ATTACKS ON THE RSA ALGORITHM

Dragan Savić[1],
Slobodan Damjanović[2]

[1]Singidunum University,
Department for postgraduate studies
32 Danijelova Street, Belgrade, Serbia,
[2]Ministry of Defence Republic of Serbia,
Belgrade, Serbia

Abstract:
The aim of this paper is to provide an overview of the current achievements in the domain of public-key cryptography within the framework of existing knowledge in literature, international standards and best practice as far as the RSA algorithm is concerned. This paper is particularly dedicated to the attacks on the RSA algorithm, whereas the ways to defend are suggested. Methods of attacks on the RSA algorithm are given and further retrospective of results obtained during the research are separately treated in the final part of the paper through the description of attacks with use of force, low-exponent attack, chosen-plaintext attack and timing attack.

Key words:
RSA algorithm, cryptography, attack, symmetric and asymmetric cryptography.

1. ATTACKS ON THE RSA ALGORITHM

Cryptography based on the public key enables the access to the private key. With a couple of values (e,n) which represent the public key, the attacker can obtain the private key. By analogy, the attack on the RSA can be easily carried out if the exponent is known. In general, this type of the attack is called the brute forced attack.

The most effective attack against a RSA algorithm up to now has been the factorization of the number n. If the attacker factorizes n, he can easily discover $\varphi(n)=(p-1)(q-1)$ as well, and in this way define the secret exponent d from $d \equiv 1(\mathrm{mod}\,\varphi(n))$ by using the Euclidean algorithm. The safety of the RSA algorithm lays in the factorization, namely at the factorization of n which has over 200 decimal digits with the primitive method of dividing by all simple numbers smaller than $\sqrt{n}$, with the help of a computer which is able to perform 10^9 divisions of this kind in one second, about 10^{81} years is needed for the factorization. Presently, the fastest algorithms need $O\left(e^{c(\log n)^{1/3}(\log\log n)^{2/3}}\right)$ operations for factorization, which means that not one polynomial algorithm is known for factorization. It is important to highlight that there are the cases when n is easier to factorize that normally. This is the case when the numbers p and q are very close to each other or if $p-1$ and $q-1$ have small simple factors. These cases should be avoided while choosing the parameter for the RSA's coding system.

Correspondence:
Dragan Savić

e-mail:
dragansavic.rm@gmail.com

DOI: 10.15308/Sinteza-2016-131-136

Also, the attack based on the attempts to calculate $(p-1)\cdot(q-1)$, is possible, yet the time complexity of this attack is not easier than the previously described attack. It is possible to search directly for the number *d*, but it has been proven that this procedure is more complex that previously described possibilities.

There are several algorithms for factorization:

- division – represents the oldest and the least effective method, but it implies tryout of all primitive numbers smaller or equal to $n^{\frac{1}{2}}$ (the exponential complexity),
- quadratic Sieve algorithm – the fastest algorithm for the numbers smaller than 110 figures,
- Multiple Polynomial Quadratic Sieve – faster version of the previous algorithm,
- GNFS – General Number Field Sieve,
- SNFS – Specific Number Field Sieve.

The above-mentioned algorithms represent the best options for the attack on the RSA algorithm. The Sieve algorithms have the so-called super-polynomial complexity (sub-exponential), and the complexity of the Sieve algorithms with number fields asymptotically approach the polynomial behavior.

Table 1 shows the time in relation to the length of the code needed for a computer with 1 MIPS speed from the public key to the secret key[1]. Keys of 1024, 2048 or 4096 bits are used for the files encryption.

Time needed for calculating the secret key from the public one

Length of the key in bits	*Time needed*
50	3.9 hours
100	74 years
150	10^6 years
200	$3{,}8*10^9$ years

