RSA Cryptanalysis using multiplicative group order

Amina Miroud
Research Laboratory on Computer Sciences Complex Systems ReLa(CS)²
University of Oum El Bouaghi
P.O. Box 358
04000 Oum El Bouaghi, Algeria
miroud.aminaa@gmail.com

Abdelhabib Bourouis
Research Laboratory on Computer Sciences Complex Systems ReLa(CS)²
University of Oum El Bouaghi
P.O. Box 358
04000 Oum El Bouaghi, Algeria
a.bourouis@univ-oeb.dz

Lakhdar Derdouri
Research Laboratory on Computer Sciences Complex Systems ReLa(CS)²
University of Oum El Bouaghi
P.O. Box 358
04000 Oum El Bouaghi, Algeria
derdouril@yahoo.fr

Rohallah Benaboud
Research Laboratory on Computer Sciences Complex Systems ReLa(CS)²
University of Oum El Bouaghi
P.O. Box 358
04000 Oum El Bouaghi, Algeria
r_benaboud@yahoo.fr

Mouna Nouadri
Research Laboratory on Computer Sciences Complex Systems ReLa(CS)²
University of Oum El Bouaghi
P.O. Box 358
04000 Oum El Bouaghi, Algeria
mouna_93@live.com

***Abstract*—Four decades of attempts to break the RSA cryptosystem produced various attacks. However, none of these attacks is considered as mean for breaking the system, because most of them focus on the misuse of the RSA system or the bad choices of its parameters that could easily be avoided. Therefore, we need generic attacks that can be applied in a reasonable amount of time to break RSA. A generic cryptanalysis, presented in this paper, consists in finding out the trapdoor which allows to invert the RSA function. To determine the trapdoor, we search for the exponent of the RSA multiplicative group. It is obtained by calculating the ordrers of a small set of numbers then taking the highest value. We then define the trapdoor as the inverse of the public key modulo this found exponent. Therefore, using this trapdoor, it is easy to decrypt any RSA ciphertext. The interest of this attack is that it can be applied regardless of the implementation of the RSA system. Furthermore, it can not be thwarted using an OAEP padding.**

***Keywords*—Cryptanalysis, RSA cryptosystem, Attack, Modulus, Trapdoor.**

I. Introduction

Since the use of written communication, people have been interested in trying to find means to protect the content of their communications assuming that an adversary can read all of their messages. This has led to the introduction of numerous methods and techniques for secret communication, a science known as cryptography. Public key cryptography (or asymmetric cryptography) has been the most significant and striking development in the history of cryptography. This revolutionary concept has been introduced in the famous paper "New Directions in Cryptography" [1]. Nowadays, There are many asymmetric schemes for generating keys to encrypt and decrypt messages. The RSA public key cryptosystem was the first practical realization of a public key encryption scheme.

A secure public key cryptosystem requires a mathematical operation which is easy to compute (encryption) but computationally difficult to reverse (decryption) in a realistic time without knowing a special secret information, called the trapdoor, which is the private key. Such a mathematical function is called a trapdoor one-way function. Whereas the objective of cryptography is to develop secure trapdoor one-way encryption functions, a parallel domain, called cryptanalysis, aims to find out means to reverse these functions or find possible weaknesses. Thus, cryptanalysis is essential for asymmetric cryptosystems to provide higher levels of security by avoiding previously discovered vulnerabilities.

Cryptanalysts are continuously targeting all asymmetric cryptosystems without exception where some of them are now considered weak. Among the public key cryptosystems that have proved their resistance to the various attempts of cryptanalysis, we find the RSA cryptosystem. This cryptosystem is the most popular form of public key cryptography in the world today because of its simplicity and its robustness against the different proposed attacks since its invention.

