Ciphertext-Only Attack on RSA Using Lattice Basis Reduction

Anas Ibrahim[1,2], Alexander Chefranov[1], and Rushdi Hamamreh[3]
[1]Computer Engineering Department, Eastern Mediterranean University, North Cyprus
[2]Computer Engineering Department, Palestine Technical University, Palestine
[3]Computer Engineering Department, Al-Quds University, Palestine

Abstract: *We use lattice basis reduction for ciphertext-only attack on RSA. Our attack is applicable in the conditions when known attacks are not applicable, and, contrary to known attacks, it does not require prior knowledge of a part of a message or key, small encryption key, e, or message broadcasting. Our attack is successful when a vector, comprised of a message and its exponent, is likely to be the shortest in the lattice, and meets Minkowski's Second Theorem bound. We have conducted experiments for message, keys, and encryption/decryption keys with sizes from 40 to 8193 bits, with dozens of thousands of successful RSA cracks. It took about 45 seconds for cracking 2001 messages of 2050 bits and for large public key values related with Euler's totient function, and the same order private keys. Based on our findings, for RSA not to be susceptible to the proposed attack, it is recommended avoiding RSA public key form used in our experiments.*

Keywords: *Ciphertext-only attack, encryption key, euler's totient function, Gaussian lattice basis reduction, RSA, shortest vector problem.*

Received May 13, 2020; accepted September 28, 2020
https://doi.org/10.34028/iajit/18/2/13

1. Introduction

We Consider Ciphertext-Only Attack (COA) on textbook RSA [38], hereafter RSA, without preprocessing of the plaintext such as Optimal Asymmetric Encryption Padding (OAEP) used in RSA standard [33]. LLL algorithm [29] of lattice basis reduction is used for COA on RSA [10, 13, 14, 19, 20, 42, 43] and other public key cryptosystems such as NTRU [21, 22, 27, 50]. Most of the attacks require either message broadcasting, or prior knowledge of a part of a message/private key. And the problem of attacking RSA is considered as a problem of finding the shortest vector Shortest Vector Problem (SVP) in a lattice dimension of which grows with the growth of the encryption exponent, e. LLL algorithm computational complexity exponentially depends on the lattice dimension [23, 24], and, hence, it solves SVP efficiently for low-dimensional lattices but the solution is infeasible for lattices with dimension greater than 400 [25]. That is why, attacks on RSA using LLL assume low encryption exponent value [31]. Herein, we propose a new line of COA on RSA using LLL [29] algorithm to solve SVP in a 2-dimensional lattice. It is based on the first found herein opportunity of RSA encryption representation in terms of 2-dimensional lattice. It requires neither message broadcasting, nor prior knowledge of a part of a message/private key contrary to all known approaches. Below, in subsections 1.1-1.3, we briefly review literature on the attacks on RSA private key and plaintext message, and introduce the paper structure attacks on RSA are reviewed, e.g., in [7, 34, 49].

1.1. Lattice-Based Attacks against RSA Private Key

In [13], prime factors of $N = p \cdot q$, used as the modulus value in RSA encryption (3) with the public key, e, and decryption (5) with the private key, d, are found as roots of a bivariate polynomial constructed using high order $\left(\frac{1}{4}+\varepsilon\right)\log_2 N$ bits of p, $\varepsilon > 2/\log_2 N$ (high order bits of q are known by division of N by p). LLL lattice basis reduction algorithm is used for dimension $r = 2k + 1, k > 1/(4\varepsilon)$. In [8, 9], LLL method is used to disclose the private key, $d < N^{\delta}, \delta = 0.292$, that extends the attack applicability compared to attack [46] assuming $\delta = 0.25$. In [45], two parameters define the attack applicability: δ, and β such that $\Delta = |p - q| = N^{\beta}$. In [45], Figure 1, specifies that known attacks on RSA are not applicable for $\beta \in [0.5, 1]$, and mainly not applicable for $\delta \in [0.5, 1], \beta \in [0.25, 0.5]$. Using β, the attack [45] extends applicability of [9] attack up to $\delta \to 1$ for $\beta \to 0.25 + \varepsilon, \varepsilon \to 0$. In [10, 42, 43], LLL algorithm is used to disclose secret RSA exponent provided that part of it (least- or most-significant bits) are known. Figures 1, 2 in [42, 43] show that δ can be extended to 0.57 and 0.37 for the use of most-and least-significant bits, respectively.

1.2. Lattice-Based COA against RSA Messages

In [14], an encrypted RSA message is disclosed as a root of a univariate polynomial of low order, e. Exponent considered in the paper is e=3 resulting in the polynomial of order, k=e=3. Message, m, to be found shall be rather small: $|m| < N^{\frac{1}{k}-\varepsilon}, \varepsilon = \frac{1}{\log N}, k < \log N$, (see section 2 in [14]). Respective lattice size is

$$\begin{aligned} Size = 2h \cdot k = k \geq 2 \cdot \frac{1}{k\varepsilon}k - k \\ = 2\log_2 N - k > 0, \end{aligned} \tag{1}$$

Where h is such that $h \cdot k \geq 7$ and $h - 1 \geq (hk-1)(\frac{1}{k} - \varepsilon)$. It is known from NTRU security requirements [25] that if the size of a lattice is, $Size \geq 400$, then LLL attack is unfeasible. Thus, from (1), it follows that already for 512-bit RSA the attack is not feasible because $\log_2 N = 512$, $k < 512$, and, hence, $Size \geq 1024 - k \geq 512$. Note that in [14], estimates of RSA parameters, such that the proposed attack is feasible, are not defined. Hastad [19] showed that the message, *m*, can be revealed in polynomial time when it is encrypted with several public keys, (e_i, N_i) each having the same public exponent, *e*, and different moduli values, $N_i, i=1, \ldots, k$, expected to be mutually relatively prime, and meeting (2):

$$\begin{gathered} m < \min_{i=1,\cdots,k} N_i, k > \frac{e(e+1)}{2}, \\ N > n^{\frac{e(e+1)}{2}}(k+e+1)^{\frac{(k+e+1)}{2}} 2^{\frac{(k+e+1)^2}{2}}(e+1)^{e+1}, \\ N = \prod_{i=1}^{k} N_i, \end{gathered} \tag{2}$$

Method [20] is practically the same as in [19] with slightly different lattice constructed, and, thus, slightly differing from 2 inequalities [20] and is also applied to a broadcasted message. The broadcasted message, *m*, is revealed by applying LLL algorithm to the lattice defined using coefficients of the polynomials resulting from the message encryption using different moduli. Also, Chinese Remainder Theorem (CRT) is used.

1.3. Non-Lattice-Based Attacks against RSA Private Keys and Messages

RSA secret key can be disclosed if the integer modulus, *N*, is factorized. Methods of integer factorization are reviewed in [2, 36], and application of one of them, Number Field Sieve (NFS), in [28] in December, 2009, resulted in factoring 768-bit RSA modulus, RSA-768. RSA moduli RSA240 and RSA250 with 795 and 829 bits were factored by NFS in December 2, 2019, and February 28, 2020, [47] which took 4000 and 2700 core-years of Intel Xeon Gold 6130 CPUs as a reference (2.1GHz), respectively [44, 51].

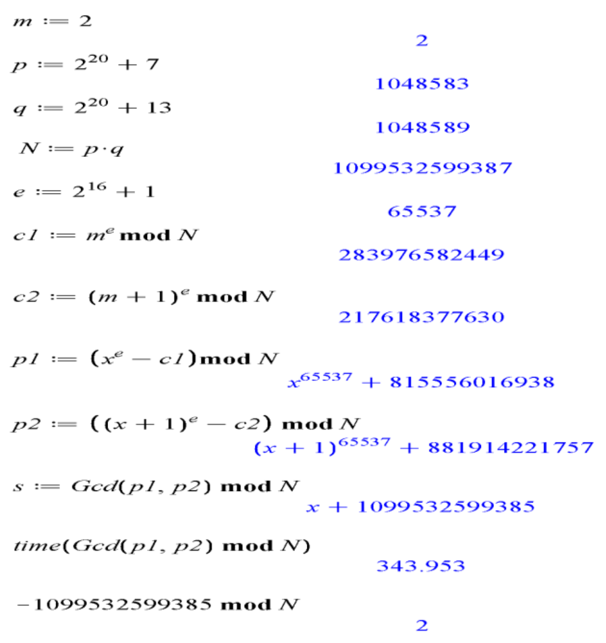
In [12], a method of factoring RSA modulus, $N = pq$, $q < p < 2q$, in time polynomial in $logN$ is proposed under assumption that an encryption exponent, *e*, meets $e \cdot x - (p^2-1)(q^2-1)y = z, gcd(y,x) = 1, z \neq 0, x \cdot y < 2N - 4\sqrt{2}N^{\frac{3}{4}}, |z| < (p-q)N^{\frac{1}{4}}y, e < (p^2-1)(q^2-1)$. In [46], continued fractions are used for d disclosure with $\delta \leq 0.25$. In [48], applicability of the attack [46] is extended to $d \leq N^{\delta} \cdot 2^r, r \leq 7$. In [17], a method of N factorization is proposed applicable when $|N^{0.5} - p^{0.5} \cdot q^{0.5}|$ is sufficiently small (less than 2^{112} as explained on in [17]).

