. Simmons and Norris [3] observed that after a number of repeated encryptions the original message could eventually be recovered. This would lead to a tremendous weakness of the RSA if the number of required encryptions would be small. This is not the case if the primes are large and chosen at random, so the best method to protect the system against the super-encryption attack is to choose the prime numbers properly.

Methods Against the Low Private Exponent

This attack is a concern if the private exponent d is deliberately chosen to be small, in an attempt to improve the efficiency of decryption or signing. In such cases, the RSA is approximately 10 times faster than in the other cases. M. Wiener has demonstrated that in this case the RSA algorithm is particularly weak because the attacker has a good chance to determine the private exponent and to break the system. He has created the following theorem: *Let N = p * q with q < p < 2 * q* and $d < \frac{1}{3} * N^{1/4}$. Given the equation: e * d = 1 mod (p - 1)(q - 1), there is the possibility to efficiently recover the number d .

The demonstration of the theorem is based on approximations of continued fractions. For a 1024 bits modulus, Wiener has proved that the private exponent has to be at least 256 bits long to prevent this attack (this might be a problem for particular implementations of the algorithm for smartcards). He has also presented other methods of preventing this attack [4]. Boneh and Durfee have proved that for a private exponent $d < N^{0,292}$, the attack cannot be prevented [5]. The best way to block this type of attack is to choose a high private exponent, even if this will increase the working time of the algorithm.

Methods Against the Low Public Exponent

In real life RSA implementations, software developers are tempted to use a small public exponent in order to speed up the encryption and/or the verification of the signature. In this case, the danger of compromising the private key is smaller. But for typical values of the public exponent, there are still several possibilities of attack.

Coppersmith published the following theorem: *Given N an integer and* $f \in Z_N[x]$ *a monic polynomial of degree d, let* $X = N^{\frac{1}{d}-\varepsilon}$, *where* $\varepsilon > 0$. *If the pair (N, f) is known, there is the possibility to efficiently find out all numbers* $|x_0| < X$ *which satisfy the equation* $f(x_0) = 0 \bmod N$.

This offers an efficient way of calculating the roots for the *f mod N* equation (based on a lemma by Howgrave-Graham [6]). A first application of the theorem is the improvement of an older method called the "broadcast attack". The message that had to be sent to k participants was encrypted k times with the corresponding public keys (N_i, e_i). If, for simplicity reasons, all the members of the group would use $e_i = 3$ and if the group would consist of 3 users (k=3), we would obtain the following equation system:

$$\{C_1 = M^3 \bmod N_1 \quad C_2 = M^3 \bmod N_2 \quad C_3 = M^3 \bmod N_3$$

Applying the Chinese Remainder Theorem (CRT) on this system, it is possible to recover the message M computing the real cube root. The attack is possible when a small public exponent is used so the best method to prevent this attack is to use bigger exponent values, or to complete the message with a sequence at the end. Hastad mentions another attack scheme directed against related messages. Completing the messages with a linear polynomial $i * 2^m$ will also not prevent the attack [7]. The use of random padding will destroy any known relation between messages, and will avoid this attack. Related messages should not be encrypted with the same RSA key.

The RSA encryption is yet not effective on small messages, especially when the public exponent is low. In particular, if $C = M^e < N$, the message can be recovered from the cipher text by ordinary root extraction. There are two options, either the public exponent should have a big value or the messages should be large enough. The second suggestion should be considered since a small public exponent is often preferred. However, the developer has to ensure that the large message he has to encrypt is not the multiple of a known value, such as a large power of 2 (this would be the case if the message would be padded at the end with 0) [8].

Another interesting attack that can be used to decrypt related messages is known also as the Franklin-Reiter method: *Assuming that the messages* M_1 *and* M_2 *satisfy the following equation* $M_1 = f(M_2) \bmod N$, *and that* C_1, C_2 *and the function* $f \in Z_N[x]$ *are known, it is possible to find out the messages* M_1 *and* M_2 *for any small public exponent.*

The theorem was used also by Coppersmith to design another extended attack directed against random padded messages. This method is applicable only when e=3 and the padding sequence is shorter than 1/9th of the message length. For public exponent values other than 3, the method does not work

The partial exposure of the private key is one of the most dangerous attacks, because it leads to the recovery of the private key. Boneh, Durfee and Frankel have proved that it is possible to successfully recover the key if $e < \sqrt{N}$ and the least n/4 bits of d are known. This result is underlining once again just how important it is to keep the private key safe [9].

Attacks against Different Implementations

Another category of attacks, even though it is not significant as a threat to the RSA itself, is very interesting because of the pitfalls it reveals. All details such as design mistakes and small implementation bugs can bring even a good algorithm to an undesired breakdown. The famous timing attacks belong to this category. Kocher has proved that by knowing the time that is necessary to compute an encryption, it is possible to find out, bit by bit the private key [10]. To prevent this method, it is necessary to introduce random timeouts in the algorithm. Another possibility to perform this attack is to watch the power consumption (especially for

smartcards). Other successful attacks that have used other implementation specific vulnerabilities are the random faults attack [11], or the Bleichenbacher method (PKCS1) [12], etc.