2. THE ATTACK ON THE RSA ALGORITHM BY USING A SMALL EXPONENT

It appears that, if the *e* is relatively small, it does not influence the safety of the RSA algorithm itself. If the exponent for the coding is small (for example 3, 17, 65537), the operation of coding is much faster. The only drawback of the usage of the small exponent is visible in the coding of short messages if the exponent is chosen. If we assume that we have three users with various values of the public module n_1, n_2, n_3 and that they use the same public exponent $e=3$. Then, we assume that someone wants to send the identical message m. Yet, we have already seen if any relatively simple exponent with $\varphi(n)$ is fine, that we can easily choose $n=pq$ so that the number 3 is a relatively simple number $(p-1)(q-1)=\varphi(n)$. Now, the coding of m message, $m^3 \bmod n$. Only then the adversary can discover the following ciphertext

$$c_1 \equiv m^3 (\bmod n_1), c_2 \equiv m^3 (\bmod n_2), c_3 \equiv m^3 (\bmod n_3)$$

After that, the adversary can find the solution of the system of linear configurations by using the Chinese theorem on the remainder

$$x \equiv c_1 (\bmod n_1), x \equiv c_2 (\bmod n_2), x \equiv c_3 (\bmod n_3)$$

In this way, the adversary will get the number *x* with the characteristic $x \equiv m^3 (\bmod n_1 n_2 n_3)$

However, having in mind that $m^3 < n_1 n_2 n_3$ is equal to $x = m^3$, so that the adversary can calculate the original message *m* and discover $\sqrt[3]{x}$.

This attack has been described by Coopersmith, Franklin, Patarin and Reiter.

If we code an open text *m* and then *m+1*. The above-mentioned authors claim that *m* could be discovered. We have the following coded text:

$$c_1 = m^3$$

$$c_2 = (m+1)^3 = m^3 + 3m + 3m^2 + 1 = c_1 + 3m + 3m^2 + 1$$

Now we are trying to solve m. The next step is:

$$\frac{c_2 + 2c_1 - 1}{c_2 - c_1 + 2} = \frac{(m+1)^3 + 2m^3 - 1}{(m+1)^3 - m^3 + 2} = \frac{3m^3 + 3m^2 + 3m}{3m^2 + 3m + 3} = m$$

This can be generalized. Firstly you can generalize the message m and $\alpha m + \beta$ for the known α, β. Secondly, it works for the exponents bigger than 3. That the attack works in the timeframe $O(e^2)$ and it is possible for small exponents. Finally, it can work for k messages related to the higher degree of polynomials.

Another way – if we choose number 3 for e and if we take $M < n^{\frac{1}{3}}$ (the message shorter than $\sqrt[3]{n}$), the message can be easily decoded by the operation $M^{\frac{1}{3}}$ $\sqrt[3]{M}$ as:

$$M^3 \bmod n = M^3, \text{ if } M \le n^{1/3}$$

i.e.: $M < n^{1/3} = M$

Coding and verification of signatures with the help of the RSA algorithm are faster if the small value is used for *e*, it can be uncertain as well. If you want to code $\frac{e(e+1)}{2}$ of the linear dependent messages with different public keys which have the same values e, that kind of the system can be attacked. If a message has a smaller number or if they are not related, no problem can arise. If the messages are identical, the e message is sufficient.

1 For example, the Pentium I computer had about 150 MIPS.

The simplest solution is to expand the messages with independent random values. "A random pad" is added to the messages prior to the coding. In this way, we would never send entirely identical messages. It this way $m^e \bmod n \neq m^e$ is also secured. This is done in most of the realizations of the RSA algorithm, for example PEM and PGP. However, there are attacks based on the Coppersmith's result and LLL algorithm which indicate that neither the RSA coding system with small exponent e is safe.

More precisely, the Coppersmith's result is used (1997):

If $f \in Z[x]$ nominal polynomial s degree δ, also $n \in N$. If x_0 exists so that $f(x_0) \equiv 0 (\bmod n)$ and $|x_0| \leq X = n^{\frac{1}{\delta - \varepsilon}}$, than x_0 could be found in the timeframe which is polynomial in $\log n$ and $\frac{1}{\varepsilon}$.