Timing attacks are type of attacks where an intruder compromise secrete parameters from the execution of the cryptosystem rather than from any ingrained weakness in the mathematical properties of the system [5]. In [1] a timing attack is proposed against RSA private key using genetic algorithm. A Super-encryption (successive encryption of the ciphertexts) is proposed in [41, 4]. However, in [26, 37], it is shown that the probability of success is about 10^{-90} for the parameters proposed for RSA in [38] because *p*-1, *q*-1 shall have large prime factors, and similar for them as well. Bleichenbacher [6] defines that plaintexts m_i are related if $m_i = f_i(m)$ for some known polynomials f_i and shows that having l RSA public keys $(e_1,N_1),\ldots,(e_l,N_l), N = N_1N_2\ldots N_l$ and $c_i = f_i(m)^{e_i} \bmod N_i$ for $i=1,\ldots,l$, the plaintext *m* can be computed in time polynomial in log *N* using Coppersmith's algorithm [14], A method of the broadcasted message disclosure is proposed based on the use of the CRT allowing reducing a number of modular equations to a single equation and then finding e-th order root over integers, in the simplest case of broadcasting one and the same message, [6], or a univariate polynomial root finding using Coppersmith method [14] for broadcasting related messages based on a small message. The paper considers messages, m_i, related to the base message, *m*, by an affine transformation, $m_i = \alpha i \cdot m + \beta_i \bmod n_i$ [6], p. 242, whereas in Coppersmith method only translation transformation is expected to be used: $m' = m+t$ [14], DeLaurentis [16] considered two cases. In Case 1, a probabilistic algorithm is proposed that allows factoring modulus $N = p \cdot q$, using information on the public-private key pair of the attacker (insider) but not of the other users, neither public, nor private keys, within average number of runs at most 2. In Case 2, without factoring of N, an own encryption-decryption key pair as well as an encryption key of another valid user are used to disclose an equivalent for the private key of another user that may be used to disclose his messages and to forge his signature. Simmons [40] considers one message encrypted by two different encryption keys resulting in two ciphertexts of one and the same message. If the encryption keys are co-prime, their mutual inverses may be found and used for the message disclosing. In [30], it is said: "Values such as 3 and 17 can no longer be recommended, but commonly used values such as $2^{16}+1=65537$ still seem to be fine. If one prefers to stay on the safe side one may select an odd 32-bit or 64-bit public exponent at random."

If a known plaintext-ciphertext pair, (P,C) is known, Discrete Logarithm Problem (DLP) solution can be used to disclose the private key as $d = log_{C,N}P$. DLP computational complexity is of the order of that of integer factorization and in parallel with factorization respective DLP solving is reported in [44, 51].

In [14], a method for recovering RSA messages is proposed for rather large encryption exponent such as, $e = 2^{16}+1$. The method assumes that two plain messages are encrypted with the same encryption exponent, e, and modulus, N, and one of the messages, m_2, is related with another one, m_1, by an affine transformation, $m_2 = a \cdot m_1 + b$, and two respective ciphertexts are known, c_1,c_2. The message, m_1, is found as a root of a polynomial which is the Greatest Common Divisor (GCD) of two univariate polynomials modulo N, $p_1(m_1) = m_1^e - c_1, p_2(m_1) = (a \cdot m_1 + b)^e - c_2$. The GCD is obtained using Euclid's algorithm. The method is generalized for the cases of m_2=p(m_1), where $p(\quad)$ is a polynomial, and for multiple messages polynomially related, $p(m_1,\ldots, m_k) = 0$. As far as all the related messages, $m_2,\ldots, m_k$ depend on the single message, m_1, this mode of operation can be considered as "broadcasting" of the message m_1 and its dependent messages, $m_2,\ldots, m_k$ encrypted each with its own encryption exponent. Maximal encryption exponent mentioned in the paper is $e = 2^{16}+1$. We found GCD of two univariate polynomials, $p_1 = x^e - C_1$ mod N and $p_2=(x+1)^e$ - C_2 mod N, where $N= p \cdot q = (2^{20} +7) \cdot (220+13)$, $C_1= m^e$ mod N, $C_2= (m + 1)^e$ mod N, $e = 2^{16}+1$ by Maple 2016 (Intel i7-7700 CPU 3.60 GHz, 8GB RAM), nearly in 6 minutes. Then, message, $m = 2$, is recovered as the root of GCD (Figure 1), Boneh [11] proposed attacking n-bit RSA message, m, using Meet-In-The-Middle (MITM) attack. MITM attack is applied by two steps. A pre-computation step where the message is represented as $m = m_1m_2$ with $m_1 \leq 2^{n_1}$ and $m_2 \leq 2^{n_2}$. Hence, $c/m_2^e = m_1^e \: mod \: N$. A table of size 2^{n_1} has to be built containing the values $m_1^e \: mod \: N$ for all $m_1 \in 0,1,\cdots,2^{n_1} - 1$. Then, in the search step we check for each$m_2 \in 0,1,\cdots,2^{n_2} - 1$, whether $c/m_2^e \: mod \: N$ is present in the table. Any collision reveals the message m. We implemented MITM attack [11] using NTL [39] library (Intel i5-8250U CPU 1.60 GHz, 8GB RAM). Our implementation shows that the time to recover a 40-bit message encrypted with $e = 2^{16}+1$ (see Example 1)

Is 2.25 seconds for pre-computation step and 0.202 second for searching step. Thus, from the analysis conducted we see that known lattice-based attacks against RSA private key section 1.1 and against RSA messages practically use small public encryption exponent, large part of the message to be known in advance, or a message to be broadcast. On the other hand, a non-lattice based attack in [15] has cost of $O(e^2)$ for computing GCD [3], where e is the RSA encryption exponent represents the degree of polynomials, while MITM [13] has cost of $O\left(n\sqrt{2^n}\right)$, where n is the message length in bits. Table 1 shows features of the known RSA attacks. The analysis of the attacks on RSA conducted above shows that they are not applicable for key-size greater than 829 bits, with p,q such that $p - 1$, $q - 1$ have large prime factors, encryption and decryption keys are greater than $N^{0.5}$, and $round(N^{0.5} - \lfloor p^{0.5} \rfloor \cdot \lfloor q^{0.5} \rfloor)$ is large. Herein, we propose a new fast attack using LLL against RSA messages based on the first found herein opportunity of RSA encryption representation as an element of 2-dimensional RSA lattice. Our attack works in the conditions specified above where other attacks can't work, and requires neither knowledge of any part of the message in advance, nor limitations on the size of public exponent e, nor message broadcasting as shown in the last row of Table 1 but imposes constraints on the recoverable messages. Our COA attack computational complexity is $O(n^2)$, see Section 3.4. In our experiments, see example 3, our attack on 2001 RSA 2050-bit messages took 45.775 seconds with about 0.1 success rate. The rest of the paper is organized as follows. In section 2, we introduce RSA algorithm, lattice concepts, and LLL algorithm. In section 3, we introduce 2-dimensional RSA lattice and COA on RSA using LLL is proposed, its complexity is estimated. Additional experiments on application of our attack to RSA cracking with up-to 8193-bit messages are given in section 4. Section 5 concludes the paper.



```
m := 2
                                  2
p := 2^20 + 7
                               1048583
q := 2^20 + 13
                               1048589
 N := p·q
                            1099532599387
e := 2^16 + 1
                                65537
c1 := m^e mod N
                             283976582449

c2 := (m + 1)^e mod N
                             217618377630

p1 := (x^e − c1)mod N
                        x^65537 + 815556016938

p2 := ((x + 1)^e − c2) mod N
                   (x + 1)^65537 + 881914221757

s := Gcd(p1, p2) mod N
                         x + 1099532599385

time(Gcd(p1, p2) mod N)
                               343.953

 −1099532599385 mod N
                                  2
```

Figure 1. Maple code implementation of GCD attack [12], recovering RSA message encrypted with large exponent e=2^{16}+1, as a root of a polynomial which is the GCD of two polynomials $P1$ and $P2$ nearly in 6 minutes.