In this paper we introduce a new cryptanalysis attempt to break the RSA cryptosystem. The proposed attack aims to find a trapdoor for the RSA encryption function in a realistic amount of search time. To find the trapdoor, we must search the exponent of the RSA multiplicative group $\lambda(N)$. It is the maximal order of elements of $(\mathbb{Z}/N\mathbb{Z})^*$, where N is the RSA public modulus and λ refers to the Carmichael's lambda function. In order to determine $\lambda(N)$, it is necessary to compute the order of a restricted set of elements in the considered RSA modulus. Once $\lambda(N)$ is obtained, we define the trapdoor by $e^{-1} \mod \lambda(N)$. Thus, we can decrypt any ciphertext in the RSA modulus N.

The remainder of this paper is structured as follows: Section 2 shows the basic functioning principle of the RSA cryptosystem. The RSA security, the RSA problem and cryptanalytic attack models are explained in section 3. A categorization of the proposed attacks against this cryptosystem are discussed

in section 4. Section 5 describes in detail the proposed attack and provides an algorithmic formalization of the procedure of calculating the value of $\lambda(N)$. Conclusions and future research directions are presented in Section 6.

II. The RSA cryptosystem

The RSA cryptosystem is the most widely known and widely used public-key cryptosystem in the world today since it supports both secrecy and authentication, and therefore can be used for encryption, key transport, authentication (digital signature), and certification. Thus, it is at the heart of web traffic and electronic credit-card payment systems. We describe below a simplified version of the RSA encryption system.

Like any public key cryptosystem, RSA cryptosystem consist of three efficiently computable algorithms : a key generation algorithm, an encryption algorithm, and a decryption algorithm.

1) **Key Generation:** The key generation algorithm consists of randomly generating two balanced primes p and q so that their product $N = pq$ is called the RSA modulus. After choosing p and q, the algorithm also chooses an encryption exponent e co-prime to $\varphi(N) = (p-1)(q-1)$, where φ is the Euler totient function, and then computes the private exponent d as the inverse of the public encryption exponent modulo $\varphi(N)$ (or modulo $\lambda(N) = lcm(p-1, q-1)$, where λ denotes Carmichael's lambda function and *lcm* is the least common multiple). The key generation algorithm outputs the public key (e, N) and the private key (d, N).
2) **Encryption:** In order to use the RSA cryptosystem to encrypt messages, it is necessary to encode them as a sequence of numbers of size less than N. Thus, the first step of the encryption algorithm is to convert the plaintext of characters into an integer. This can be done easily by assigning distinct integers to the distinct characters, for example, by converting each character to its ASCII code. The next step takes the public key (e, N) and a plaintext message $m \in \mathbb{Z}_N$ as input in order to compute the ciphertext c as following : $c \equiv m^e \mod N$.
3) **Decryption:** Since only the owner of the private key knows it, only him can decrypt a message encrypted with the associated public key. Therefore, the decryption algorithm takes the private key (d, N) and a ciphertext $c \in \mathbb{Z}_N$ as input, where $c \equiv m^e \mod N$ for some $m \in \mathbb{Z}_N$, and outputs the plaintext $m \equiv c^d \mod N$. The relationship between e and d ensures the correctness of the decryption algorithm.

III. The security of RSA and cryptanalytic attack models

The security of RSA depends on the hardness of solving the RSA problem (RSAP). It consists in recovering the plaintext m from the corresponding ciphertext c, given only the RSA public key (e, N). Mainly, it is the problem of computing the e^{th} root of c modulo N. Thus, it is clear that RSA can be completely broken using a modular e^{th} root algorithm which would allow the adversary to recover encrypted messages even without knowing the private key. However, at the present time, there is no efficient algorithm known for this problem and the only known mean is to factor N. But currently, there is no efficient algorithm to factor large numbers and the discovery of a such algorithm would "break" RSA.

For these reasons, it is generally believed that the security of RSA relies on the difficulty of factoring the modulus N. However, it may be possible to solve the RSA problem without factoring. Hence, the RSA algorithm is not based completely on the hardness of factoring.

