Discrete Logarithm Attack against RSA Cryptosystem

Imad Salah 1, Abdel-Salam Obeidat1, Abdullah Darwish2 and Saleh Oqeili3

Received on June 25, 2006 Accepted for publication on Feb. 5, 2007

Abstract

In this paper Discrete Logarithm Attack (DLA,) as a new class of attacks against the Rivest, Shamir, and Adleman (RSA) cryptosystem, is proposed. This attack is derived from geometric properties of Euler and Carmicheal functions and depends on solving Discrete Logarithm Problem (DLP). The DLA enables the recovery of plaintext messages from their cipher texts and recovery the factors of RSA modulus. At the end, some numerical examples and experimental results are given to explain the proposed attack.

Keywords: RSA, Discrete Logarithm, LFSR, low-exponent, One-Way function.

1. Introduction

We begin by describing a simplified version of RSA encryption. Let N = p*q be the product of two large primes of the same size. Let e, d be two integers satisfying e*d = 1 mod Ø(N) where Ø(N) = (p-1)*(q-1) is the Euler’s totient function of N. We call N the RSA modulus, e the encryption exponent, and d the decryption exponent. The pair (N, e) is the public key, which is used to encrypt messages. The pair (N, d) is called the private key and is known only to the recipient of encrypted messages. To encrypt a message M, one computes C = M^e mod N. To decrypt the cipher text C, the legitimate receiver computes C^d mod N.

Many researchers and attackers are interested in attacking the RSA Cryptosystem by using different types of attacks and tricks [1]. In this paper, we present a new attack to break RSA Cryptosystem, which is called Discrete Logarithm Attack (DLA). This attack is based on solving Discrete Logarithm Problem (DLP) to attack the RSA Cryptosystem.

This paper is organized as follows: an Euler and Carmicheal theorems as basic facts of our approach are reviewed. Discrete Logarithm Problem (DLP) is defined. Two algorithms for solving DLP are suggested. After that, we present a first part of Discrete Logarithm Attack, which is used to recover an original message of the RSA
Cryptosystem, and the second part attack, which is used to recover factors of the RSA modulus N. At the end, we display experimental results of our proposed attack.

2. Supporting Theorems

The security of many cryptographic techniques depends on the intractability of Discrete Logarithm Problem (DLP). Like the Factoring Problem, the DLP is believed to be difficult and the hard direction of a one-way function. For this reason, it has the basis of several public-key cryptosystem. The DLP bears the same relation to these systems as factoring does to the RSA system: the security of these systems rests on the assumption that Discrete Logarithm (DL) is difficult to compute. Before giving a scientific definition of DLP, some important theorems and definitions should be reviewed.

**Theorem 1. (Euler’s theorem).** Let a and n be positive integers with $\text{GCD}(a,n)=1$, then $a^{\varphi(n)} \equiv 1 \pmod{n}$.

The proof of this theorem can be found in [10]. Here, GCD(a,n) is the Greatest Common Divisor of a and n and $\varphi(n)$ is the Euler’s totient function of n.

**Definition 1.** If $n>1$ and $\text{GCD}(a,n)=1$, then the order of a modulo n is the smallest positive integer x for which $a^x \equiv 1 \pmod{n}$.

The order x of a modulo n is well defined, by Euler’s theorem, $a^{\varphi(n)} \equiv 1 \pmod{n}$ when $\text{GCD}(a,n)=1$, so that $1 \leq x \leq \varphi(n)$.

**Definition 2.** An integer a whose order modulo n is $\varphi(n)$ is called a primitive root modulo n.

If a is primitive root modulo n, then $\text{GCD}(a,n)=1$, because the order of a would be undefined if $\text{GCD}(a,n)>1$. Notice that some positive integers have primitive roots and some do not. Given the RSA public key $(N,e)$, Euler’s theorem says that: $a^x \equiv 1 \pmod{N}$, where $x=\varphi(N)$ and $\text{GCD}(a,N)=1$. The Euler’s theorem does not mean that x is the smallest value for which the equality is true. In other words, there are multi values less than x for which the equality is true. The following function, first proposed by Carmichael, is a very useful number theoretic function.