Table 1. Comparison between lattice basis reduction COA and other known RSA attacks.

Attack	Attack's Requirements		
	Prior knowledge of number of bits	**Small value of exponent *e***	**Broadcast messages**
Coppersmith [13]	Yes	No	No
Boneh *et al*. [10]	Yes	No	No
Takayasu and Kunihiro [43]	Yes	No	No
Coppersmith [14]	No	Yes	No
Hastad [20]	No	No	Yes
Bleichenbacher [6]	No	No	Yes
Hastad [19]	No	No	Yes
Simmons [40]	No	No	Yes
DeLaurentis [16]	No	No	Yes
Boneh [11]	No	No	No
Bunder [12]	No	Yes	No
Lattice Basis Reduction COA	**No**	**No**	**No**

2. RSA Algorithm, Lattice Concepts, and LLL Algorithm

In this section, we

1. Review RSA [38],
2. Introduce lattice concepts including Minkowski Second Theorem [35], which sets an upper bound for the norm of the shortest vector in a 2-dimensional lattice,
3. Introduce LLL [29] to find a shortest vector in a 2-dimensional lattice.

2.1. Review of RSA

A message, $m \in Z_N$, is encrypted using

$$c = m^e mod\ N, \tag{3}$$

Where $N = p \cdot q$, p and q are two different prime numbers, and the encryption exponent, *e*, is chosen according to

$$\gcd(e, (p-1)(q-1)) = 1 \tag{4}$$

The message, *m*, is retrieved by decryption of the ciphertext, *c*, from (3) as follows

$$m = c^d mod\ N, \tag{5}$$

Where the decryption exponent, *d*, is the multiplicative inverse of *e* satisfying

$$e \cdot d\ mod\ (p-1)(q-1) = 1. \tag{6}$$

The public key is (*N*,*e*), and the private key is(*N*,*d*).

- *Example* 1 Example of 40-bit RSA encryption/decryption. Let $p = 2^{20} + 33 = 10485609$ and $q = 2^{20} + 13 = 1048589$ be two prime numbers. Then modulus $N = p \cdot q = 1099559862701$. According to (4), let encryption exponent, $e = 2^{16} + 1 = 65537$. According to (6), decryption exponent, *d*=1082377437569. The public key is (*N*,*e*)=(1099559862701,65537), and the private key is (*N*,*d*)=(1099559862701,1082377437569). Let the message, *m*=986648, then the ciphertext is calculated according to (3):

$$c = m^e mod\ N = 480808351840. \tag{7}$$

Message, *m*, is retrieved by decryption of the ciphertext (7) according to (5) as shown in (8):

$$m = c^d\ mod\ N = 986648. \tag{8}$$

2.2. Lattice Concepts

In the following, $||x||, (x \cdot y), \lceil a \rfloor$, and $\mathbb{Z}$ denote Euclidean norm [18] of the vector *x*, dot product of the vectors, *x* and *y*, rounding of the real number, *a*, and the set of integer numbers, respectively.

Let $E(V_1, V_2) \subset \mathbb{Z}^2$ be a 2-dimensional lattice with basis vectors, V_1 and V_2 shown in (9):

$$E(V_1, V_2) = \{a_1V_1 + a_2V_2 : a_1, a_2 \in \mathbb{Z}\}. \tag{9}$$

The same lattice can be represented by different bases. SVP is one of the most widely studied computational problem on lattices [32] defined as follows [23], p. 395:

- *Definition* 1 SVP is the problem of finding a shortest nonzero vector in a lattice *L*, i.e., $v \in L$ that minimizes the Euclidean norm $||v||$.
- *Remark* 1 There may be more than one solution to the SVP.

For example, the integer lattice $\mathbb{Z}^2$, is the set of all 2-dimensional vectors with integer entries. Integer lattice $\mathbb{Z}^2$ can be represented by basis vectors $V_1 = (1,1)$ and $V_2 = (1, 2)$, while the four nonzero vectors $(0,\pm1)$, $(\pm1,0)$ are the solutions to the SVP.

Minkowski's Second Theorem [35], sets an upper bound for the norm, *l*, of the shortest nonzero vector in a 2-dimensional lattice given by (10):

$$\lambda \le \sqrt{\gamma_2}\, \det(L)^{\frac{1}{2}}, \tag{10}$$

Where $\gamma_2 = \frac{2}{\sqrt{3}} \approx 1.154$ is Hermit's constant [35], p. 41, and *det*(*L*) is the determinant of the lattice matrix formed by its basis vectors. Hence,

$$\gamma \le \sqrt{1.154 \det(L)} \approx 1.07 \sqrt{\det(L)}. \tag{11}$$

2.3. LLL lattice Basis Reduction Algorithm

LLL [29] is a lattice reduction algorithm, on termination returns the shortest vectors in the lattice, beginning with the shortest vector v_1, and then with vectors whose lengths increase as slowly as possible until we reach the last vector in the basis in $E(V_1,V_2)$. In next section we propose COA on RSA using LLL.

3. COA on RSA using LLL

We introduce COA on RSA using LLL algorithm. More specifically we

1. Show that RSA encryption forms a 2-dimensional RSA lattice,
2. Show that the plaintext message can be revealed as a component of the shortest vector in the RSA lattice.
3. Propose using LLL for COA on RSA by solving SVP in the RSA lattice.
4. Evaluate complexity of the proposed COA on RSA, and conduct experiments for up to 8193-bit messages.

3.1. 2-Dimensional RSA Lattice

RSA message recovery problem can be formulated as SVP in a 2-dimensional lattice, *E*(V_1,V_2). From (3), we can see that:

$$c = m^j \cdot m^{e-j} mod\ N, j = 1..e-1, \tag{12}$$

And, hence,

$$m^j = \left(m^{e-j}\right)^{-1} \cdot c\ mod\ N. \tag{13}$$

From (13), we see that for any pair of integers, *A* and *B*, satisfying:

$$B\ =\ A \cdot c\ mod\ N \tag{14}$$

(*A*,*B*) is likely to be $(\left(m^{e-j}\right)^{-1}, m^j)$, or (m^{-j}, m^{e-j}). Hence, Equation (14) can be written as

$$A \cdot c + N\ \cdot r\ =\ B, \tag{15}$$

Where *r* is an integer. It forms a 2-dimensional RSA lattice,

$$A \cdot V1 + r \cdot V2 = (A, B), \tag{16}$$

Where V_1=(1,*c*) and V_2=(0,*N*) are basis vectors, at least one of them having Euclidean norm of order *O*(*N*), and determinant of the lattice equal to *N*.

3.2. RSA Message as the Shortest Vector in the RSA Lattice

According to Minkowski's Second theorem (11), vector (*A*,*B*) (16) likely is the shortest vector in the RSA lattice, if

$$||A, B|| < 1.07\sqrt{N}. \tag{17}$$

Hence, our task is to find a pair of comparatively small, (*A*,*B*), satisfying (16) where V_1=(1,*c*) and V_2=(0,*N*) are known vectors. Then, (*A*,*B*), is likely to be $\left(\left(m^{e-j}\right)^{-1}, m^j\right)$, or $\left(m^{-j}, m^{e-j}\right)$. In our attack we adopt LLL to find the shortest vector in the 2-dimensional RSA lattice (16).

3.3. LLL Attack on RSA Message as a Shortest Vector in the RSA Lattice

We want to find a shortest vector w from *E*(V_1,V_2) using LLL that might disclose

$$(A, B) = \left(\left(m^{e-j}\right)^{-1}, m^j\right) \tag{18}$$

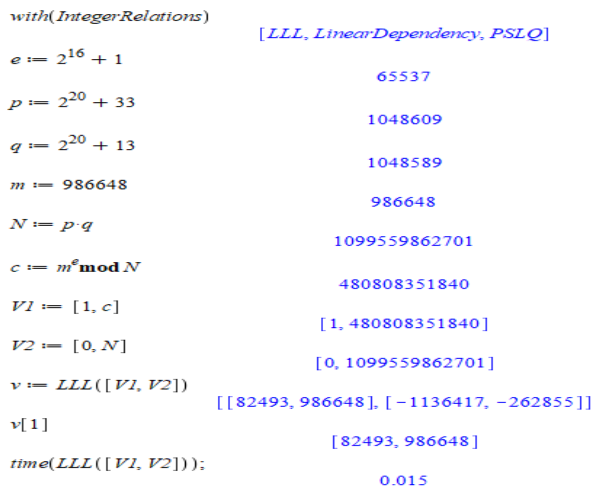
if $\left\|\left(\left(m^{e-j}\right)^{-1}, m^j\right)\right\|$ from (18) is of the order of $O(\sqrt{N})$ meeting (17). In our experiments we used LLL algorithm implemented in Maple 2016.2.