When talking about cryptanalytic attacks, it is interesting to recall that there is various models or types. The commonly admitted classification specifies the kind of access a cryptanalyst has to a system under attack. The greater the access the cryptanalyst has to the system, the more critical and useful information he can get to exploit to break the cryptosystem. The cryptanalyst analyzes the ciphertext trying to "break the cryptosystem, to recover the plaintext and to obtain the used key so that future ciphertexts could be easily recovered. Modern cryptography adopts the Kerckhoffs's principle which assumes that the encryption and decryption algorithms are public knowledge and available to anyone.

The *Ciphertext-Only Attack* (COA) is the most likely encountered model in real life cryptanalysis. The cryptanalyst has access only to ciphertexts without access to the corresponding plaintexts. Modern cryptosystems are required to be extremely resistant to this type of attack. The brute force (or exhaustive key search) are common attacks in this model but the cryptanalyst has to obtain some information about the plaintext to allow deciding that the tried key is correct or not.

In the *Known-Plaintext Attack* (KPA), the cryptanalyst has access to at least a limited number of pairs of ciphertexts and corresponding plaintexts. The cryptanalyst tries to find a correlation between bits of ciphertexts and corresponding plaintaxts. The *confusion* and *diffusion* are two properties of the operation of a secure cryptosystem identified by Claude Shannon in his report A Mathematical Theory of Cryptography [2]. These properties, when carefully introduced, prevent the use of statistics and other methods of cryptanalysis. Modern cryptosystems are generally very resistant to the various attacks in this model.

The Chosen-Plaintext Attack (CPA) model assumes that the cryptanalyst is able to choose a number of plaintexts and have access to the corresponding ciphertexts. The widely used public-key cryptosystems are in essence the most exposed to the attacks of this model because the encryption key is publicly distributed. Hence, anyone can choose plaintexts and encrypt them easily. So public-key algorithms must be very resistant to all chosen-plaintext attacks. The *Adaptive Chosen-Plaintext Attack* (CPA2) is a subcategory of CPA where the cryptanalyst uses results obtained using previous plaintexts as a guide to choose future plaintexts and conduct a more targeted attack.

The other attack models give the cryptanalyst more access and power which are not easily obtained in real life

cryptanalysis. So, attacks in these models are not considered serious problems. The *Chosen-Ciphertext Attack* (CCA) and its subcategory known as *Adaptive Chosen-Ciphertext Attack* (CCA2) are typical examples of these models. In these models, the cryptanalyst can choose a set of ciphertexts and obtain easily the corresponding plaintexts. The *Open key model attacks* is another model requiring more power assuming that the cryptanalyst has some knowledge about the used keys.

IV. Breaking RSA

Four decades of cryptanalysis of the RSA system have led to a number of rigorous attacks, but none has been described as a destructive. In practice, most successful discovered attacks focus on insecure implementations of the RSA system. Hence, they illustrate the dangers of the misuse of RSA. These attacks are not considered as means for breaking the RSA system, because they exploit weaknesses in specific implementations. In the absence of these weaknesses, these attacks remain ineffective. For a survey of these specific attacks, see [3].

We turn our attention to an entirely different category of attacks. Rather than attacking a specific implementation, these attacks focus on the RSA algorithm. Therefore, they aim at the cryptosystem itself regardless of any weaknesses in its implementations. For this reason, we can consider them as generic attacks. The attacks of this category are the most dangerous on the RSA system because they are not related to specific implementations or cases to be applied. We classify these attacks into two main classes. Attacks of the first class enable the attacker to find out the private key and thus to read all messages encrypted under a given key. However, a successful attack of the second class enables the attacker to read only one ciphertext, because the success of the attack does not lead to the discovery of the private key. Therefore, to read a second ciphertext, the attacker has to start the procedure again. Thus, successful generic attacks allow attackers to break the RSA cryptosystem. Known attacks of the latter category are outlined below.