The next attack in this category is Hastad's attack (1985). If we assume that the data depending on the user is added before the coding at the beginning of each message, for example

$$c_i = \left(i \cdot 2^h + m\right)^e \bmod n_i, i = 1,...,k.$$

We have k of the polynomial $g_i(x) = \left(i \cdot 2^h + x\right)^e - c_i$ and we look for m with characteristic

$$g_i(m) \equiv 0 (\bmod\ n_1)$$

If $n = n_1 n_2 ... n_k$. By using the Chinese theorem on the remainder we can find t_i so that

$$g(x) = \sum_{i=1}^{k} t_i g_i(x) \text{ and } g(m) \equiv 0 (\bmod n)$$
$$\left(t_i \equiv 1 (\bmod n_i), t_i \equiv 0 (\bmod n_j)\right) \text{ for } j \neq i$$

The polynomial g is normalized and the degree e. If $k > e$; *i.e.* if we have several users (intercepted ciphertext) than the public exponent, than $m < \min_i n_i < n^{\frac{1}{k}} < n^{\frac{1}{e}}$, so m can be effectively found by using the mentioned Coppersmith's result.

The next attack of the type "low exponent of decoding" on the RSA algorithm was discovered by Michael Wiener (1990). This type of the attack d is reconstructed, where d could reach maximum of one fourth of n, while e is less than *n*.

$$ed - k\varphi(n) = 1$$
$$\varphi(n) \approx n \Rightarrow \frac{k}{d} \approx \frac{e}{n}$$

If $p < q < 2p$. If $d < \frac{1}{3} n^{0,3}$, than

$$\left|\frac{k}{d} - \frac{e}{n}\right| < \frac{1}{2d^2}$$

According to the classical Legendre's theorem from Diophantine approximations, d must be the directory of a convergent p_m/q_m in the development of the continued fraction of the number *e/n*, so that d can be effectively calculated from the public code (n, e). The number of convergent in total is $O(\log n)$, while each convergent can be tested in polynomial time.

Here we have to mention Verheul – van Tilborg attack as well (1997) which represents the expansion of Wiener's attack which is applicable when d has several more bitss than $n^{0,25}$. For $d > n^{0,25}$, their attack uses the search by brute force for $2t + 8beats$ with certain assumptions to partial quotient in a continued fraction, where $t = log_2(d/\ n^{0,25})$.

Also, the Boneh – Durfee (1990) and Blomer – May (2001) attacks belong to this type of an attack. They are based on the Coppersmith's technique which uses LLL-algorithm for calculating the small roots of modular polynomial equations. These attacks are »heuristic«, and in the practice they are satisfactory if $d < n^{0,292}$.

It is believed that the secret exponent $d > \sqrt{n}$ should be used, as it is known that all the above-mentioned attacks are completely useless in that case.

A small modification of the Verheul – van Tilborg attack was made in 2004 based on Vorli's result (1981) from Diophantine approximations, which means that all rational numbers *p/q* which satisfy the inequalities

$$\left|\alpha - \frac{p}{q}\right| < \frac{c}{q^2},$$

For a positive realistic number c, are in the form

$$\frac{p}{q} = \frac{rp_m + 1 \pm sp_m}{rq_m + 1 \pm sq_m}$$

For a $m \geq -1$ and non-negative whole numbers r and s in the way that *rs<2c*.

Ibrahimpašić (2008) claims that Vorli's result is the best possible, in the sense that the condition *rs<2c* cannot be replaced by *rs<(2-ε)c* for $\varepsilon > 0$.

In the both mentioned expansion of the Winner's attack, candidates for the secret exponent take the form of

$$d = rq_{m+1} + sq_{m+1}.$$

All the possibilities for d are tested, while the number of all possibilities is roughly speaking (the number of possibilities for *r*) x (the number of possibilities for *s*), which is $O(D^2)$, where $D = d/n^{0,25}$.

More precisely, the number of possible couples (r, s) in Verheul – van Tilborg attack is $O(D^2 A^2)$, with

$$A = \max\{a_i : i = m+1, m+2, m+3\}$$

While in Andrej Duella's variant from 2004 is $O(D^2 \log A)$.

The new modification of the Verheul – van Tilborg attack was proposed by Sun, Wu and Chen (2007). This modification requests heuretical search by brute force for *2t - 10* bits, so its complexity is also $O(D^2)$.

Drastic improvements cannot as Steinfeld, Contini, Wang and Pieprzyk (2005) proved that among the algorithms of this type there is not an algorithm with sub-exponential dependence on D.

3. THE ATTACKS ON THE RSA ALGORITHM BY USING THE CHOSEN CHIPERTEXT

The attack on the RSA algorithm by using the chosen chipertext attack is based on the presumption that the attacker in some way manages to find the chipertext of his choice.