- *Example* 2 shows LLL attack on Example 1 message.
- *Example* 2 LLL attack on 40-bit RSA message from Example 1.

Ciphertext from Example 1, *c* = 480808351840, and modulus *N* = 1099559862701.

Hence, V_1=(1, 480808351840), and V_2=(0, 1099559862701). LLL attack with V_1=(1, 480808351840), V_2=(0, 1099559862701), defined in (16) terminates in 15 milliseconds using Maple, obtaining the shortest vector (see Figure 2) given in (19):

$$v_1 = (82493{,}986648). \tag{19}$$



```
with(IntegerRelations)
                        [LLL, LinearDependency, PSLQ]
e := 2^16 + 1
                                65537
p := 2^20 + 33
                               1048609
q := 2^20 + 13
                               1048589
m := 986648
                                986648
N := p·q
                            1099559862701
c := m^e mod N
                             480808351840
V1 := [1, c]
                          [1, 480808351840]
V2 := [0, N]
                          [0, 1099559862701]
v := LLL([V1, V2])
                 [[82493, 986648], [-1136417, -262855]]
v[1]
                           [82493, 986648]
time(LLL([V1, V2]));
                                0.015
```

Figure 2. LLL attack on RSA message in example 1 using maple 2016.2.

We also run the experiment in C using NTL [39] and found that LLL attack terminates in 4 × 10^{-5} seconds. Thus, we see that our attack, both in Maple and C, takes less time than attacks mentioned in Section 1.3. LLL attack succeeds to retrieve message since it is a component of a shortest vector in the lattice,

$$||(m^{e-1})^{-1}, m|| \approx 990090.6 < \ 1.07\sqrt{N}\ \approx\ 1124497.2.$$

3.4. Complexity of LLL Lattice Basis Reduction Algorithm

Lenstra *et al.* [29] state that for n-dimensional lattices with integer input basis vectors of bounded length *N*, the LLL algorithm terminates after at most *O*(n^2log*N*) iterations.

4. Experiments on RSA Cracking for Up to 8193-Bit Messages

We have conducted experiments using Maple 2016.2 in Windows 8.1 on Lenovo laptop with Intel i5-6200U CPU 2.30 GHz, 8 GB RAM, for RSA with p, q values specified in Table 2 with sizes of

$$N = p \cdot q \tag{20}$$

from 40 to 8193 bits more than twice exceeding recommended RSA key size, 4047 bits, for 2050 year according to the requirements of [30], Table 1. Values of p, q are defined as integer expressions (see Table 2). Note that the prime values (*p*,*q*) used in Rows 1, 2 of Table 2 are strong according to [38], since $p-1$, $q-1$ have large primes as their factors, that is confirmed by the following Maple code:

```
p1 := 3*2^250 + 17; isprime(p1);
5427754182999196660479889922282245680622030531201901439З
    4957425037092795188 9
                                        true
q1 := (2^129 - 1)^2 - 2; isprime(q1);
46З1683569492647816942839400347516314117188091948785023О
    3976837601925445713919
                                        true
ifactor(q1-1)
(2) (11) (13)
    (
    1619469779542883852077915874247383326614401430751323434629289б4
    1964774285713)
ifactor(p1-1);
(2)^4 (103) (2639809)
    (
    124764335585903797489729800929088384303001190653264216177036223
    5159)
```

It can be checked that (*p*,*q*) values in rows 1, 2, and 6 of Table 2 have large $round(N^{0.5} - \lfloor p^{0.5} \rfloor \cdot \lfloor q^{0.5} \rfloor)$ values precluding attack [17].

Table 2. Pairs (*p*,*q*), bit size of *N* used in our experiments, (*a*,*b*) pairs from (27)-(29) for which RSA was cracked, number of cracked messages, and respective δ_{min} and δ_{max} from (30), for Digits=10 and *C*=0 in Maple.

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Pair(***p***,***q***)#	***p***	***q***	Bit size of ***N***	(***a***,***b***)from (24) for which RSA was cracked, ***k***= 1	Number of RSA cracks	δ_{min}	δ_{max}
0	$2^{20}+33$	$2^{20}+13$	40	$(8,1),(4,\pm1),(2,\pm1)$	153	0.025	0.508625
1	$2^{130}-5$	$2^{131}+39$	261	$(14,-1),(4,2),(2,\pm1)$	58	0.01	0.5010325
2	$3\times2^{250}+17$	$(2^{129}-1)^2-2$	509	$(2,\pm1),(4,-1),(22,\pm1)$	59	0.01	0.5007125
3	$3\times2^{512}+349$	$3\times2^{512}-511$	1026	$(4,1),(8,-1),(2,\pm1)$	85	0.01	0.5007065
4	$3\times2^{1024}+545$	$3\times2^{1024}-1717$	2050	$(20,6),(4,2),(2,\pm1)$	64	0.01	0.5005
5	$3\times2^{2048}+595$	$3\times2^{2048}-1105$	4098	$(26,5),(4,2),(2,\pm1)$	68	0.001	0.5007
6	$3\times2^{4096}+1075$	$2^{4096}-2549$	8193	$(28,4),(4,2),(2,\pm1),(14,-1)$	66	0.00375	0.50003

In our experiments, messages are defined via a parameter,

$$\delta \in (0,1) \tag{21}$$

As follows,

$$m = int(N^{\delta}) + ii, ii \in -C, \cdots, C, C \geq 0, \tag{22}$$

Where $C \geq 0$ is an integer and *int*( ) returns integer part of its input. Calculations on the float-point numbers are done with accuracy of 10, 15, 100, 200, 600, 800, and 1600 digits:

```
> Digits := 1600;#600;#200;#10;#15
                                   Digits := 1600
```

We try vectors

$$v(j) = \left(\left(m^{e-j}\right)^{-1}, m^j\right) \tag{23}$$

Meeting the following two-dimensional lattice equation

$$v(j)_1 \cdot V_1 + r \cdot V_2 = v(j) \tag{24}$$

With

$$V_1 = \binom{1}{c}, V_2 = \binom{0}{N} \tag{25}$$

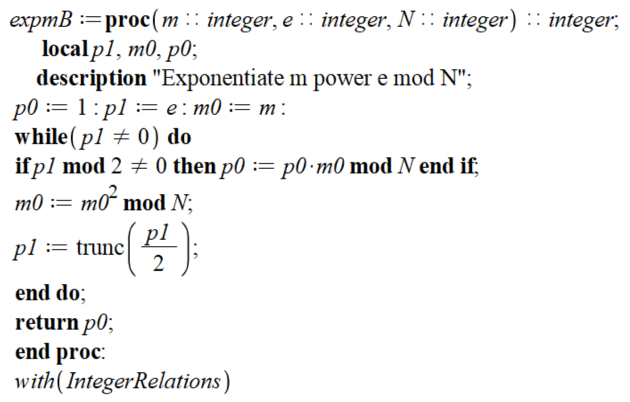
For *j*=1,…,100, according to (16), by the following code:

- *Code* 1. Maple code for RSA cracking using LLL with *j*∈ {1,⋯ ,100}. Initial conditions for the code are defined in Code 3 and example 3. It trie cracking 2001 RSA messages in the range *m*0-1000…*m*0+1000, where *m*0 is defined in its first line as trunc (N^{δ}).

```
st := time( ) : bnd := 1.07·N^0.5 : lb := [ ] : gb := [ ] : m0 := trunc(N^δ) :
 for ii from -1000 to 1000 do#−992 do#−1000 to 1e3 do
 m := m0 + ii;
p0 := expmN(m, e − 1, N); igcdex(N, p0,'z','me1');
 c := p0·m mod N;
nrm := (me1^2 + m^2)^0.5 ;
 V := [[1, c], [0, N]] ; VR := LLL(V,'integer');
 for j to 1e2 do
  if abs(VR[1, 1]) = m^j or abs(VR[1, 2]) = m^j or abs(VR[2, 1]) = m^j or
abs(VR[2, 2]) = m^j then
    if nrm < bnd then lb := [op(lb), [ii, j]] else gb := [op(gb), [ii, j]]
end if
  end if
 end do:
 end do: ft := time( ) : tot := ft − st;
```