A. Factoring the RSA modulus N

The first possible attack against the RSA cryptosystem consists in attempting to factor the public modulus N, into its two prime factors, p and q. Factoring an RSA modulus make the attack damaging, because it would allow the attacker to get easily the private exponent d which would allow him to read all messages encrypted with the public key and to forge signatures.

Although researchers have made a great progress in the field of integer factorization, we do not know yet an efficient factorization algorithm to threaten the security of RSA. In particular, there is no known polynomial factorization algorithm in the size of the modulus to date. Hence, the security of the RSA cryptosystem depends on the presumed difficulty of the factorization problem. We specify that the fastest known factoring algorithm today is the General Number Field Sieve (GNFS) [4], which can factor a number of n bits in time $exp(\mathcal{O}(1)(\log n)^{1/3}(\log\log n)^{2/3})$. For more information about factorization methods, we refer the reader to [5] [6].

B. Cycling Attacks

In a cycling Attack, the attacker re-encrypt a ciphertext c until it cycles back to itself i.e., he computes the sequence $c_{i+1} = RSA_{\langle N,e\rangle}(c_i) \equiv (c_i)^e \mod N$, where $c_0 = m$ and $c_1 = c \equiv m^e \mod N$, until c is obtained for the first time. Thus, if $r+1$ re-encryptions are performed to recover the ciphertext c, then

$c_{r+1} = RSA_{\langle N,e\rangle}(c_r) \equiv {c_r}^e \mod N \equiv c^{e^{r+1}} \equiv c \mod N$,

and so $c_r = c^{e^r} \equiv m \mod N$. This result is due to the fact that RSA encryption function is a permutation of a finite set. It has been shown by Friedlander, Pomerance and Shparlinski [7], that for sufficiently large RSA modulus N, the realization of the so-called cycling attack on the RSA cryptosystem is expected to be unfeasible.

A generalized version of the cycling attack has been proposed by Williams and Schmid in 1979. This modified version find either a non-trivial factor of N or, in rare cases, a cycle modulo N as in the basic cycling attack [8]. Hence, it can be considered as a factorization method. However, breaking RSA via a generalized cycling attack is no more effective than other factorization methods. A more precise analysis of the low efficiency of the generalized cycling attacks on RSA modulus is provided in [9].

Based on the RSA cyclic encryption function, a recent attack, presented in [10], can find a new trapdoor different from the original one, but plays the same role by decrypting any ciphertext in the RSA modulus N. The decrypting algorithm is presented below where the function "*search_degree_max*" is called only once to determine the composition maximum degree d_max of the ciphering function and this by applying the ciphering function to a restricted set of residues in the modulus N (the first twenty residues of the modulus). The new decrypting key is defined by e^{d_max}. This cryptanalysis can be viewed as a very efficient optimization of the cycling attack because of its realistic performance time.

```
1:  procedure decrypting (e, N, C : BigInteger)
2:      M, d_max : BigInteger;
3:      function search_degree_max (e, N : BigInteger)
4:          m, max, i : BigInteger;
5:          m ← 0;
6:          max ← 0;
7:          i ← 2;
8:          while (m <> 1) or (i = 20) do
9:              m ← 1;
10:             while i^(e^m) mod N <> i do
11:                 m ← m + 1;
12:             end while
13:             if m > max then max ← m;
14:             end if
15:             i ← i + 1;
```

```
        end while
        return max
    end function
    d_max ← search_degree_max(e, N) − 1;
    M ← c^{e^{d_max}} mod N;
end procedure
```

V. The proposed attack

Cyclic groups are groups in which every element is a power of a fixed element called a generator of the group. The multiplicative group of nonzero elements $(\mathbb{Z}/n\mathbb{Z})^*$ is a cyclic group if $n = 2, 4, p^k$ or $2p^k$ for any prime $p \geqslant 3$ and natural number $k \geqslant 1$. In the case, where this group is cyclic, it is referred to any cyclic generator of the group as a primitive root modulo n. Thus, cyclic generators are characterized by a maximal order equal to $|\mathbb{Z}_n^*|$ which is in turn equal to $\varphi(n)$. Recall that the multiplicative order of an element e of $\mathbb{Z}_n^*$ modulo n is the smallest positive integer i with $e^i \equiv 1 \mod n$.