**Definition 3. (Carmichael’s $\lambda$-function).** $\lambda(N)$ function is defined as follows:

- $\lambda(p)=\varphi(p)=p-1$ for prime p,
- $\lambda(p^a)=\varphi(p^a)$ for $p=2$ and $a \leq 2$, and for $p \geq 3$,
- $\lambda(2^a)=1/2^a\varphi(2^a)$ for $a \geq 3$,
- $\lambda(N)=\text{lcm}(\lambda(p_1^{a_1}), \lambda(p_2^{a_2}), \ldots, \lambda(p_k^{a_k}))$ if $N=\prod_{i=1}^{k} p_i^{a_i}$.
Theorem 2. (Carmichael’s Theorem). Let \( a \) and \( n \) be positive integers with \( \text{GCD}(a,n)=1 \). Then \( a^{\lambda(N)} \equiv 1 \mod n \), where \( \lambda(N) \) is Carmichael’s function. The proof of this theorem can be found in [10].

So, it can be difficult to find Euler’s function of an element \( a \) modulo \( n \), but sometimes it is possible to prove that every integer \( a \) modulo \( n \) must have an order smaller than the number \( \Theta(N) \). This order is actually the number \( \lambda(N) \) or multiplies of it.

Example 1. Let \( a=11 \) and \( n=65520=2^4 \cdot 3^2 \cdot 5 \cdot 7 \cdot 13 \), then \( \text{GCD}(65520,11)=1 \), and we have: \( \Theta(65520)=8 \cdot 6 \cdot 4 \cdot 6 \cdot 12 = 13824 \), \( \lambda(65520)=\text{lcm}(4,6,4,6,12)=12 \). Note that \( 11^{13824} \mod 65520=1 \) and \( 11^{12} \mod 65520=1 \).

Definition 4. Let \( a \) be primitive root modulo \( n \). If the integer \( b \) is relatively prime to \( n \), then there is an integer \( k \) such that \( a^k \equiv b \mod n \) and \( 0 \leq k \leq \Theta(n) \), this integer \( k \) is called the Discrete Logarithm of \( b \) to base \( a \) modulo \( n \).

In notation language, we write \( \log_a b = k \mod n \). The DLP applies to mathematical structures called groups. A group is a collection of elements together with a binary operation, which are referred to as group multiplication.

Our task is to find Carmichael’s function \( \lambda(N) \) or multiplies of it in order to recover an original message \( M \) which is sent in Discrete Logarithm Attack Part I, and to find an Euler’s function \( \Theta(N) \) in order to recover prime factors of RSA modulus \( N \) in Discrete Logarithm Attack Part II. Next, we suggest two approaches to solve Euler and Carmichael functions in order to solve DLP.

3. Solving Discrete Logarithm Problem (DLP)

In this section, we present two approaches to solve Discrete Logarithm Problem (DLP). In other words, finding \( x \) in the Euler’s equation: \( a^x \equiv 1 \mod N \), where \( N \) is RSA modulus and \( \text{GCD}(a,N)=1 \). As we mentioned before, \( x \) can be Euler’s function or Carmichael’s function \( \lambda(N) \) or fractions of \( \lambda(N) \).

3.1 Linear Feedback Shift Register (LFSR) approach

The LFSR is a new method for solving Euler’s theorem by using simple Add, Shift, and Compare binary operations to find \( x \) such that: \( 2^x \equiv 1 \mod N \), for \( a=2 \) and RSA modulus \( N \), where \( \text{GCD}(2,N) \) is always 1, since \( N \) is always odd number. The basic idea of this method is to find a number \( Z \) such that: \( Z \cdot N \) all binary bits are equal to one. For some value of \( x \) there is: \( 2^x-1=Z \cdot N \).

For some value of \( Z \), a solution \( 2^x-1 \) is represented in binary as:

\[
\begin{align*}
1 \\
11 \\
111 \\
1111 \\
11111 \\
\end{align*}
\]
The LFSR algorithm is shown in algorithm 1. It finds such a multiply by simple binary operations. The algorithm sets bits of an accumulator to one by adding shifts of the modulus N, working from Least Significant Bit (LSB) to Most Significant Bit (MSB) of the accumulator. Eventually, the accumulator is all 1’s and the number of 1’s yields a divisor of $\phi(N)$.