Another common concern is how RSA security is affected if the pseudo primes used are, in fact, no primes at all. As a result, the factors of the modulus would be smaller, and it would be much easier for the attacker to factor it. Even if this seems to be more or less a theoretical concern, taking into account the modern methods of prime number generation, we still encourage you to pay attention to the algorithm that you are using.

The Bellare and Rogaway scheme, also called Optimal Asymmetric Encryption Padding, provides a supplement of protection to the encryption process [13]. Depending on the security level, the amount of padding is 16 bytes or more. Hence, one of the benefits is the fact that the achieved security can be related to the pseudorandom generator and the quality of the hash function.

Conclusions

It is clear that RSA is the most deployed public key algorithm today. One can find it in different operating systems (Windows, Apple, SunOS, Novell), in different hardware components (telephone systems, smartcards), or in communication protocols (S/MIME, SSL, IPSec, PKCS, S/WAN).

There are many methods of attack directed against the algorithm and many vulnerabilities that have to be considered in order to be able to implement a robust implementation of RSA. A deep analysis and a detailed knowledge of these can lead to a secure version. Some of these methods have both a theoretical and a practical value, depending on the impact that they have on the algorithm's security. Many of the presented attacks could become serious security problems if they would be mounted on raw forms of the RSA. We have also seen that there are efficient countermeasures for the real life implementations [14].

We note that in practical RSA implementations there are also other attack methods or potential threats, beyond those covered here, that should be considered from a security perspective. For example, the random key generator and the storage of the private key are vital issues in a secure implementation of RSA. There can be no secure RSA implementation without considering all the system's factors.

References

1. D. Boneh, R. Venkatesan, “Breaking RSA may not be equivalent to factoring, Advances in Cryptology” - Eurocrypt ‘98, Springer-Verlag, 1998, p. 59-71.

2. A. Shamir, “RSA for paranoids” - CryptoBytes, RSA Laboratories, 1995, vol. 1, no. 3.

3. G.J. Simmons, M.J. Norris, “Preliminary comments on the MIT public-key cryptosystem”, Cryptologia, 1977, Vol. 1(4), p. 406-414.

4. M. Wiener, “Cryptanalysis of short RSA secret exponents”, IEEE Transactions on Information Theory, 1990, vol.36, p.553-558.

5. D. Boneh, G. Durfee, “New results on cryptanalysis of low private exponent RSA”, Preprint, 1998.

6. N. Howgrave-Graham, “Finding small roots of univariate modular equations revisited”, Cryptography and Coding, Lecture Notes in Computer Science, Springer-Verlag, vol. 1355, p. 131-142.

7. J. Hastad, “Solving simultaneous modular equations of low degree”, SIAM Journal of Computing, 1988, p.336-341.

8. B. Kaliski, M. Robshaw, “The Secure Use of RSA”, RSA Laboratories, 1995, vol. 1, no. 3.

9. D. Boneh, G. Durfee, Y. Frankel, “An attack on RSA given a fraction of the private key bits”, AsiaCrypt ‘98, Lecture Notes in Computer Science, Springer-Verlag, 1998, vol. 1514, p. 25-34.

10. P. Kocher, “Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems”, Crypto ‘96, Lecture Notes in Computer Science, Springer-Verlag, 1996, vol. 1109, p. 104-113.

11. D. Boneh, R. DeMillo, R. Lipton, “On the importance of checking cryptographic protocols for faults”, Eurocrypt ‘97, Lecture Notes in Computer Science, Springer-Verlag, 1997, vol. 1233, p.37-51.

12. D. Bleichenbacher, “Chosen cipher-text attacks against protocols based on the RSA encryption standard PKCS #1”, Crypto ‘98, Lecture Notes in Computer Science, Springer-Verlag, 1998, vol. 1462.

13. M. Bellare, P. Rogaway, “Optimal asymmetric encryption” Advances in Cryptology, Eurocrypt ‘94, Springer-Verlag, 1995, p. 92-111.

14. D. Boneh., “Twenty years of attacks on the RSA cryptosystem”, American Mathematical Society, 1999, vol. 46, no. 2, p. 203-213.

Cristian Marinescu, University Politehnica of Bucharest, Splaiul Independentei, No. 313, Sector 6, Bucharest, Romania, Tel.: +40-21-4100325, E-mail: ntapus@cs.pub.ro

Nicolae Ţăpuş, University Politehnica of Bucharest, Splaiul Independentei, No. 313, Sector 6, Bucharest, Romania, Tel.: +40-21-4100325, E-Mail: cristian.marinescu@omicron.at.