It is well-known that for the RSA composite modulus N (where $N = pq$ and p, q are different odd primes), the multiplicative group $(\mathbb{Z}/N\mathbb{Z})^*$ is not cyclic. Hence, we definitely cannot talk about generators of $(\mathbb{Z}/N\mathbb{Z})^*$. However, Carmichael extended the concept of primitive roots to primitive λ-roots which are elements in $\mathbb{Z}_N^*$ with the maximal order, since the generators of cyclic groups are also elements with the maximal order. Therefore, a primitive λ-root modulo N is an element which generates the largest cyclic subgroup of $\mathbb{Z}_N^*$. Briefly, a primitive λ-root modulo N is an integer coprime to N such that the multiplicative order of this integer modulo N is equal to $\lambda(N)$.

There are several proposed estimates for the least prime primitive root, and the least primitive root of cyclic groups, which is not the case for the least primitive λ-root, because it is not that simple to generalize the concept of primitive root to composite modulus. Martin [11] had given a bound for $g^*(q)$, the least prime λ-root modulo q, for almost all integers $q \geqslant 2$, of the form $g^*(q) \ll_\varepsilon \omega(\varphi(q))^{44/5+\varepsilon}(\log q)^{22/5}$, where $\omega(\varphi(q))$ is the number of distinct prime factors of $\varphi(q)$, and ε is a positive real number. But this may not be sharp, since the least primitive λ-root is not necessarily prime. Next, Ambrose in [12] had established an upper bound for $s^*(n)$, the least λ-root modulo a positive integer n, of the form $s^*(n) \ll_\varepsilon n_c^{\frac{1}{2}+\varepsilon}$, such that $s^*(n)$ is expressible as a sum of two squares and n_c denotes the largest odd cube-free divisor of n.

To achieve our proposed attack against the RSA system, we need to use an upper bound for the least primitive λ-root modulo the RSA composite modulus N. But, as we have seen, calculating an upper bound for the least primitive λ-root modulo N using the current proposed estimates needs to know additional information that we do not have as cryptanalysts of the RSA system, such as the largest odd cube-free divisor of N. Furthermore, these estimates may not be accurate, since the least primitive λ-root is not necessarily prime. Therefore, we propose the following upper bound which is based only on the known information public modulus N.

Conjecture 1. *For an RSA composite modulus $N = pq$, where p and q are two distinct primes, we define an upper bound for $g_\lambda(N)$, the least primitive λ-root modulo N, as follows:*

