1. Dragan SAVIĆ[1], 2. Petar MILIĆ[2], 3. Borislav MAŽINJANIN[3], 4. Petar SPALEVIĆ[2]

Singidunum University (1), University of Pristina – Kosovska Mitrovica, Faculty of Technical Sciences (2), Academy for National Security (3)
ORCID: 1. https://orcid.org/0000-0003-0427-8379

doi:10.15199/48.2022.02.04

Cryptanalytic attacks on RSA algorithm and its variants

***Abstract**. The goal of this paper is to provide a review of principles and techniques used in public-key cryptanalysis with special attention on RSA algorithm. The ways to defend against attacks on RSA algorithm are suggested. Furthermore, we provide a retrospective of results obtained during the research separately treated in the final part of the paper through the description of brute-force, low-exponent attack, chosen-plaintext attack and timing attack.*

Streszczenie.** Celem tego artykułu jest przedstawienie przeglądu zasad i technik stosowanych w kryptoanalizie klucza publicznego ze szczególnym uwzględnieniem algorytmu RSA. Sugerowane są sposoby obrony przed atakami na algorytm RSA. Ponadto przedstawiamy retrospektywę wyników uzyskanych podczas badań oddzielnie potraktowanych w końcowej części artykułu poprzez opis ataku brute-force, ataku o niskim wykładniku, ataku z wybranym tekstem jawnym i ataku czasowego.* (Ataki kryptoanalityczne na algorytm RSA i jego warianty***)

Keywords: public-key, RSA, cryptanalysis, attacks.
Słowa kluczowe: kryptografia, RSA, ataki kryptograficzne.

Introduction

De-facto standard used as a strong encryption/decription and digital signature scheme in securing network communication is RSA public key cryptosystem. This public key cryptosystem enables practical way for maintaining privacy and integrity of information. Lots of cryptographic algorithms are designed to make the critical transactions secure. Among them, RSA has been the most popular cryptographic algorithm. The simplicity and effectiveness of this algorithm makes it more attractive to the security providers. The security of this algorithm lies in computing the factors of a large composite integer. Currently 1024 bits (for non-critical information) and 2048 bits or above (for critical information) composite integer is used to achieve desired security level. The workload of factoring a 1024-bit composite integer is assumed to be as complex as workload of 2^{80} instructions which is the current benchmark used in cryptography (likely to increase in future).

In RSA [1] public and private keys are calculated as inverse of each other. Private key can also be used by the sender to generate the digital signatures. Anyone who knows the public key of the sender can verify the message. If the message is intended for the particular receiver, the user applies the receiver's public key on the message.

Due to the use of keys of large size, involved computations are increased. The use of keys of large size becomes bottleneck if the system is working on small device or heavily loaded systems. Consequently, the demand of the cryptosystem with fast computations arises.

The public key cryptography concept is to avoid the prior distribution of the shared encryption key. For this purpose, two keys are used for communication, one key is public known to everyone (public key) whereas the other is kept as secret (private key). These keys should be computed such as to use one key for encryption (public key) and the other key for decryption (private key).

So far, the most effective attack against RSA algorithm has been the factorization of the number n which represent modulus used for factorization. This modulus is product of two chosen prime numbers (p, q). If the attacker factorizes n, he can easily discover that $\varphi(n) = (p-1)(q-1)$ as well as the way to define the secret exponent d from $d \equiv 1(mod\varphi(n))$ by using the Euclidean algorithm. The safety of the RSA algorithm lies in the factorization. Namely, at the factorization of n which has over 200 decimal digits with the primitive method of dividing 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. Up to now the fastest algorithms need $O(e^{c(logn)^{1/3}c(loglogn)^{2/3}})$ operations for the factorization, which means that polynomial algorithms is unknown for the factorization. It is important to highlight that there are cases when n is easier to factorize than it would normally be. 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 encryption system.

Although, the attack based on the attempts to calculate $(p-1)\,x\,(q-1)$, is possible, the time complexity of this attack is not easier than the previous one described before. It is possible to search directly for the number d, but it has been proven that this procedure is more complex than the previous possibilities described above.