**Algorithm 1. (LFSR algorithm).**

- **Input:** $N$ is a binary representation of RSA modulus.
- **Output:** The discrete logarithm $x = \log_2 1 \mod N$.
- **Algorithm:**
  1. Set $z=0$; $j=1$;
  2. While $z$ contains a 0 bits do
    // The first 0 starting from the right is denoted by $j$, shifting $N$ to the right by the continuous number of ones at the right side is denoted by $N<<(j-1)$.
    2.1. If bit $j$ of $z$ is 0 then $z=z+(N<<(i-1))$;
    2.2. $j=j+1$;
  3. Return the number of ones in binary representation of $z$.
- **End Algorithm 1. (LFSR algorithm).**

**Example 2. (LFSR algorithm).**

Let $N=35$, then to find $x$ in equation $2^x \equiv 1 \mod 35$ do the following:

Representation of 35 in binary is 100011, then

\[
\begin{array}{c}
100011 \\
10001100 \\
10101111 \\
1000110000 \\
1011011111 \\
10001100000 \\
11100111111 \\
100011000000 \\
111111111111 \Rightarrow \text{yields } x=12, \text{ so, } 12 = \log_2 1 \mod 35.
\end{array}
\]
Discrete Logarithm Attack against RSA Cryptosystem

Note that, \( \varphi(35) = 24 \) and \( \lambda(35) = 12 \). As shown, a product of 35 times \( Z \) is developed from right to left forcing the low order bits to one, the algorithm terminates, when the bits are all one. The LFSR algorithm depends on simple binary operations: Add, Shift and Compare operations. Let \( n \) be a number of bits in RSA modulus \( N \), then the total number of operations is needed to achieve LFSR approach is \( O(2^n) \) operations. Unfortunately, this algorithm is inefficient when modulus \( N \) is large, since finding \( x \) can take up to \( \varphi(N) \) steps, since \( \varphi(N) \) is almost as large as modulus \( N \). we can improve the LFSR approach by running a similar approach works left to right. Therefore, the two algorithms can work simultaneously and meet in the middle of the sequence of even numbers. So the maximum number of steps is \( O(N)/2 \) not \( O(N) \).

3.2 Modified Shank’s algorithm

In this section, we discuss Shank’s algorithm for computing discrete logarithm, and how can we modify it to be applicable to breaking RSA cryptosystem, by solving Euler’s theorem. Modified Shank’s algorithm is shown in algorithm 2.

**Definition 5.** Let \( m = \left\lfloor \sqrt{N} \right\rfloor \), where \( N \) is the RSA modulus. If \( x = \log_a N \), then we can uniquely write \( x = i + jm \), where \( 0 \leq i, j < m \).

**Algorithm 2. (Modified Shank’s algorithm).**

1. Set \( m = \left\lfloor \sqrt{N} \right\rfloor \).
2. Construct table with entries \((j,a^j)\) for \( 0 \leq j < m \). Sort this table by second component.
3. Compute \( a^{-m} \) mod \( N \), set \( \beta = b \).
4. For \( i = 0 \) to \( m-1 \) do
   1. Check if \( \beta \) is the second component of some entry in the table.
   2. If \( \beta = a^i \) then return \( (x+i)m+j) \).
   3. Set \( \beta = \beta * a^m \) mod \( N \).
5. Return (“failure”).

**End Algorithm 2. (Modified Shank’s algorithm).**

Example of Modified Shank’s algorithm can be found in [5].

A new modification of original Shank’s algorithm to compute discrete logarithm is by using hash table method for constructing steps 2 and 4 of Algorithm 2. Hash table contains two entries key and value, so, the hashing is done on the second component and
the resulting memory address checked. Hash tables can retrieve and add elements in constant time. Modified Shank’s algorithm allows computing discrete logarithm for relatively large modulus $N$ (at most 50 decimal digits). Unfortunately, Modified Shank’s algorithm requires $O(\sqrt{N})$ storage. The table takes $O(\sqrt{N})$ multiplications to construct, and $O(\sqrt{N} \log N)$ comparisons to sort, so, the running time for Modified Shank’s algorithm is $O(\sqrt{N})$, where $N$ is RSA modulus. So this method is inefficient for very large values of $N$ compared with general-purpose factoring methods.