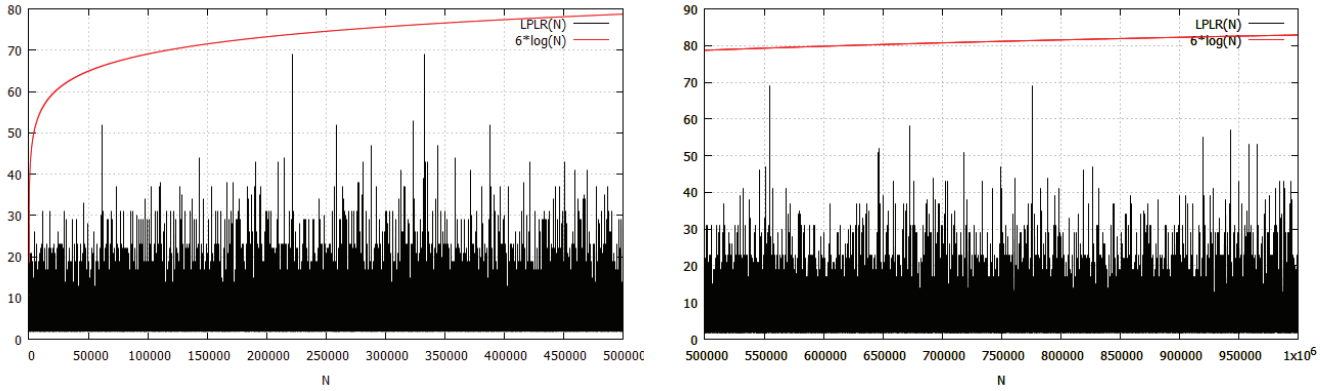
$$g_\lambda(N) \ll 6\log(N)$$

This conjecture followed as a consequence of our empirical approach which was carried out on composite modulus of the form $N = pq$, such that p and q are two distinct primes and their product belong to the finite interval $[1, 10^6]$. Our experimental approach has followed the next steps :

1) First, we have searched for the least primitive λ-root for composite integers N which have the features mentioned above.
2) We then have used the results of the first step to get the next curves, where $LPLR(N)$ denotes the least primitive λ-root modulo N.
3) The previous conjecture is formulated after observing the curves in Fig. 1.

The use of the proposed upper bound ensures us to find at least one primitive λ-root. Thus, it ensures that we find the least primitive λ-root. Since primitive λ-roots are elements of $(\mathbb{Z}/N\mathbb{Z})^*$ with the maximal order $\lambda(N)$, the value of this latter can be deduced by calculating the order of the elements of $(\mathbb{Z}/N\mathbb{Z})^*$ that does not exceed the defined upper bound. That way, the value of λ is the largest calculated order. The procedure of calculating the value of λ is formalized in the algorithm presented next.

```
procedure GET-EXPONENT(N : BigInteger)
    i, upperBound, lambda : BigInteger;
    upperBound ← ⌊6 log(N)⌋;
    lambda ← 2;
    i ← 2;
    while i ≤ upperBound do
        if GCD(i, N) = 1 ∧ i^{lambda} mod N <> 1 then
            lambda ← Get-Order(i, N, lambda);
        end if
        i ← i + 1;
    end while
    function GET-ORDER(m, N, lambda : BigInteger)
        msg, k : BigInteger;
        finish : boolean;
        finish ← false;
        k ← lambda + 2;
        while finish = false do
            msg ← m^k mod N;
            if msg = 1 then
                finish ← true
            else
                k ← k + 2;
            end if
        end while
        return k
    end function
end procedure
```



(a) Moduli from 1 up to 500000.

(b) Moduli from 500000 up to 1000000.

Fig. 1: Upper bound and LPLR of integer Moduli in the interval $[1, 10^6]$.

TABLE I: Comparison of the Cycling and the proposed attacks using different modulus sizes.

		Cycling Attack			Proposed Attack		
Public key (N,e)	Key Size (bits)	Decryption key	Decryption key Size (bits)	Execution time in seconds	Decryption key	Decryption key Size (bits)	Execution time in seconds
$N = 2623$, $e = 65537$	12	95797049048725827802717724647094688568252151173414913	177	0.168	173	08	0.005
$N = 5251$, $e = 65537$	13	1109524890182450626553175509403290672795204402888945414153983576364372708728756982772451667049361064901874787760895221297741 7388033	433	0.509	1121	11	0.012
$N = 10379$, $e = 65537$	14	2583235108630990766845404255753590803953026769173270625397365785836726452699218686445063971128250229143645458986914349057	401	0.72	1601	11	0.015
$N = 25019$, $e = 65537$	15	2991912254793377750080816210039260247844264168504877281465871239436802824035546750356036689152386489154349403445622258582434799410723006932097878347903869984015691168006394699525441587965431775887737	657	1.742	341	09	0.008
$N = 40301$, $e = 65537$	16	3775978461112251315854025031446376184879598824395443494554724432493760581282996076711578842187175991014650395403407406636341717618919542739065999872350133412665999163393	561	21.103	3113	12	0.016