The attacker taps the communication channel over which the RSA coded messages are exchanged, discovers the message C, in a way that he wants to discover its original content, *i.e.* mathematically he was to discover:

$$M = C^d \bmod n$$

With the assumption that the attacker knows the public key (e,n), in order to obtain M the attacker firstly chooses a random message R, where $R < n$, than he codes the message with the public key:

$$X = R^e \bmod n$$

Chipertext message C is multiplied by using the X: $Y = X \cdot C \bmod n$

Also, the attacker calculates the modular inverse values from R: $T = R^{-1} \bmod n$

While the attacker assumes that:

$$X = R^e \bmod n \text{, and } R = X^d \bmod n$$

The attacker must wait for the user to digitally sign Y with his private key, which is how he effectively decodes Y, and sends $U = Y^d \bmod n$ to the attacker. The attacker must calculate the following:

$$T \cdot U \bmod n = \left(R^{-1} \bmod n\right) \cdot \left(Y^d \bmod n\right) \bmod n =$$
$$\left(R^{-1} \bmod n\right) \cdot \left[\left(X \cdot C \bmod n\right)^d \bmod n\right] \bmod n =$$
$$\left(R^{-1} \bmod n\right) \cdot \left[\left(X \cdot C\right)^d \bmod n\right] =$$
$$\left(R^{-1} \bmod n\right) \cdot \left[\left(X^d \bmod n\right) \cdot \left(C^d \bmod n\right) \bmod n\right] \bmod n =$$
$$\left(R^{-1} \bmod n\right) \cdot \left[R \cdot M \bmod n\right] \bmod n =$$
$$R^{-1} \cdot R \cdot M \bmod n = M$$

4. TIME BASED ATTACK ON THE RSA ALGORITHM

Amelioration of the cryptography based on the public key has revealed some facts and regularities. For example the modular and exponential operations used for the RSA algorithm request discrete time intervals. If the RSA operations are carried out by using the Chinese Remainder Theorem, the attacker can use small time differences while conducting the RSA operations, and that way in many cases discover *d*. This type of the attack is based on passive tapping of the RSA operations.

The attacker passively observes k operation and measures the time T needed for calculating $M = C^d \bmod n$.

The assumption is that the attacker recognizes *C* and n. This method will enable someone who knows the exponents $d_0, d_1, ..., d_{s-1}$ to discover the bit d_s; obtain the exponent d, starting from d_0, repeating the attack until he discovers the entire exponent $d = d_0, d_1, ..., d_{s-1}, d_s, ..., d_\beta$. Now, we start from d_0 the least important bit in comparison to *d*. Having in mind that d is an odd number, we know that $d_0 = 1$. In this phase we have: $d_0 = 1$, $M \equiv C$, $C \equiv C^2 \pmod n$.

Than we consider d_1. If $d_1 = 1$ than the victim will have to do $M \leftarrow M \cdot C \pmod n$, $C \leftarrow C^2 \pmod n$ if $d_1 = 0$, than $C \leftarrow C^2 \pmod n$.

If t_i is needed for the hardware calculation $M_i \cdot C_i \equiv M_i \cdot M_i^2 \pmod n$. Of course, t_i is different one from the other, as the time for calculation $M_i \cdot M_i^2 \pmod n$ depends on the value of M_i.

This attack requests monitoring of the cryptographical operations in the real time, which to a larger extent limits the possibility to carry out the attack itself.

5. ATTACKS ON THE RSA ALGORITHM OF THE TYPE OF "JOINT MODULUS"

One of the possible realizations of the RSA algorithm gives the same value *n*, but different values for exponent e and d. Sadly, this does not function. The most visible problem is: if the same message is ever coded with two different exponents (both have the same modulus) and the two exponents are coprime (as in the general case), then the open text can be reconstructed without a single decoding exponent.

If *m* message is in the form of an open text, the keys for the decoding are e_1 and e_2. Joint modulus is n. Two decoded messages are: $c_1 = m^{e_1} \bmod n$ and $c_2 = m^{e_2} \bmod n$.