4. Discrete Logarithm Attack (DLA)

In this section, we produce a Discrete Logarithm Attack (DLA) against the RSA cryptosystem as new link between discrete logarithm and integer factorization problems. This attack is based on solving DLP of Euler’s theorem. In other words, finding $x$ in equation: $a^x \equiv 1 \pmod N$, where $N$ is the RSA modulus for any given number $a$ such that $\gcd(a, N) = 1$. From the RSA definition, $\varphi(N) = (p-1)(q-1)$ for two primes $p$ and $q$. $\varphi(N)$ is always even number (i.e. $\varphi(N)$ is divisible by 2), so, the general form of $\varphi(N)$ is $2^k$, where $k$ is unknown large integer.

The question here is “Can we break the RSA without factoring modulus $N$?”. In other words, “Can we compute $\varphi(N)$ or private exponent $d$ without factoring the RSA modulus $N$?”. It is known that if an attacker could compute $\varphi(N)$, then he/she could break the whole system by computing private exponent $d$ as the multiplicative inverse of public exponent $e$ modulo $\varphi(N)$. However, the knowledge of $\varphi(N)$ can lead to an easy way of factoring $N$, since $p$ and $q$ are roots of the following quadratic equation:

$$z^2 - (\varphi(N) + 1)z + N = 0$$

Thus, breaking the RSA cryptosystem by computing $\varphi(N)$ is not easier than breaking the cryptosystem by factoring $N$. In fact, Rivest and others [6] conjectured that any method of breaking the RSA cryptosystem must be as difficult as factoring.

In our approach, we are trying to prove that there is an algorithm to solve Euler’s theorem without needing to factor $N$, i.e., solving DLP in feasible way caused breaking RSA cryptosystem easily. As we mentioned before, we produce a DLA in two parts, the first part is concerned with recovering the original message $M$ which is sent (i.e. partial broken) by finding Carmichael’s function $\lambda(N)$ or multiplies of it, the second part concerns with recovering private key $d$ (or factors of $N$) by finding Euler’s totient function $\varphi(N)$ (i.e. total broken).

4.1 Discrete Logarithm Attack Part I

The primary objective of an adversary here is to systematically recover plaintext from cipher text intended for some other entity $A$. If this is achieved, the encryption scheme is informally said to have been broken. Discrete Logarithm Attack Part I algorithm for recovering an original message $M$ is shown in algorithm 3.
Algorithm 3. (DLA Part I).

- **Input:** RSA public key \((N,e)\), and cipher text \(C\).
- **Output:** An original message \(M\).
- **Algorithm:**
  1. Find \(x\) in the equation \(2^x \equiv 1 \mod N\) (or equation \(e^x \equiv 1 \mod N\)) by using LFSR (Algorithm 1.) or Modified Shank’s (Algorithm 2.).
  2. Compute \(d \equiv e^{-1} \mod x\).
  3. Recover an original message \(m = C^d \mod N\).

**Example 3. (DLA Part I).**

Let \(N=5515596313\), \(e=65537\) and \(C=9726^{65537} \mod 5515596313 = 4199456257\). First step, attacker should compute \(2^x \equiv 1 \mod 5515596313\), applying Modified Shank’s algorithm yields \(x=114905160\). Then compute \(d = 65537^{-1} \mod 114905160 = 5055213\). At last step, recover original message \(M = 4199456257^{5055213} \mod 5515596313 = 9726\).

The major task of DLA Part I is in the first step, which is a hard task if RSA modulus \(N\) becomes large. The expected running time of DLA Part I is \(O(\sqrt{N})\) time, which is bad when \(N\) is large. More experimental results can be shown later.

4.2 Discrete Logarithm Attack Part II

The primary objective of an adversary here is the private key recovery. If this is achieved, then encryption scheme is informally said to have been completely broken, since the adversary has the ability to decrypt all cipher texts sent.