The algorithm requires the input of only one parameter which is the RSA public modulus N to provide the exponent of the multiplicative group $\mathbb{Z}_N^*$ given by the Carmichael function $\lambda(N)$. The exponent is the maximal order $\lambda(m)$ where $m \in \mathbb{Z}_N^*$ and is always an even number. The function "*Get-Order*" provides the multiplicative order modulo N of the elements m of $\mathbb{Z}_N^*$ belonging to the finite interval $[2,\ 6\log(N)]$. In order to speed up the computational time of $\lambda(N)$ and thus improve the performance of our attack, we tried to reduce the number of elements for which we calculate the order. To achieve that we must check, before each call of the function *Get-Order*, the congruence $m^{lambda} \equiv 1 \mod N$, where *lambda* is the last saved maximum order and m is the element for which we want to calculate the order.

In the case, where the congruence is satisfied, we do not need to compute the order of m since *lambda* represents either the order of m or a multiple of this latter. But, in the opposite case, we conclude that $\lambda(N)$ is not yet reached. And therefore, we must call the function *Get-Order* again to determine the order or a multiple of the order of m. In order to quickly reach

the exponent $\lambda(N)$, we call *Get-Order* using the last stored maximum order *lambda* as one of its input parameters.

Once the value of $\lambda(N)$ is obtained, we get to the last step of our attack process which consists in calculating the value of the trapdoor derived by $e^{-1} \mod \lambda(N)$. This can easily be done using the Extended Euclidean Algorithm. We note that recent RSA standards use the Carmichael function $\lambda(N)$ instead of the Euler totient function $\varphi(N)$ used in the original definition of the RSA key generation process. And thus, the value of the trapdoor would be different from that computed by $e^{-1} \mod \varphi(N)$. However, both of them allow inverting the RSA encryption function by decrypting any ciphertext in the RSA modulus N.

Through the implementation of our proposed attack and the cycling attack presented in [10] and thanks to all the tests [1] we did and the obtained results (some of them are depicted in Table I for illustration), we noted that :

- The required time to perform an RSA cryptanalysis increases with the growth of the modulus size and this whatever the used method (cyclic method or our proposed method).
- Because of the reduced size of the trapdoor found by our proposed attack, the decryption time is realistic which is not the case for the cyclic attack where the new discovered trapdoor is of an important size which implies problems of storage and calculation and thus a considerable decryption time.
- Our attack is more efficient than the attack presented in [10] because it allows to find the original trapdoor itself in a reasonable time compared to the cyclic attack which finds a new trapdoor different from the original one in a longer period.
- Applying the OAEP padding before the RSA encryption process, as described in [13], provides the highest level of security to the RSA system (the full proof is given in [14]) which prevents weaknesses due to the RSA function properties. Therefore, many attacks on the RSA system including the most dangerous ones are successfully defeated using this proper padding, excepting the attack by factorization which is the only generic attack that can not be thwarted using an OAEP padding step. Thus, it is important to remark that our proposed attack is a new attempt to break the RSA cryptosystem that cannot be avoided using the proper OAEP padding.
- The proposed attack is applicable even for the multiprime RSA as defined in [15], where the modulus $n = \prod_{i=1}^{i=u} r_i,\ i \geqslant 2$ and r_i are distinct odd prime factors ($r_1 = p$ and $r_2 = q$).

VI. Conclusion and perspectives

In this paper, we categorized attacks on the RSA system into two categories: (1) specific attacks that exploit specific weaknesses resulting from the misuse of the RSA system or the bad choice of its parameters during its implementation, (2) generic attacks that focus on the RSA algorithm itself without relying on specific implementations or cases in order to be applied. We specified that the second category of attacks is the most dangerous on the RSA system.