In the Code 1, with *C* = 1000, we check the both returned by LLL vectors and each their component on equality to m^j. Exponentiation function and LLL used in Code 1 are introduced in Code 2 as follows:

- *Code* 2. Maple code introducing exponentiation function and LLL.



```
expmB := proc(m :: integer, e :: integer, N :: integer) :: integer;
    local p1, m0, p0;
   description "Exponentiate m power e mod N";
p0 := 1 : p1 := e : m0 := m :
while(p1 ≠ 0) do
if p1 mod 2 ≠ 0 then p0 := p0·m0 mod N end if;
m0 := m0^2 mod N;
p1 := trunc(p1/2);
end do;
return p0;
end proc:
with(IntegerRelations)
```

RSA was successfully cracked under conditions (27)-(29) on the encryption key, *e*, defined via Euler totient function,

$$\phi(N) = (p-1)\cdot(q-1), \tag{26}$$

In a general form

$$e = k\cdot\frac{\phi(N)}{a} - b, \tag{27}$$

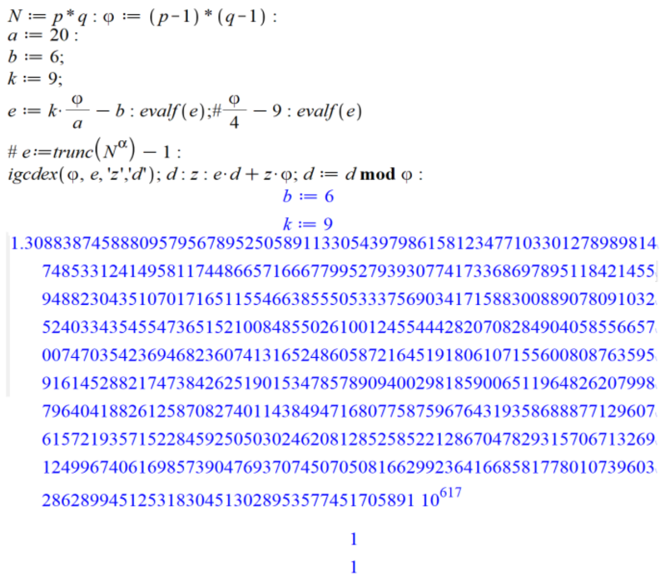
Such that

$$\gcd\big(e, \phi(N)\big) = 1, \tag{28}$$

$$\phi(N)\ mod\ a = 0. \tag{29}$$

It is implemented in Maple by the following Code 3, for Digits=1600:

- *Code* 3. Maple implementation of RSA encryption key, *e*, calculation according to (27)-(29), for *N* of 2050 bit size from Table 2.



```
N := p*q : φ := (p-1)*(q-1) :
a := 20 :
b := 6;
k := 9;
e := k·φ/a − b : evalf(e);#φ/4 − 9 : evalf(e)
# e:=trunc(N^α) − 1 :
igcdex(φ, e,'z','d'); d : z : e·d + z·φ; d := d mod φ :
                                  b := 6
                                  k := 9
1.308838745888095795678952505891133054397986158123477103301278989814
    7485331241495811744866571666779952793930774173368697895118421455
    9488230435107017165115546638555053337569034171588300889078091032
    5240334354554736515210084855026100124554442820708284904058556657
    0074703542369468236074131652486058721645191806107155600808763595
    9161452882174738426251901534785789094002981859006511964826207998
    7964041882612587082740114384947168077587596764319358688877129607
    6157219357152284592505030246208128525852212867047829315706713269
    1249967406169857390476937074507050816629923641668581778010739603
    286289945125318304513028953577451705891 10^617

                                     1
                                     1
```

For the example of data shown in Code 2, $e \approx \frac{N}{2}, d \approx \frac{N}{10}$, thus attacks described in section 1 are not applicable. We try finding a range of the parameter,

$$\delta \in [\delta_{min}, \delta_{max}], \tag{30}$$

Or a set of values, $\{\ \delta_{min}, \delta_{max}\}$, for which our method successfully cracks RSA (see Table 2, columns 7, 8). In Table 2, columns 5, 6, pairs (a, b), for which RSA was successfully cracked, and number of successful cracks are given (for $C = 0$ in (22)). We found that for all successful cracks,

$$j = |b|, \tag{31}$$

holds, where *j*,*b* are from (23), (24), and (27), respectively, i.e., the power of the plaintext message, *m*, revealed by our attack on RSA, always is equal to *jbj* from (27). Thus, in the experiments, we find two conditions, (29) and (31), holding that need explanation. Also, the results of all our experiments show that condition (32) holds

$$\delta_{max}\cdot|b| \approx 0.501. \tag{32}$$

To verify (32), we have conducted special massive investigation of its validity for (*p*,*q*) pair from Table 2, row 4, results of which are given in Table 3, and confirm its validity. Hence, we need explaining (29), (31), and (32).

Table 3. Results of experiments in Maple 2016 on RSA cracking for Digits=600, p := $3{\cdot}2^{1024}$+515, q:=$3{\cdot}2^{1024}$+1717, (27)-(29) hold, *k*=1, C = 1000, δ_{max} is from (30).

a	b	Number of cracks	δ_{max}	$\delta_{max}\cdot\lvert b\rvert$
20	6	9205	0.0835	0.501
20	10	4987	0.050107	0.50107
20	-4	2082	0.12521	0.50084
20	-6	1642	0.038348	0.50088
20	-8	1913	0.062615	0.50092
20	-14	1336	0.035769	0.500766
5	-1	5626	0.501	0.501
5	-25	2896	0.020045	0.501125
4	2	21599	0.25066	0.50132
4	6	22937	0.083528	0.501168
10	13	6469	0.0385503	0.501154
	Total cracks:	80692	Average $\delta_{max}\cdot\lvert b\rvert$:	0.501022

Explanation of (29). Consider (20), (21), (22) for *C* = 0, (26), and (27). Then, RSA ciphertext, *c*, is defined as follows:

$$c = m^e mod\ N = m^{k\cdot\frac{\phi(N)}{a}-b} mod\ N. \tag{33}$$

Experiments show that with high probability, ranging from 0.1 to 0.5, (34) holds:

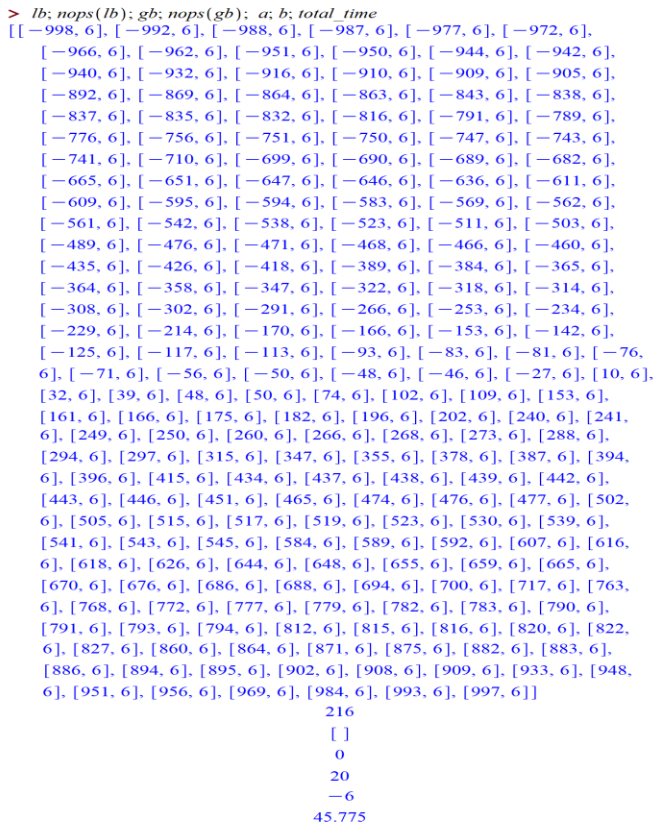
$$m^{\frac{k\phi(N)}{a}} mod\ N = 1. \tag{34}$$

Note that due to Euler's theorem [2],

$$m^{k\phi(N)} mod\ N = 1, \tag{35}$$

And the left-hand side (LHS) of (34) is *a*-th root of unity from LHS of (35), which is highly likely to be also unity. The probability of our COA on RSA success estimate is illustrated by example 3.