A cryptanalyst knows n, e_1, e_2, c_1, c_2. The description of the way it is being reconstructed m follows.

As e_1 and e_2 are coprime, by expanded Euclidean algorithm can be found r and s, so that: $re_1+re_2=1$.

Assuming that r is negative (or r or s must be negative value, why we consider the r is negative), then the expanded Euclidean algorithm can be again applied in order to calculate c_1^{-1}. Than $\left(c_1^{-1}\right)^{-\gamma} c_2^5 = m$.

There are two other treacherous attacks on this type of the system. One uses probability method for factorizing n. The other uses algorithm for calculating someone's secret key for factorizing the module. Both attacks are described in detail in [95].

Do not allow a group of users to share a single n.

6. THE ATTACK ON CODING AND SIGNING BY THE RSA ALGORITHM

It makes sense that a message is signed before coding, but not everyone sticks to this rule. When the RSA algorithm is used, the attack can be carried out on the protocols doing the coding before the signing.

Alice wants to send a message to Bob. She firstly codes the message with Bob's public key, then signs with her private key. Her key and the signed message look like this:

$$\left(m^{e_B} \bmod n_b\right)^{d_A} \bmod n_A$$

The description of how Bob can claim that Alice sent him m`, and not m. Bear in mind the following: as Bob knows the factors n_B (his modulus), he can calculate discrete algorithms in relation to n_B. That's why, he is supposed to discover x so that: $m' = m \mod n_b$.

Then, if he can give xe_B as his new public exponent and keeps n_B as his modulus, sent him the message m` coded with this new exponent.

7. CONCLUSION

It could be concluded that the RSA algorithm for four decades after having appeared still represents the safe solution, whose usage with up to know techniques of attacks is still safe. This claim is rooted in the fact that even though the detailed studying of the RSA algorithm is ongoing, a method has not yet been discovered that would completely destroy the RSA. Everything comes down to discover individual weaknesses, which gives a warning how to choose parameters for the implementation of the RSA. All this enables that the RSA, for the time being, is assessed as a safe cryptosystem.

REFERENCES

[1] Song Y. Y. (2008). *Cryptoanalytic Attacks on RSA*, Bedfordshire, Massachusetts: University of Bedfordshire, UK and Massachusetts Institute of Technology, USA, Springer Science+Business Madia, LLC.

[2] Jason Hinek, M. (2001). *Cryptoanalysis of RSA and its variants*, London, New York: Chapman&Hall/CRC, Taylor&Francis Group, Boca Raton.

[3] Katzenbeisser, S. (2001). *Recent Advances in RSA Cryptography*, Vienna: University of Technology, Austria, Springer Science+Business Media, LLC.

[4] Davida, G.I. (1982) *Chosen Signature Cryptanalysis of the RSA (MIT) Public Key Cryptosystem*, Winsconsin: University of Wisconsin, Departement of EECS.

[5] Rivest, R. L., Kaliski, B. (2005) *RSA problem. In H. C. A. van Tilborg, editor, Encyclopedia of Cryptography and Security*, Springer.

[6] Sun, H.-M., Wu, M.-E., Chen, Y.-H. (2007). *Estimating the prime-factors of an RSA modulus and an extension of the Wiener attack.* In J. Katz and M. Yung, editors, ACNS, volume 4521 of Lecture Notes in Computer Science, Springer.

[7] Steinfeld, R., Contini, S., Wang, H., Pieprzyk, J. (2005). *Converse results to the Wiener attack on RSA.* In S. Vaudenay, editor, Public Key Cryptography, volume 3386 of Lecture Notes in Computer Science, Springer, 2005.

[8] Ibrahimpasic, B. (2008). Cryptoanalysis of KMOV cryptosystem with short secret exponent, Central European Conference on Information and Intelligent Systems, CECIIS.

[9] Boneh, D., Durfee, G., Frankel, Y. (1998). An attack on RSA given a small fraction of the private key bits. In K. Ohta and D. Pei, editors, ASIACRYPT, volume 1514 of Lecture Notes in Computer Science.

[10] Blomer, J., May. A. (2005). A tool kit for finding small roots of bivariate polynomials over the integers. In R. Cramer, editor, EUROCRYPT, volume 3494 of Lecture Notes in Computer Science, Springer.