There are two types of factorization method, general purpose and special purpose. General purpose factorization methods depends on the length of the number to be factored while special purpose factorization methods depend on the length of the smallest factor of the number to be factored. There are many factorization methods available in literature:

- Division Method - represents the oldest and the least effective method, but implies the try out of all primitive numbers smaller or equal to $n^{\frac{1}{2}}$ (the exponential complexity),
- Fermat's Method - the complexity of this method is sub exponential numbers of bits,
- Pollard $p-1$ Method,
- Pollard rho Method,
- Continued Fraction Factorization Method,
- Quadratic Sieve Method - the fastest algorithm for the numbers smaller than 100 decimal digits,
- Number Field Sieve Methods (NFS), and
- Elliptic Curve Factorization (ECM).

After the study of different methods it is clear that the complexity of general purpose methods is exponential or sub exponential in the size of the number to be factored. NFS method can be used to factor small number, and 232 digit number is the largest factored number by this method till 2009 [2]. On the other hand, complexity or special purpose methods is exponential or sub exponential to the size of the smallest factor of the number to be factored. ECM is the best example for this type of numbers, 83 digit number is the largest factor computed in 2013 by this method [2]. The above-mentioned methods represent the best options for the attack on the RSA algorithm.

Table 1 shows the time in relation to the length of the code needed for a computer with 1 MIPS speed to reach from the public key to the secret key.

Table 1. Time needed to calculate 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	$3{,}8 \cdot 10^9$ years

In this paper we provide a review of principles and techniques used in cryptanalysis of RSA algorithm along with coutnermeasuers and pitfalls that should be avoided. Inspecting flaws using weak public/private keys, integer factorization problem and some specific low parameter selection attacks can exploit possible vulnerabilities, where we give more deep unterstanding about underlying mathematics and proper parameter selection. It can be shown that a well implemented algorithm is unbreakable and can survive a number of cryptanalytic attacks. In Section 2 of the paper we provide a short description of elementary attacks carried out on the RSA with basic mathematical concepts behind. Section 3 provides user with contemporary network communication interception attacks as well as ways how to defend against them and how to properly implement variants of RSA algorithm. Finally, conclusions and future remarks are given at the end of the paper.

Mathematical concepts of elementary attacks

Cycling attack

In 2018, Ye, Liu and Gardner [3] showed that repeatedly encrypted text can be deciphered again (after *a + 1* encryption steps), and the original text found before this cipher text is the actual message (after *a* encryption). For small messages, *a* is very small. In 2017, Barak [4] generalized the cycling attack. Overmars and Venkatraman [5] in 2020 showed that for balanced primes and sufficiently large modulus, cycling attack becomes ineffective. The experimental results obtained during implementation of this attack are shown in Table 2. These results reveal little dependence on modulus *N*.

Table 2. Cycling attack's experimental results

Size of N	*Average running time (seconds)*
8	0,000012
16	0,000010
32	0,000008
64	0,000010
128	0,000011
256	0,000013
512	0,000011
1024	0,000011
2048	0,000016
4096	0,000015

Common modulus attack

Common Modulus attack is applicable in scenarios where the central authority distribute private and public exponents to the users. All users share common modulus without the knowledge of the prime factors. So, if d has somehow been leaked, the only work is to generate another private key from same N and prime factors. In 2010, Simmons [6] showed that the message can be recovered if the same plaintext is encrypted with two different and relatively prime public exponents with the same modulus. In 2018, Ye, Liu and Gardner [3] showed the failure of common modulus RSA deterministically. They proved that the user can recover other's private key in the group with the use of his pair of public and private key. This attack is not applicable for a single user with multiple instances with the same modulus. In 2015, Peng at al. [7] described the strongest common modulus attack (single user) with a small private exponent. For two instances of e and d with same modulus, the security can be broken if $< N^{\frac{5}{14}}$. For more than 7 instances with the same modulus the bound becomes $< N^{\frac{1}{2}}$. [8]

Attacks in polynomial time

Takayasu and Lu at al. [9] [10] proved that the message M can be computed in polynomial time. In 2017, Boudabra [11] generalized this attack and proved that with small public exponents and different moduli which related to each other by some known function $f(x)$ and the number of users that are greater than $\max(e_i \cdot degree(f_i(x))$, plaintext M can be computed in polynomial time.