- *Example* 3 Conducting calculations by Code 1 in Maple 2016.2, with Digits=1600, q=$3{\cdot}2^{1024}$-1717, p = $3\cdot 2^{1024} + 515, \delta$ = 0:071435 considering 2001 numbers, m = $\lfloor N^{\delta} \rfloor$ + ii, ii ∈ [-C,⋯ , C],C = 1000, we find 216 cases when (34) holds, in particular, for ii = −998,−992,−988, etc., Respective Maple output is shown below:



```
> lb; nops(lb); gb; nops(gb);  a; b; total_time
[[−998, 6], [−992, 6], [−988, 6], [−987, 6], [−977, 6], [−972, 6],
    [−966, 6], [−962, 6], [−951, 6], [−950, 6], [−944, 6], [−942, 6],
    [−940, 6], [−932, 6], [−916, 6], [−910, 6], [−909, 6], [−905, 6],
    [−892, 6], [−869, 6], [−864, 6], [−863, 6], [−843, 6], [−838, 6],
    [−837, 6], [−835, 6], [−832, 6], [−816, 6], [−791, 6], [−789, 6],
    [−776, 6], [−756, 6], [−751, 6], [−750, 6], [−747, 6], [−743, 6],
    [−741, 6], [−710, 6], [−699, 6], [−690, 6], [−689, 6], [−682, 6],
    [−665, 6], [−651, 6], [−647, 6], [−646, 6], [−636, 6], [−611, 6],
    [−609, 6], [−595, 6], [−594, 6], [−583, 6], [−569, 6], [−562, 6],
    [−561, 6], [−542, 6], [−538, 6], [−523, 6], [−511, 6], [−503, 6],
    [−489, 6], [−476, 6], [−471, 6], [−468, 6], [−466, 6], [−460, 6],
    [−435, 6], [−426, 6], [−418, 6], [−389, 6], [−384, 6], [−365, 6],
    [−364, 6], [−358, 6], [−347, 6], [−322, 6], [−318, 6], [−314, 6],
    [−308, 6], [−302, 6], [−291, 6], [−266, 6], [−253, 6], [−234, 6],
    [−229, 6], [−214, 6], [−170, 6], [−166, 6], [−153, 6], [−142, 6],
    [−125, 6], [−117, 6], [−113, 6], [−93, 6], [−83, 6], [−81, 6], [−76,
    6], [−71, 6], [−56, 6], [−50, 6], [−48, 6], [−46, 6], [−27, 6], [10, 6],
    [32, 6], [39, 6], [48, 6], [50, 6], [74, 6], [102, 6], [109, 6], [153, 6],
    [161, 6], [166, 6], [175, 6], [182, 6], [196, 6], [202, 6], [240, 6], [241,
    6], [249, 6], [250, 6], [260, 6], [266, 6], [268, 6], [273, 6], [288, 6],
    [294, 6], [297, 6], [315, 6], [347, 6], [355, 6], [378, 6], [387, 6], [394,
    6], [396, 6], [415, 6], [434, 6], [437, 6], [438, 6], [439, 6], [442, 6],
    [443, 6], [446, 6], [451, 6], [465, 6], [474, 6], [476, 6], [477, 6], [502,
    6], [505, 6], [515, 6], [517, 6], [519, 6], [523, 6], [530, 6], [539, 6],
    [541, 6], [543, 6], [545, 6], [584, 6], [589, 6], [592, 6], [607, 6], [616,
    6], [618, 6], [626, 6], [644, 6], [648, 6], [655, 6], [659, 6], [665, 6],
    [670, 6], [676, 6], [686, 6], [688, 6], [694, 6], [700, 6], [717, 6], [763,
    6], [768, 6], [772, 6], [777, 6], [779, 6], [782, 6], [783, 6], [790, 6],
    [791, 6], [793, 6], [794, 6], [812, 6], [815, 6], [816, 6], [820, 6], [822,
    6], [827, 6], [860, 6], [864, 6], [871, 6], [875, 6], [882, 6], [883, 6],
    [886, 6], [894, 6], [895, 6], [902, 6], [908, 6], [909, 6], [933, 6], [948,
    6], [951, 6], [956, 6], [969, 6], [984, 6], [993, 6], [997, 6]]
                                  216
                                  [ ]
                                   0
                                  20
                                  −6
                                45.775
```

Thus, probability of (34) holding, and thus our attack takes 45.775 seconds, its success probability under conditions (27)-(29), may be estimated as 216/2001=0.1079, and (29) is explained. Now, we explain (31) and (32).

Explanation of (31) and (32). Our method of cracking of RSA ciphertext is as follows (recall (12)-(16), (23), (24)). Rewrite (33):

$$c = m^{e-j} \cdot m^j \ mod\ N, 0 < j < e. \tag{36}$$

From (36), we get

$$c\left(m^{e-j}\right)^{-1} = m^j \ mod\ N. \tag{37}$$

Reminding (23), from (37), we arrive at (24). Applying LLL algorithm to the lattice defined by (25), we obtain a shortest vector, $\begin{pmatrix} S_1 \\ S_2 \end{pmatrix}$, of the lattice such that $\begin{pmatrix} S_1 \\ S_2 \end{pmatrix} = v(j)$, if the norm of $v(j)$ meets Minkowski's Second theorem

$$\left\|\begin{pmatrix} S_1 \\ S_2 \end{pmatrix}\right\| \le \|v(j)\| = \sqrt{v(j)_1^2 + v(j)_2^2} \le \sqrt{\gamma_2 \cdot N} = \sqrt{\frac{2}{3}} \cdot N, \tag{38}$$

Where $\gamma_2 \approx 1.1547$ is Hermite's constant for the 2-dimensional lattice. To meet (38), from (23), we have

$$\sqrt{(m^{-e+j})^2 + m^{j^2}} \le \sqrt{\frac{2}{\sqrt{3}} \cdot N} \tag{39}$$

From, (22) with *C* = 0, (27), (34), (39), we have

$$\sqrt{(m^{b+j})^2 + m^{j^2}} = \tag{40}$$

$$\sqrt{\lfloor N^{\delta} \rceil^{b+j} + \lfloor N^{\delta} \rceil^{j^2}} \le \sqrt{2 \cdot \frac{N}{\sqrt{3}}} \approx N^{0.50005}$$

From (40), we have two cases

- Case 1: $b \ge 0$. Let $j = 0$ in (40). Then, $v(j) = \begin{pmatrix} m^b \\ 1 \end{pmatrix}$, and we have

$$\sqrt{\lfloor N^{\delta} \rceil^{b^2} + 1} \approx N^{b \cdot \delta} \le N^{0.5005}, \tag{41}$$

And thus,

$$b \cdot \delta \le 0.50005. \tag{42}$$

- Case 2: $b < 0$. Let $j = -b = |b|$. Then, $v(j) = \begin{pmatrix} 1 \\ m^b \end{pmatrix}$, and we have

$$\sqrt{\lfloor N^{\delta} \rceil^{|b|^2} + 1} \approx N^{|b| \cdot \delta} \le N^{0.5005}, \tag{43}$$

And then,

$$|b| \cdot \delta \le 0.50005. \tag{44}$$

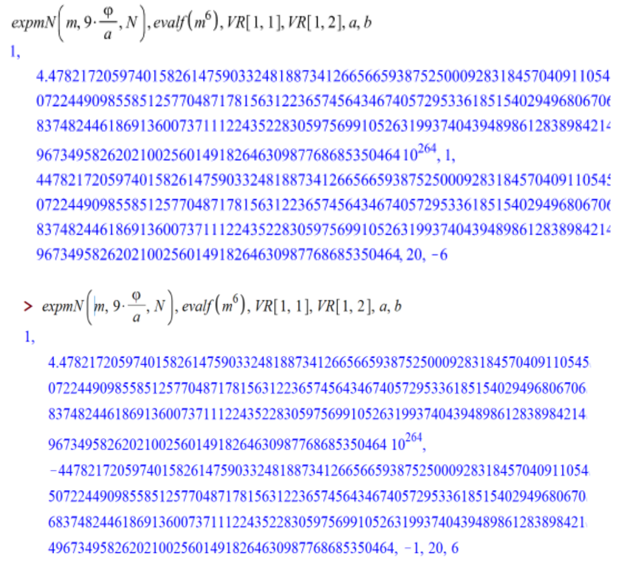
Thus, from (41), (43), we may have RSA cracks in the from

$$|v(j)| = \begin{pmatrix} m^b \\ 1 \end{pmatrix} \text{or } |v(j)| = \begin{pmatrix} 1 \\ m^b \end{pmatrix}, \tag{45}$$

that have been observed in all our experimental results shown in Tables 2, and 3.

Example 4 confirms that (45) holds in a particular experiment as in all other ones.