In the second part of Discrete Logarithm attack, we are trying to recover exact value of \(\phi(N)\). However, the knowledge of \(\phi(N)\) can lead to an easy way of factoring modulus \(N\), so the whole system is broken.

The basic idea behind the DLA Part II is using a Modified Shank’s algorithm to find \(x\) in \(e^x \equiv 1 \mod N\), where \((N,e)\) is RSA public key, then we use Exhaustive Search technique to find an exactly value of \(\phi(N)\) after minimizing the number of iterations as possible. Algorithm 4 presents DLA part II.

Algorithm 4. (DLA Part II).

- **Input:** Given RSA public key \((N,e)\).
- **Output:** A prime factors \(p\) and \(q\) of RSA modulus \(N\).
- **Algorithm:**
  1. Find \(x\) in the equation \(e^x \equiv 1 \mod N\), using Modified Shank’s algorithm.
2. Compute \( \delta = \left( \left( N - 1 \right) - 2 \sqrt{N - 1} \right) / x \).

3. For \( i = \delta \) downto 0 do
   3.1. Let \( p \) and \( q \) be the roots of the equation:
   \[ z^2 - (N - i^2 + 1)z + N = 0. \]
   3.2. If \( p \) and \( q \) are positive integers then return \((p, q)\).

End Algorithm

Example 4. (DLA Part II).

Let \( N = 5515596313 \), \( e = 65537 \). First step, attacker should compute \( 65537^x \equiv 1 \mod 5515596313 \), applying Modified Shank’s algorithm yields \( x = 6383620 \). Then compute \( \delta = \left( \left( (5515596312) - 2 \sqrt{5515596312} \right) / 6383620 \right) = 864 \). Then, solve the quadratic equation:
\[ z^2 - 148634z + 5515596313 = 0, \]
yields \( p = 71593 \) and \( q = 77041 \). Thus, \( \varphi(5515596313) = 5515447680 \), and private key \( d = 65537^{-1} \mod 5515447680 = 1875455873 \). Attacker recovers all messages that are encrypted.

The major task of Discrete Logarithm Attack II is using Modified Shank’s algorithm to solve a DLP in equation: \( a^x \equiv 1 \mod N \). It cannot use LFSR approach to solve Euler’s equation since \( a \) does not equal to \( 2 \).

The expected running time of DLA Part II is also \( O(\sqrt{N}) \) time, and needs \( \sqrt{N} \) storage, which is bad when \( N \) is large.

5. Experimental Results

In this section, we introduce the experimental results of the proposed Discrete Logarithm Attacks and solving Discrete Logarithm algorithms. The algorithm is implemented in JAVA language by using multi methods and techniques, such as hash table and BigInteger classes. We measure the performance of the proposed algorithm according to execution time, storage requirements, and number of operations. We run the above algorithms on Pentium 360, 833 MHz CPU with 256 MB RAM computers under Windows 2000 OS.

5.1 Experimental Results for LFSR algorithm

The proposed LFSR algorithm to solve \( 2^x \equiv 1 \mod N \), where \( N \) is RSA modulus, is new approach to recover an original plaintext of the RSA Cryptosystem. The “new” here is using two and simple binary operations to find \( x \). Add, Shift, and Compare binary operations are fundamental functions of a computer, so, we believe that this approach is faster than multiplication or division factoring approach. Unfortunately, the number operations involved is in the order \( O(2^n) \), where \( n \) is the number of bits in binary representation of the RSA modulus \( N \). Although each operation is very fast because it is binary and simple, the number of operations is enormous.

The amount of work required for the LFSR algorithm to complete is a linear function of the number of bits in the RSA modulus as Figure 1 shows.
Discrete Logarithm Attack against RSA Cryptosystem

![Figure 1: Complexity of LFSR approach](image1)

The LFSR approach is faster by about $2\log N$ than trial division factoring method, but it is slower than NFS, QS, and EC factoring methods as shown in Figure 2.

![Figure 2: Comparison between LFSR approach and Trial Division algorithm](image2)

In fact, there is no reason to think that finding $x$ in equation $2^x \equiv 1 \mod N$ using the LFSR approach is easier than directly finding the factors $p$ and $q$ of modulus $N$.