Based on a novel upper bound for the least primitive λ-root modulo the RSA composite modulus N, we proposed a new generic attack on the RSA system. The interest of this upper bound is that it depends only on the public modulus N and thus, it is the first upper bound of its kind that allows to avoid an exhaustive search of $\lambda(N)$ which calculates the order of all the elements of $\mathbb{Z}_N^*$. The proposed attack allows to invert, in a realistic time, the RSA encyption function by uncovering the trapdoor d (private key) derived by $e^{-1} \mod \lambda(N)$. This cryptanalysis presents also the advantage of being resistant to an OAEP padding.

The attack presented in this paper leaves a promising avenue open for future research. Thus, in perspective and in order to show the interest of our attack :

- Enhancing the search time of the maximum order $\lambda(N)$ by additional restrictions on the number of considered elements to calculate the order.
- Elaborating a comparative study between our attack and the attack by factorization.

[1] These tests are carried out on a machine with the following characteristics: TOSHIBA, SATELLITE C855-1KM, InsydeH2O version 03.72.016.10, Intel(R) Core(TM) i3-2310MCPU@2.10GHz(4CPUs), 2.1GHz, 4096MB RAM, Windows 7 Professional 32 bits (6.1, version 7601).

References

[1] Diffie Whitfield, and Martin Hellman. "New directions in cryptography." IEEE transactions on Information Theory 22.6 (1976), pp. 644-654.

[2] Shannon, Claude E. "A mathematical theory of cryptography." Mathematical Theory of Cryptography (1945).

[3] Boneh, Dan. "Twenty years of attacks on the RSA cryptosystem." Notices of the AMS 46.2 (1999): 203-213.

[4] J. P. Buhler, H. W. Lenstra, and C. Pomerance. "Factoring integers with the number field sieve." The development of the number field sieve. Springer, Berlin, Heidelberg, 1993. 50-94.

[5] Brent, Richard P. "Recent progress and prospects for integer factorisation algorithms." International Computing and Combinatorics Conference. Springer, Berlin, Heidelberg, 2000.

[6] Lenstra, Arjen K. "Integer factoring." Towards a quarter-century of public key cryptography. Springer, Boston, MA, 2000. 31-58.

[7] Friedlander, John, Carl Pomerance, and Igor Shparlinski. "Period of the power generator and small values of Carmichaels function." Mathematics of Computation 70.236 (2001): 1591-1605.

[8] Nemec, Mat. The properties of RSA key generation process in software libraries. Diss. Masarykova univerzita, Fakulta informatiky, 2016.

[9] Rivest, Ronald L., and Robert D. Silverman. "Are strong primes needed for RSA". The 1997 RSA Laboratories Seminar Series, Seminars Proceedings. 1997.

[10] Derdouri, Lakhdar, and Noureddine Saidi. "Decrypting the ciphertexts of RSA cryptosystem with ciphering function." Sciences of Electronics, Technologies of Information and Telecommunications (SETIT), 2012 6th International Conference on. IEEE, 2012.

[11] Martin, Greg. "The least prime primitive root and the shifted sieve." arXiv preprint math9807104 (1998).

[12] Ambrose, Christopher. "On the least primitive root expressible as a sum of two squares." Mathematisches Institut (2013).

[13] Bellare, Mihir, and Phillip Rogaway. "Optimal asymmetric encryption." Workshop on the Theory and Application of of Cryptographic Techniques. Springer, Berlin, Heidelberg, 1994.

[14] E. Fujisaki, T. Okamoto, D. Pointcheval, J. & Stern. "RSA-OAEP is secure under the RSA assumption". Journal of Cryptology, 17(2), pp.81-104. 2004.

[15] K. Moriarty, B. Kaliski, J. Jonsson, and A. Rusch. "PKCS# 1: RSA Cryptography Specifications Version 2.2". November 2016.