If we assume that data that are depending on the user is added before the encryption at the beginning of each message, they can be represented with following equation

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

k can be represented as polynomial $g_i(x) = \left(i \cdot 2^h + x\right)^e - c_i$ while m has following characteristic

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

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

$g(x) = \sum_{i=1}^{k} t_i g_i(x)$ and $g(m) \equiv 0 (\bmod n)$

$\left(t_i \equiv 1(\bmod n_i), t_i \equiv 0(\bmod n_j)\right)$ for $j \neq i$

The polynomial g is normalized and if $k > e$; i.e. if we have several users (intercepted ciphertext), 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 Padmaja's result. [12]

The best case is, if the figures are of the same width as the column – 8 cm. The letters in the figures should be not smaller than 2 mm. Figure 1 presents an example of a graph.

Small exponent *e* attack

Small public exponent is always attractive for the low cost involved in the cryptosystem. 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 ciphering is small, the operation of ciphering is much faster. The only drawback of the usage of the small exponent is visible in the ciphering of short messages. When the exponent is chosen, 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 $e = 3$. After that we assume that someone wants to send an identical message . If we have already seen that any relatively simple exponent with $\varphi(n)$ is fine, than we can easily choose $n = pq$ so that the number 3 is a relatively simple number $(p-1) \cdot (q-1) = \varphi(n)$. Now the encryption of m message is, $m^3 mod\ n$. Only in this case the attacker can discover the following ciphertext

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

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

(4)
$$x \equiv c_1(mod\ n_1), x \equiv c_2(mod\ n_2), x \equiv c_3(mod\ n_3),$$

In this way the attacker will get the number x with the characteristic

(5) $$x \equiv m^3(mod\ n_1 n_2 n_3)$$

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

This attack has been described by Coopersmith, Franklin, Patarin and Reiter [12]. They claim that m could be discovered in following manner. If we have the following ciphered text:

(6) $$c_1 = m^3$$

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

m can be obtained as:

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

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

Another way is - 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 $M^{\frac{1}{3}} \cdot \sqrt[3]{M}$ as:

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

Generation and verification of the signatures with the help of the RSA algorithm is faster if the small value is used for e, but it can be uncertain as well. If we want to code $\frac{e(e+1)}{2}$ of the linear dependent messages with different public keys which have the same values e, the system became vulnerable. On the other hand, if the message consisted of small numbers or they are not related, problems are avoided. If the messages are identical, the e message is sufficient.

In 2016, Savić and Damjanović [13] showed that if two plaintexts related with some function (with known constants) are encrypted by same small public exponent $(e = 3)$, then the plaintext can be computed in polynomial time.

Wang at al. [14] proved that for small e, k_0 in the RSA equation can known (since $k_0 < e$). This gives the information about the most significant bits (MSBs) of the private exponent; few bits (say d_1) can be known where $d - d_1 < p + q$. It means one can get half of the MSBs of d for balanced primes. For $e = 3$, it is very simple to guess the value of k_0. Also, if one knows the bits of d, k_0 can be recovered easily. This attack is not the total failure of the cryptosystem. He signifies that small e, half of the MSB of d can be made public.

Also, Takayasu and Kunihiro in 2017 described [15] attacks which belong to this type of the attacks. 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 practice they reach satisfactory level 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 otherwise.

Small exponent *d* attack

As the public and private exponents are inverse of each other, one can get small private exponent by selecting the large public exponent. If someone requires the fast decryption speed, small secret exponent is required. But very small secret exponent can break the complete system.

In 2015, Peng at al. [7] gave the attack for balanced primes with small private exponent. In this kind of attack, the d is reconstructed, where d could reach maximum of one fourth of n, while e is less than n.

(9) $$ed - k\varphi(n) = 1$$

(10) $$\varphi(n) \approx n \Rightarrow \frac{k}{d} \approx \frac{e}{n}$$

If $p < q < 2p$. If $d < \frac{1}{3} n^{0,25}$, than

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

According to the classical Legendre's theorem from Diophantine approximations, d must be the directory of a convergent $\frac{p_m}{q_m}$ in the development of the continued fraction of the number $\frac{e}{n}$, 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.

In 2015, Meng and Zheng [16] extended the Wiener's bound using exhaustive search. This type of attack is applicable when d has several more bites than $n^{0,25}$. For $d > n^{0,25}$, their attack uses the search by brute force for $2t + 8bits$ with certain assumptions to partial quotient in a continued fraction, where $t = log_2\left(\frac{d}{n^{0,25}}\right)$.