5.2 Experimental Results for Modified Shank’s algorithm

Modified Shank’s algorithm finds $x$ in the expression $a^x \equiv 1 \mod N$, where $N$ is RSA modulus. We choose two values for $a$ to test the performance of Modified Shank’s algorithm. We choose $a=2$ as default choice of the LFSR algorithm, and $a=65537$ as
most common value for public exponent e in RSA cryptosystem. Table 1 shows the resultant values of x from running Modified Shank’s algorithm.

Table 1: Modified Shank’s algorithm results with a=2 and a=65537

<table>
<thead>
<tr>
<th>N</th>
<th>a</th>
<th>a</th>
</tr>
</thead>
<tbody>
<tr>
<td>91</td>
<td>2</td>
<td>65537</td>
</tr>
<tr>
<td>11413</td>
<td>700</td>
<td>2800</td>
</tr>
<tr>
<td>5515596313</td>
<td>114905160</td>
<td>6383620</td>
</tr>
<tr>
<td>2367216591499337</td>
<td>73995515196008</td>
<td>24658505065336</td>
</tr>
<tr>
<td>32254013411746110071</td>
<td>2687834449900109916</td>
<td>4031751674850164874</td>
</tr>
<tr>
<td>331232550936322409694169</td>
<td>82808137733790528267124</td>
<td>82808137733790528267124</td>
</tr>
</tbody>
</table>

Note that the time needed to compute x when a=65537 is less than the time needed when a=2. Figure 3 shows this result.

Figure 3: Modified Shank's algorithm time with inputs a=2 and a=65537

As a default result, when the size of modulus N increases, the time needed to compute discrete logarithm also increases. Figure 4 shows this result. Note that at size of modulus greater than 45 decimal digits, Modified Shank’s algorithm fails.

362
Discrete Logarithm Attack against RSA Cryptosystem

Figure 4: Complexity of Modified Shank’s algorithm

Modified Shank’s algorithm computes the discrete logarithm of an element of a finite cycle group with complexity about $O(\sqrt{N} \cdot \log N)$ group operations and $O(\sqrt{N})$ space, where $N$ is RSA modulus. So, this method is not efficient for very large values of $N$.

5.3 Experimental Results for Discrete Logarithm Attack

The DLA can be applied to some private-key or public-key systems, like Pohlig-Hellman or RSA, or to a key exchange protocol, like Diffie-Hellman. One day faster algorithms for discrete logarithms might be discovered. If that happen, these systems may be out of business or one may have to use larger groups.

The time needed to solve the problem depends mostly on the size of the group, although it might be easy for special parameters. Usually, the size of the group is chosen large enough so that the problem cannot be solved in reasonable time. But if it is chosen too large, then the algorithm speed will suffer.

The first step of DLA is finding $x$ in equation $a^x \equiv 1 \pmod{N}$, which is very hard task when RSA modulus $N$ becomes larger. As the size of modulus $N$ increases the required time to break RSA cryptosystem also increases.

Unlike other attacks on RSA Cryptosystem, the DLA does not put any preconditions on public exponent $e$ or private key $d$. Compared with Low Public Exponent Attacks and Low Private Exponent Attacks [1], DLA does not care about the values of $e$ and $d$ like.

The first part of DLA recovers an original plaintext that was encrypted using RSA public keys by generating $d'$ private key. As was mentioned before, $d'$ is not necessarily equal to original private key $d$. This means that other keys can decipher the cipher text. If this is achieved, the encryption scheme is informally said to have been broken. DLA Part I is better than Chosen Cipher text Attack, since the Chosen Cipher text Attack requires more decryptions with each candidate key to identify the expected clear text statistics.
When applying Chosen Cipher text Attack to RSA scheme, the time complexity grows cubically with the size of modulus $N$. Figure 5 shows this result. Also, unlike Chosen Cipher text Attack, DLA Part I does not require any response from the victim to compute any function for the attacker.