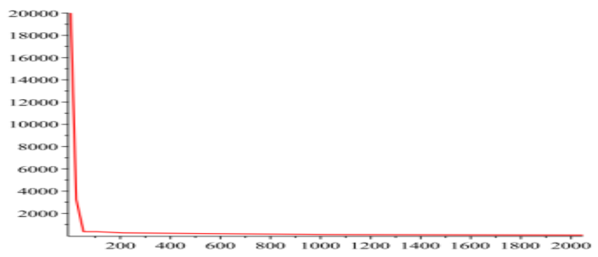
- *Example* 4 Maple output for RSA cracking with k = 9, a = 20, b = ±6, d = 0:071435, showing that (34) holds, and values found by LLL in *VR* [1,1..2], see Code 1, meet (45).



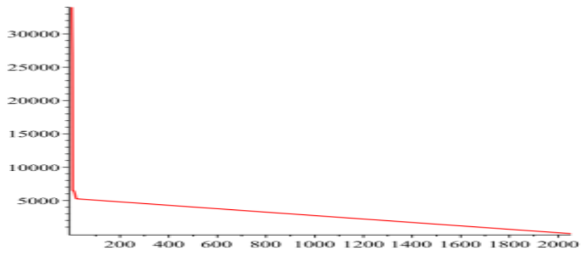
```
expmN(m, 9·φ/a, N), evalf(m^6), VR[1, 1], VR[1, 2], a, b
1,
   4.4782172059740158261475903324818873412665665938752500092831845704091105
   072244909855851257704871781563122365745643467405729533618515402949680670
   8374824461869136007371112243522830597569910526319937404394898612838984214
   96734958262021002560149182646309877686853504641 0^264, 1,
   4478217205974015826147590332481887341266566593875250009283184570409110545
   072244909855851257704871781563122365745643467405729533618515402949680670
   8374824461869136007371112243522830597569910526319937404394898612838984214
   96734958262021002560149182646309877686853504 64, 20, −6

 > expmN(m, 9·φ/a, N), evalf(m^6), VR[1, 1], VR[1, 2], a, b
1,
    4.4782172059740158261475903324818873412665665938752500092831845704091105 45
    072244909855851257704871781563122365745643467405729533618515402949680670 6
    8374824461869136007371112243522830597569910526319937404394898612838984214
    967349582620210025601491826463098776868535 0464 10^264,
    −4478217205974015826147590332481887341266566593875250009283184570409110 54
    5072244909855851257704871781563122365745643467405729533618515402949680670
    6837482446186913600737111224352283059756991052631993740439489861283898421
    4967349582620210025601491826463098776868535 0464, −1, 20, 6
```

Also, range for δ defined by (42), (44) is confirmed by our experiments. From Table 3, last row, we see that (44) holds on average with accuracy 0.00097=0.50102-0.50005. Table 3 contains number of RSA successful cracks for different values of *a*, *b*, maximal δ_{max} from (30) and LHS of (44). Thus, (45) explains (31), and

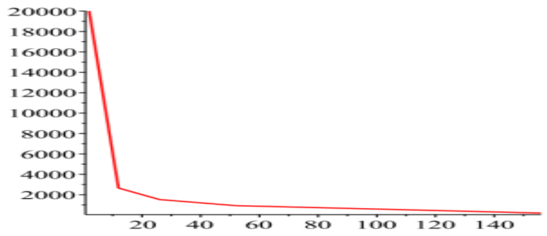
(44) explains (32). To find the relation between a and number of RSA successfully cracked messages, we run Code 1 with p, q from rows 3-6 of Table 2, $\delta \in 0.01, \ldots, 0.52$ yields to launch 104,052 attacks on each (p, q, a) value. Figure 3 shows an inverse proportion between value of a and number of successful cracks. Thus, decreasing of the public key leads to decreasing of the success rate of our attack



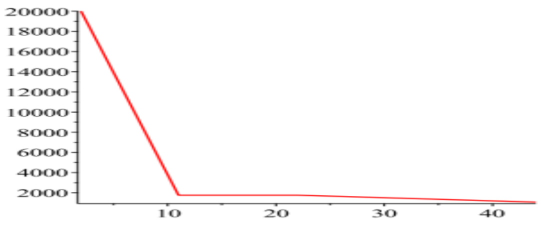
a) Shows 20010 message cracks at a =2 and drops to 51 message cracks at a = 2048 out of 104,052 message attacks.



b) Shows 34017 message cracks at a = 2 and drops to 18 message cracks at a=2053 out of 104,052 message attacks.



c) Shows 20010 message cracks at a = 2 and drops to 0 message cracks at a = 33739 out of 104,052 message attacks.



d) Shows 20010 message cracks at a = 2 and drops to 199 message cracks at a = 222 out of 104,052 message attacks.

Figure 3. Inverse relation between value of parameter a in (27) and number of successful RSA message cracks out of 104,052 message attacks. (a)-(d) show results for (p, q) from rows 3-6 in from Table 2 respectively. Horizontal and vertical axes represent a and the number of successfully cracked RSA messages, respectively.

5. Conclusions

In this paper, we show that RSA-encrypted message considered as a component of a shortest vector of the RSA lattice can be revealed by LLL attack. LLL attack runs in time quadratic in the bit number of modulus N (see section 3.4). LLL attack targets messages meeting (13)-(17) being a shortest vector in the RSA lattice. Our attack works in the conditions discussed in Section 1 in which known attacks can't work, and it does not impose any other requirements, such as the need for very small public exponent, e, part of the plaintext to be known in advance, or a message broadcasting to sufficiently many participants, each holding a different modulus with a known affine transformation, or using common modulus as other attacks do [10, 19, 20, 21, 22, 28]. Our attack shows significant speed (15 milliseconds using Mupad, and 4×10^{-5} seconds using NTL [39] library for Example 2) in recovering a 40- bit message in comparison to our implementation for Boneh MITM attack [11] where 2.202 seconds are needed to recover the same length message (2 seconds for pre-computation step, and 0.202 seconds

For searching step using NTL [39] library). Additionally, we have conducted experiments with the proposed method for N with bit sizes up to 8193 in Maple 2016.2, with results presented in Tables 2-3, in which thousands of successful RSA cracks were conducted using Code 1 run-time of which in the conditions of example 3 for 2001 RSA 2050-bit messages cracking is about 45 seconds. The cracks were made for large public key values meeting (27)-(29) for which truth of (29), (31), (32) was discovered. Based on these findings, for RSA not to be susceptible to the attack proposed herein, it is recommended RSA public keys to be selected such that (27)-(29) are not satisfied.

References

[1] Ali H. and Al-Salami M., "Timing Attack Prospect for RSA Cryptanalysis Using Genetic Algorithm Technique," *The International Arab Journal of Information Technology*, vol. 1, no. 1, pp. 80-85, 2004.

[2] Alimorad R. and Arkian H., "Integer Factorization Implementations," *ICTACT Journal on Communication Technolgy*, vol. 7, no. 2, pp. 1310-1314, 2016.

[3] Belhaj S. and Kahla H., "on the Complexity of Computing the GCD of two Polynomials via Hankel Matrices," *ACM Communications in Computer Algebra*, vol. 46, no. 3/4, pp. 74-75, 2013.

[4] Berkovits S., "Factoring Via Superencryption," *Cryptologia*, vol. 6, no. 3, pp. 229-237, 182.

[5] Biswas S. and Tiwari N., "Attacks and Threats on RSA," *in Proceedings of Emerging Technologies in Data Mining and Information Security*, pp. 737-747, 2018.

[6] Bleichenbacher D., "On the Security of the KMOV Public Key Cryptosystem," *in Proceedingsof Annual International Cryptology*

Conference Lecture Notes in Computer Science, Santa Barbara, pp. 235-248, 1997.

[7] Boneh D., "Twenty Years of Attacks on the RSA Cryptosystem," *Notices of the AMS*, vol. 46, no. 2, pp. 203-213, 1999.

[8] Boneh D. and Durfee G., "Cryptanalysis of RSA with Private Key d Less than N^0:292," *in Proceedings of Eurocrypt'99 Proceedings of the 17th International Conference on Theory and Application of Cryptographic Techniques*, Berlin, pp. 1-11, 1999.

[9] Boneh D. and Durfee G., "Cryptanalysis of RSA with Private Key d Less Than N0.292," *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1339-1349, 2006.

[10] Boneh D., Durfee G., and Frankel Y., "An Attack on RSA Given A Small Fraction of The Private Key Bits," *in Proceedings of International Conference on the Theory and Application of Cryptology and Information Security*, Beijing, pp. 25-34, 1998.