In 2020, by lattice method, Bahig at al. [17] rigorously proved that RSA modulus with balanced primes can factored in polynomial time if $d < \frac{1}{3}N^{\frac{1}{4}}$. Furthermore, Asbullar at al. [18], extended the work of Meng and Zheng [16] for Wiener's bound. In 2019, Susilo, Tonien and Yang [19] proposed a variant of Wiener's attack which is having less time and space complexity as compare to the Wiener's method.

In 2019, Nitaj, Susilo and Tonien [20] claimed that Vorli's result is the best possible one, in the sense that the condition $rs < 2c$ cannot be replaced by $rs < (2-\varepsilon)c$ for > 0 .

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

(12) $$d = rq_{m+1} + sq_m$$

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

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

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

while in Duella's variant from 2004 is $O(D^2 logA)$.

The new modification of the Verheul - van Tilborg attack was proposed by Santosh, Narasimham and Pallam in 2015 [21]. This modification requests heuristic search by brute force for $2t - 10$ bits, so its complexity is also $O(D^2)$. They gave their analysis on improved bounds for private key exponent. In 2021, Mumtaz and Ping [22] presented a lattice-based attack for large values of private exponent close to $\lambda(N)$. The modulus can be factored if d satisfies $|\lambda - d| < N^{0,25}$.

Contemporary attack review

Choosen ciphertext attack

A chosen-plaintext attack is called adaptive if the attacker can choose the ciphertexts depending on previous outcomes of the attack. It is well known that plain RSA is susceptible to a chosen-ciphertext attack. [23]

The attacker taps the communication channel over which the RSA coded messages are exchanged and discovers the message C, i.e. which mathematically rewritten looks like:

(12) $$M = C^d mod\ n$$

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

(13) $$X = R^e mod\ n$$

Ciphertext message C is multiplied by using the X:

(14) $$Y = X \cdot C\ mod\ n$$

Also, the attacker calculates the modular inverse values from R:

(15) $$T = R^{-1}\ mod\ n$$

The attacker assumes that:

(16) $$X = R^e\ mod\ n$$

and

(17) $$R = X^d\ mod\ n$$

Attacker must wait for the user to digitally sign Y with his private key, which is how he effectively decodes Y, and sends

(18) $$U = Y^d\ mod\ n$$

to the victim. He must calculate the following:

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

The attack on the RSA algorithm by using the chosen ciphertext attack is based on the presumption that the attacker in some way manages to find the chipertext of his choice. An attacker who wishes to find the decryption $M \equiv C^d mod\ n$ of ciphertext C can chose a random integer S and ask for the decryption of the innocent-looking message $C' \equiv S^e C\ mod\ n$. From the answer $M' \equiv (C')^d$, it is easy to recover the original message, because $M \equiv M'S^{-1}$.

It is not necessary for an attacker to learn the complete decrypted message in a chosen-ciphertext attack. Single bits per chosen ciphertext may be sufficient. In particular, there exists an algorithm that can decrypt a ciphertext if there exists another algorithm that can predict the least significant bit of a message given only the corresponding ciphertext and the public key. [24]

Timing attack

In general, timing attacks are used to analyze differences in execution time that result from differences in input parameters of a cryptographic algorithm. These timing differences are often caused by optimizing algorithm implementations, but they may leak information about the input parameters. Using a timing attack, attacker hopes to find secret information, like bits from a secret RSA exponent [25]. The concept of timing attacks has been known for years, but Kocher's results are new and significant in the way that they can recover complete key information given only the running time of an operation. Previous attacks could only recover partial key information or required timing information on the individual steps within a cryptographic operation.

For one of the attacks to succeed, it must be possible to measure the running time of crypto graphic operations. Thus operations in an interactive protocol such as SSL, or in a cryptographic module in the attacker's possession, such as a smart card, are at the highest risk. This is true even if there is some "noise" in the measurement, such as transmission delays, since the attacker can factor out the noise by averaging enough measurements. Operations performed privately and without external feedback, on the other hand, such as off-line digital signatures or file encryption, are unaffected.

While ameliorating cryptography based on the public key, some facts and regularities have been noticed. 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 in that way he discovers d. This type of the attack is based on passive tapping of the RSA operations.