![Figure 5: Comparison between Discrete Logarithm Attack and Chosen Cipher text Attack](image)

The second part of Discrete Logarithm Attack is more ambitious than the first part since the whole system is broken and the attacker can decipher all messages sent. The difference between $\Theta(N)/x$ and delta function increases as the size of modulus $N$ increase. Table 2 shows this result, assume $\Delta = \text{delta-} \Theta(N)/x$.

<table>
<thead>
<tr>
<th>$N$</th>
<th>$x=65537^x \mod N$</th>
<th>$\Theta(N)/x$</th>
<th>delta</th>
<th>$\Delta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>91</td>
<td>6</td>
<td>12</td>
<td>12</td>
<td>0</td>
</tr>
<tr>
<td>11413</td>
<td>2800</td>
<td>4</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>5515596313</td>
<td>6383620</td>
<td>864</td>
<td>864</td>
<td>0</td>
</tr>
<tr>
<td>32254013411746110071</td>
<td>4031751674850164874</td>
<td>8</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>3312322550936322409694169</td>
<td>82808137733790528267124</td>
<td>4</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>714634754475713249199293731673</td>
<td>119105792412618589831425751590</td>
<td>6</td>
<td>9</td>
<td>3</td>
</tr>
</tbody>
</table>

Exhaustive Search algorithm can be improved by reducing the number of iterations in the inner loop. An ordinal Exhaustive Search algorithm starts from 1 to $N-1$ to find a number $y$ such that $z^2-(N-y+1)z+N=0$, so its complexity is $O(N)$ in the worst case. While DLA Part II decreases the time needed by $O(\sqrt{N})$, Figure 6 shows this result.
As we saw, the main task here is to find $x$ in the expression $e^x \mod N=1$, this is very hard computation when $N$ becomes large.

6. Conclusions

In this paper, the DLA is proposed as a new class of attacks against RSA Cryptosystem. It was shown that solving DLP causes breaking RSA easily. We present two algorithms to solve an Euler’s function, the first is LFSR algorithm that needs $O(2^n)$ binary operations, where $n$ is the number of bits in the binary representation of the RSA modulus $N$. The second algorithm is Modified Shank’s algorithm which solve DLP in $O(\sqrt{N})$, which is bad when $N$ becomes large.

The DLA Part I recovers the original message by solving Euler’s function using one of above algorithms, then computes an alternative private key to recover an original message in an efficient way. DLA Part II was proposed to recover an exact value of Euler’s function $\Omega(N)$ in order to break the whole system. An improvement to Exhaustive Search method was introduced by minimizing the number of iterations nearly to a half. Unlike other RSA attacks, the DLA does not depend on the size of the public and private exponents. To protect RSA Cryptosystem against DLA, additional bits to the original message prior to encryption can be added. For now at least, RSA is completely safe.
طريقة هجوم اللوغاريتمات المتقطعة ضد نظام التشفير RSA

عماد صلاح، عبد السلام عبید، عبد الله درويش وصالح العقيلي

ملخص

قدت ورقة البحث هذه طريقة هجوم اللوغاريتمات المتقطعة كطريقة جديدة لاختراق نظام التشفير RSA والذي يعتبر من أشهر أنظمة التشفير وأسراه انتشاراً. و窃 هذى الطريقة المقتطعة على حل مشكلة اللوغاريتم المتقطع بالاعتماد على بعض الاقتراحات الحسابية الهامة. وتم مقارنة هذه الطريقة مع طرق المهاجمة السابقة وأثبتت فعاليتها وقوتها على استرجاع النص الأصلي من نص مشفرب بواسطة نظام التشفير RSA. مع وجود بعض الانتقادات في هذه الطريقة المقتطعة.

أثبتت هذه الورقة أنه لا يمكن التوقف على نظام التشفير على مستوى بعيد، فعلى سبيل المثال، نظام التشفير RSA لم يثبت - وللغاية الآن - بأنه لا يمكن اختراقه، ولكن مع هذا ووجود عدد من الطرق التي حاولت كسر نظام التشفير RSA. يمكن الاعتراف بأن هذا النظام أثبت فعاليته وإمكانية استخدامه في التطبيقات المختلفة لعدة سنوات قادمة.

References