[11] Coppersmith, D., Franklin, M. K., Patarin, J. and Reiter, M. K. (1996). Low-exponent RSA with related messages: Advances in Cryptology - EuroCrypt '96, (pp 1-9), Berlin Springer-Verlag.

[12] Anderson, R. J., Needham, R. (1995). Robustness Principles for Public Key Protocols: Advances in Cryptology – CRYPTO 95 Proceedings, Berlin Springer-Verlag.

[13] Rosati, T. (1989). A High Speed Data Encryption Processor for Publik Key Cryptography: Proceedings of the IEEE Custom Integrated Circuits Conference, (pp. 12.3.1-12.3.5).

[14] Hastad, J. (1986). On Using RSA with Low Exponent in a Public Key Network: Advances in Cryptology – CRYPTO 85 Proceedings, (pp. 403-408), Berlin Springer-Verlag.

[15] Desmedt, Y., Odlykzo, A.M. (1986). A Chosen Text Attack on the RSA Cryptosystem and Some Discrete Logarithm Problems: Advances in Cryptology – CRYPTO 85 (pp. 516-522) Proceedings, Berlin Springer-Verlag.

[16] Adi, Sh. (1983). A Polynomial Time Algorithm for Breaking the Basic Merkle-Hellman Cryptosystem. Advances in Cryptology: Crypto 82, (pp. 279-288). Plenum Press.

[17] Miyaguchi, S. (1982). Fast Encryption Algorithm for the RSA Cryptographic System: Proceedings of Compcon 82, IEEE Press, (pp. 672-678).

[18] Chen, C.-Y., Chang, C.-C., Yang, W.-P. (1996) Cryptanalysis of the secret exponent of the RSA scheme. *Journal of Information Science and Engineering,* 12, (pp. 277–290).

[19] Diffie, W. (1992). The First Ten Years of Public-Key Cryptography: *Contemporary Integrity, G.J. Simmons, ed., IEEE Press,* (pp. 135-175).

[20] Moore, J.H. (1992). Protocol Failures in Cryptosystems, *Conterporary Cryptology: The Science of Information Integrity, G.J. Simmons, ed., IEEE Press* (pp. 541-558).

[21] Brickell, E.F., Odlyzko, A.M. (1991). Cryptoanalysis: A Survey of Recent Results, *Conteporary Cryptology: The Science of Information Integrity, G.J. Simmons, ed., IEEE Press*, (pp. 501-540).

[22] Hule, H., Muller, W.B. (1988). On the RSA Cryptosystem with Wrong Keys, *Contributions to General Algebra 6,* Vienna: Verlag Holder-Pichler-Tempsky, (pp. 103-109).

[23] Alexi, W., Chor, B.-Z., Goldreich, O., Schnorr, C.P. (1988). RSA and Rabin Functions: Certain Parts are as Hard as the Whole, *SIAM Journal on Computing*, v. 17, n. 2, (pp. 194-209).

[24] Simmons, G.J. (1983). A Weak Privacy Protocol Using the RSA Cryptosystem, *Cryptologia*, v. 7, n. 2, (pp. 180-182).

[25] DeLaurentis, J.M. (1984). A Further Weakness in the Common Modulus Protocol for the RSA Cryptosystem, *Cryptologia*, v. 8, n. 3, (pp. 253-259).

[26] Wiener, M.J. (1990). Cryptoanalysis of Short RSA Secret Exponents, *IEEE Transactions on Information Theory*, v. 36, n. 3, (pp. 553-558).

[27] Denning, D.E. (1984). Digital Signatures with RSA and Other Public-Key Cryptosystems: Communications of the ACM, v. 27, n. 4, (pp. 388-392).

[28] Wu, C.K., Wang, X.M. (1993). Determination of the True Value of the Euler Totient Function in the RSA Cryptosystem from a Set of Possibilities: Electronics Letters, v. 29, n. 1. (pp. 84-85).

[29] Quisquater, J.-J. (1982). Fast Decipherment Algorithm for RSA Public Key Cryptosystem: Electronics Letters, v. 18, (pp. 155-168).

[30] Coppersmith, D. (1995) Factoring with a hint. Technical Report RC 19905, IBM Research Report.