The attacker passively observes k operation and measures time T needed for calculating

(20) $$M = C^d \, mod \, n$$

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

$$d_0 = 1, . \; M \equiv C, . \; C \equiv C^2 (mod \, n)$$

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

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

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

The RSA private key operation consist of computing a single modular exponentiation operation. Several algorithms for implementing this operation are available, yet the runtime of these algorithms often depends on the input values. When an attacker is able to supply the input value and measure the time it takes for the target implementation to perform the secret key operation, the attacker can reveal the secret RSA exponent or the factorization of N into prime factors p and q. Several methods are available to prevent secret information to leak through timing differences. The preferred method is using RSA blinding, which is actually successfully implemented in cryptographic libraries like OpenSSL.

To defend against timing attacks, one must try to lessen the correlation between the runtime and the private exponent. One solution is to add a delay, so that every modular operation takes the same fixed time. Another solution is called blinding. This transforms the data before the private operation using a random value generated by the cryptographic module. Now the operation is performed on a random data unknown to the adversary, which precludes the attack. Important point to note is that timing attacks are not only defined against RSA.

Joint modulus attack

Attacker can under certain circumstances decrypt message sent without actually requiring a private key by using four RSA parameters $\{d, p, q, \phi(N)\}$ that form the RSA trapdoor. These four pieces of information are equally important. Knowledge of any one of them reveals the knowledge of the remaining three, and hence break the RSA encryption completely. However, if RSA is not used properly, it may will be possible to break the RSA encryption without use of any knowledge of $\{d, p, q, \phi(N)\}$. An examle of such case is improper use of the common modulus N in RSA encryption. [26]

The most visible problem is if the same message is coded with two different exponents (both have the same modulus) and the two exponents are coprime (as are in the general case), then the open text can be reconstructed without a single decoding exponent.

In 1984 an even stronger attack against Common Modulus RSA was discovered by DeLaurentis. He proved that two encryptions of a plain text is not necessesary for decryption of all encrypted messages. Any user of the system can actually create a new private key which will work with any other chosen public key. If M message is in the form of an open text than keys for the decoding are e_1 and e_2. Joint modulus is N. Two decoded messages are:

$$C_1 = M^{e_1}(mod \, N)$$
$$C_2 = M^{e_2}(mod \, N).$$

where $\gcd(e_1, e_2) = 1$. Than attacker can recover the plaintext M without using any of the trap-door information $\{d, p, q, \phi(N)\}$.

Let $N_1 = N_2$ and $M_1 = M_2$ but $e_1 \neq e_2$ and $\gcd(e_1, e_2) = 1$ such that,

$$C_1 = M^{e_1}(mod\ N)$$
$$C_2 = M^{e_2}(mod\ N).$$

Then M can be recovered easily; that is,

(21) $$\{[C_1, e_1, N], [C_2, e_2, N]\} \overset{\rho}{\Rightarrow} \{M\}$$

Since $\gcd(e_1, e_2) = 1$, then $e_1 x + e_2 y = 1$, with $x, y \in \mathrm{Z}$, which can be done by the extended Euclid's algorithm (or the equivalent continued fraction algorithm) in polynomial-time. Thus,

$$C_1^x C_2^y \equiv (M_1^{e_1})^x (M_2^{e_2})^y$$
$$\equiv M^{e_1 x + e_2 y}$$
$$\equiv M(mod\ N)$$

There are two other dangerous attacks on this type of the encryption system. One uses probability method for factoring N. The other uses algorithm for calculating someone's secret key for factoring the module. From here we can conlude that sharing of single *N* to the group of users should be prohibited.

Attack on RSA signatures

What made you think that the public key is enough to recreate the signature? It is sufficient to verify a signature that you are given, but is not sufficient to generate new ones (or we hope, if that's not true, the signature scheme is broken). When you are using RSA, the signature verification process is (effectively) checking by:

(22) $$S^e = Pad\big(Hash(M)\big)(mod\ N)$$

Definition says that S is the signature, M is the message, e and N are the public exponent and modulus from the public key, $mod\ N$ means that equality is checked, padding function is Pad, and $Hash$ is the hashing function.

Now, if we were trying to forge a signature for a message M' (with only the public key) we could certainly compute

(23) $$P' = Pad\big(Hash(M')\big)$$

however, then we would need to find a value S' with:

(24) $$S'^e = P'(modN)$$

and, if N is an RSA modulus, we don't know how to do that. The holder of the private key can do this, because he has a value d with the property that:

(25) $$(x^e)^d = x(modN)$$

for all x. That means that:

$$(P')^d = (S'^e)^d = S'(modN)$$

is the signature. Now, if we have only the public key, we don't know d, getting that value is equivalent to factoring N, and we can't do that. The holder of the private key knows d because he knows the factorization of N.

The attack against RSA encryption can be easily adapted to RSA signatures to provide an existential forgery under a chosen-message attack. [27]

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. [28]

Summary

Four decades of research into inverting the RSA function produced some insightful attacks, but no devastating attack has ever been found. Most problems appear to be the result of misuse of the system, bad choice of parameters or flaws in implementations. The attacks discovered so far mainly illustrate the pitfalls to be avoided when implementing RSA. It could be concluded that the RSA algorithm still represents a safe solution, whose usage with 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 break the RSA. Everything comes down to discover individual weaknesses which give us a warning how to choose parameters for the implementation of the RSA. According to all aforementioned, we can conclude that, with the proper selection of parameters, RSA is considered to be secure cryptosystem.

REFERENCES

[1] Jaju, A. S. and Chowhan, S. S. (2015) *A Modified RSA Algorithm to Enhance Security for Digital Signature*, In Proceedings of the 2015 International Conference and Workshop on Computing and Communication (IEMCON), Vancouver, BC, Canada, pp. 1 – 5.

[2] Jeřábek, E. (2016) *Integer factoring and modular square roots*, Journal of Computer and System Sciences, Vol. 82, No. 2, pp. 380 – 394.

[3] Ye, X., Liu, C. and Gardner, D. (2018) *Weakness of RSA cryptosystem characteristic*, In Proceedings of the International Conference of Computational Methods in Sciences and Engineering, Thessaloniki, Greece, Vol. 2040, pp. 1 – 10.

[4] Barak, B. (2017) *The Complexity of Public-Key Cryptography*, In: Lindell Y. (eds) Tutorials on the Foundations of Cryptography. Information Security and Cryptography. Springer, pp. 45 – 77.

[5] Overmars, A. and Venkatraman, S. (2020) *Mathematical Attack of RSA by Extending the Sum of Squares of Primes to Factorize a Semi-Prime*, Mathematical and Computational Applications, Vol. 25, No. 4. pp. 1 – 15.

[6] Simmons, J. G. (2010) *A weak privacy protocol using the RSA crypto algorithm*, *Cryptology*, Vol. 4, No. 1, pp. 58 – 93.

[7] Peng, L., Hu, L., Lu, Y., Sarkar, S., Xu, J. and Huang, Z. (2015) *Cryptanalysis of Variants of RSA with Multiple Small Secret Exponents*, In Proceedings of 2015 International Conference on Cryptology (INDOCRYPT 2015), Bangalore, India, pp. 105 – 123.

[8] Raghunandan, K. R., Aithal, G. and Shetty, S. (2019) *Comparative Analysis of Encryption and Decryption Techniques Using Mersenne Prime Numbers and Phony Modulus to Avoid Factorization Attack of RSA*, In Proceedings of the 2019 International Conference on Advanced Mechatronic Systems, Kusatsu, Japan, pp. 152 – 157.

[9] Takayasu, A. and Kunihiro, N. (2016) *Small secreet exponent attacks on RSA with unbalanced prime factors*, In Proceedings of the 2016 International Symposium on Information Theory and Its Applications (ISITA), Monterey, USA, pp. 236 – 240.

[10] Lu, Y., Zhang, R., Peng, L. and Lin, D. (2015) *Solving Linear Equations Modulo Unkown Divisors: Revisited*, In Proceedings of the 2015 International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT 2015), Auckland, New Zeland, pp. 189 – 2013.
[11] Boudabra, M. and Nitaj, A. (2017) *A new generalization of the KMOV cryptosystem*, Journal of Applied Mathematics and Computing, Vol. 57. No. 1. pp. 229 – 245.
[12] Padmaja, J. C., Srinivas, B. and Bhagavan, S. V. (2018) *A Systematic Mapping Study of the Published Research on Cryptanalytic Attacks on RSA*, International Journal of Pure and Applied Mathematics, Vol. 118, No. 23, pp. 283 – 291.
[13] Savić, D. and Damjanović, S. (2016) *The Attacks on the RSA Algorithm*, In Proceedings of the 2016 International Scientific Conference on ICT and E-business Related Research (SINTEZA 2016), Belgrade, Serbia, pp. 131 – 136.
[14] Wang, S., Qu, L., Li, C. and Fu, S. (2015) *A New Attack on RSA with Known Middle Bits of the Private Key*, IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, Vol. 98. No. 12, pp. 2677 – 2685.
[15] Takayasu, A. and Kunihiro, N. (2017) *A Tool Kit for Partial Key Exposure Attacks on RSA*, In Proceedings of the Cryptographer's Track at the RSA Conference 2017 (CT-RSA 2017), San Francisco, USA, pp. 58 – 73.
[16] Meng, X. and Zheng, X. (2015) *Cryptanalysis of RSA with a small parameter revisited*, Vol. 115, No. 11, pp. 858 – 862.
[17] Bahig, M. H., Nassr, I. D., Bhery, A. and Nitaj, A. (2020) *A Unified Method for Private Exponent Attacks on RSA Using Lattices*, International Journal of Foundations of Computer Science, Vol. 31, No. 02, pp. 207 – 231.
[18] Asbullah, A. M., Rahman, A. N. N., Ariffin, K. R. M., Sapar, H. S. and Yunos, F. (2020) *Cryptanalysis of RSA Key Equation of N=p^2q For Small | 2q - p | Using Continued Fraction*, Malaysian Journal of Science, Vol. 39, No. 1, pp. 72 – 80.
[19] Susilo, W., Tonien, J. and Yang, G. (2019) *The Wiener Attack on RSA Revisited: A Quest for the Exact Bound*, In the Proceedings of the Australasian Conference on Information Security and Privacy, Christchurch, New Zeland, pp. 381 – 398.
[20] Nitaj, A., Susilo, W. and Tonien, J. (2019) *Improved Cryptanalysis of the KMOV Elliptic Curve Cryptosystem*, In the Proceedings of the 13th International Conference on Provable Security, Cairns, Australia, pp. 206 – 221.
[21] Santosh, R., Narasimham, C. and Pallam, S. (2015) *Short Secret Exponent Attack on LSBS-RSA*, Vol. 12. No. 6A, pp. 714 – 718.
[22] Mumtaz, M. and Ping, L. (2021) *An improved cryptanalysis of large RSA decryption exponent with constrained key*, International Journal of Information and Computer Security, Vol. 14. No. 2, pp. 102 – 117.
[23] Kumar, V., Kumar, R. and Pandey, K. S. (2017) *An Enhanced and Secured RSA Public Key Cryptosystem Algorithm Using Chinese Remainder Theorem*, In Proceedings of the 3rd International Conference on Next Generation Computing Technologies (NGCT 2017), Dehradun, India, pp. 543 – 554.
[24] Al Barazanchi, I., Shawkat, A. S., Hammed, H. M. and Al-Badri, L. S. K. (2019) *Modified RSA-based algorithm: a double secure approach*, TELKOMNIKA, Vol. 17. No. 6, pp. 2818 – 2825.
[25] Patil, P., Narayankar, P., Narayan, D. G. and Mena, S. M. (2016) *A Comprehensive Evaluation of Cryptographic Algorithms: DES, 3DES, AES, RSA and Blowfish*, Procedia Computer Science, Vol. 78, pp. 617 – 624.
[26] Kong, F., Yu, J., Zhou, D., Jiang, Y. and Shang, J. (2017) *Security Analysis of Two Inter-Organization Cryptographic Schemes*, In Proceedings of the 4th International Conference on Machinery, Materials and Information Technology Applications (ICMMITA 2016), pp. 1323 – 1327.
[27] Abobeah, M. R., Ezz, M. M. and Harb, M. H. (2015) *Public-Key Cryptography Techniques Evaluation*, International Journal of Computer Networks and Applications, Vol. 2, No. 2. pp. 64 – 75.
[28] Akchiche, O. and Khadir, O. (2018) *Factoring RSA moduli with primes sharing bits in the middle*, Applicable Algebra in Engineering, Communication and Computing, Vol. 29, No. 3, pp. 245 – 259.