[11] Boneh D., Joux A., and Nguyen P., "Why Textbook Elgamal and RSA Encryption Are Insecure," *in Proceedings of International Conference on the Theory and Application of Cryptology and Information Security*, Kyoto, pp. 30-43, 2000.

[12] Bunder M., Nitaj A., Susilo W., and Tonien J., "A Generalized Attack on RSA Type Cryptosystems," *Theoretical Computer Science*, vol. 704, pp. 74-81, 2017.

[13] Coppersmith D., "Finding A Small Root of A Bivariate Integer Equation, Factoring with High Bits Known," *in Proceedings of International Conference on the Theory and Applications of Cryptographic Techniques, Lecture Notes in Computer Science*, Saragossa, pp. 178-189, 1996.

[14] Coppersmith D., "Finding a Small Root of a Univariate Modular Equation," *in Proceedings of International Conference on the Theory and Applications of Cryptographic Techniques*, Saint-Malo, pp. 155-165, 1996.

[15] Coppersmith D., Franklin M., Patarin J., and Reiter M., "Low Exponent RSA With Related Messages," *in Proceedings of International Conference on the Theory and Applications of Cryptographic Techniques, Lecture Notes in Computer Science*, Saragossa, pp. 1-91, 996.

[16] DeLaurentis J., "A Further Weakness in the Common Modulus Protocol for the RSA Cryptoalgorithm," *Cryptologia*, vol. 8, no. 3, p. 253-259, 1984.

[17] Ghafar A., Ariffin M., and Asbullah M., "A New LSB Attack on Special-Structured RSA Primes," *Symmetry*, vol. 12, no. 5, pp. 1-13, 2020.

[18] Gradshteyn I., Ryzhik I., Jeffrey A., and Zwillinger D., *Table of Integrals, Series, and Products, Sixth Edition 6th Edition*, Academic Press, 2000.

[19] Hastad J., "On using RSA with Low Exponent in A Public Key Network," *in Proceedings of Conference on the Theory and Application of Cryptographic Techniques*, *Lecture Notes in Computer Science*, Santa Barbara, pp. 403-408, 1985.

[20] Hastad J., "Solving Simultaneous Modular Equations of Low Degree," *Society for Industrial and Applied Mathematics*, vol. 17, no. 2, pp. 336-341, 1988.

[21] Hoffstein J., Howgrave-Graham N., Pipher J., and Whyte W., *The LLL Algorithm*, Springer Link, 2009.

[22] Hoffstein J., Pipher J., and Silverman J., "NTRU: A Ring-Based Public Key Cryptosystem," *in Proceedings of International Algorithmic Number Theory Symposium*, Portland, pp. 267-288, 1988.

[23] Hoffstein J., Pipher J., and Silverman J., *An Introduction to Mathematical Cryptography*, Springer Link, 2014.

[24] Hoffstein J., Silverman J., and Whyte W., "Estimated Breaking Times for NTRU Lattices Updated 2003," Technical Report, 1999.

[25] "IEEE Standard Specification for Public Key Cryptographic Techniques Based on Hard Problems Over Lattices," 2009.

[26] Jamnig P., "Securing The RSA-Cryptosystem Against Cycling Attacks," *Cryptologia*, vol. 12, no. 3, pp. 159-164, 1988.

[27] Kirchner P. and Fouque P., "Revisiting Lattice Attacks on Overstretched NTRU Parameters," *in Proceedings of Annual International Conference on the Theory and Applications of Cryptographic Techniques, Part I, Lecture Notes in Computer Science*, Paris, pp. 3-26, 2017.

[28] Kleinjung T., Aoki K., Franke J., Lenstra A., Thomé E., Bos J., Gaudry P., Kruppa A., Montgomery P., Osvik D., Riele H., Timofeev A., and Zimmermann P., "Factorization of A 768-Bit RSA Modulus," *in Proceedings of Annual Cryptology Conference*, Santa Barbara, pp. 333-350, 2010.

[29] Lenstra A., Lenstra H., and Lovasz L., "Factoring Polynomials with Rational Coefficients," *Mathematische Annalen*, vol. 261, no. 4, pp. 515-534, 1982.

[30] Lenstra A. and Verheul E., "Selecting Cryptographic Key Sizes," *Journal of Cryptology*, vol. 14, pp. 255-293, 2001.

[31] May A., *in The LLL Algorithm-Survey and Applications*, Springer Link, 2009.

[32] Micciancio D., "Shortest Vector Problem," *in Encyclopedia of Algorithms*., pp. 1974-1977, 2016.

[33] Moriarty J., Kaliski B., Jonsson J., and Rusch A., "PKCS# 1: RSA Cryptography Specifications Version 2.2.," Technical Report 2016.

[34] Mumtaz M. and Ping L., "Forty Years of Attacks on the RSA Cryptosystem: A Brief Survey," *Journal of Discrete Mathematical Sciences and Cryptography*, vol. 22, no. 1, pp. 9-29, 2019.

[35] Nguyen P. and Valle B., *The LLL Algorithm: Survey and Applications*, Springer Link, 2009.

[36] Rabah K., "Review of Methods for Integer Factorization Applied to Cryptography," *Journal of Applied Sciences*, vol. 6, no. 1, pp. 458-481, 2006.

[37] Rivest R., "Remarks on A Proposed Cryptanalytic Attack on The M.I.T. Public-Key Cryptosystem," Cryptologia, vol. 2, no. 1, pp. 62-65, 1978.

[38] Rivest R., Shamir A., and Adleman L., "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120-126, 1978.

[39] Shoup V., NTL: A Library For Doing Number Theory. https://www.shoup.net/ntl/, Last Visited, 2020.

[40] Simmons G., "A "weak" Privacy Protocol Using The RSA Crypto Algorithm," *Cryptologia*, vol. 7, no. 2, pp. 180-182, 1983.

[41] Simmons G. and Norris M., "Preliminary Comments on The M.I.T. Public Crypto-System," *Cryptologia*, vol. 1, no. 4, pp. 406-414, 1977.

[42] Takayasu A. and Kunihiro N., "Partial Key Exposure Attacks on RSA: Achieving the Boneh-Durfee Bound," *in Proceedings of International Conference on Selected Areas in Cryptography*, Montreal, pp. 345-362, 2014.

[43] Takayasu A. and Kunihiro N., "Partial Key Exposure Attacks on RSA: Achieving The Boneh-Durfee Bound," *Theoretical Computer Science*, vol. 761, pp. 51-77, 2019.

[44] Thome E., LISTSERV. https://listserv.nodak.edu/cgibin/wa.exe?A2=NMBRTHRY;fd743373.1912&S, Last Vested, 2020.

[45] Weger B., "Cryptanalysis of RSA with Small Prime Difference," *Applicable Algebra in Engineering, Communication, and Computing*, vol. 13, 2002.

[46] Wiener M., "Cryptanalysis of Short RSA Secret Exponents," *IEEE Transactions on Information Theory*, vol. 36, no. 3, pp. 553-558, 1990.

[47] Wikipedia. https://en.wikipedia.org/wiki/RSA_Factoring_Challenge Last Vested 2020.

[48] Wu M., Chen C., Lin Y., and Sun H., "On the Improvement of Wiener Attack on RSA with Small Private Exponent," *The Scientific World Journal*, pp. 1-9, 2014.

[49] Yan S., *Cryptographic attacks on RSA*, Springer Link, 2008.

[50] Yang Z., Fu S., Qu L., and Li C., "A Lower Dimension Lattice Attack on NTRU," *Science China Information Sciences*, vol. 61, no. 5, p. 1-9, 2018.

[51] Zimmermann P., https://lists.gforge.inria.fr/pipermail/cado-nfs-discuss/2020-February/001166.html, Last Visted, 2020.

Anas Ibrahim a Ph.D. (Computer Engineering) Candidate at Eastern Mediterranean University, North Cyprus. He is a Docent at Palestine Technical University, State of Palestine. His research interests in information security include symmetric and asymmetric ciphers, authentication and key exchange protocols, database security.

Alexander Chefranov holds the degree of a Ph.D. (Computer Science) and a Doctor of Engineering Sciences. He is currently an Associate Professor of the Department of Computer Engineering, Eastern Mediterranean University, Famagusta, North Cyprus. His research interests in information security include symmetric and asymmetric ciphers, authentication and key exchange protocols, database security.

Rushdi Hamamreh received his M.S. degree in computer engineering from the Saint Petersburg State Technical University in 1998 and his Ph.D. in Distributed Information Systems and Networks Security from the Saint Petersburg State Technical University in 2002. He is currently an associate professor of computer engineering at the department of computer engineering, Al-Quds University, Jerusalem. His research and teaching interests include design and development multiagent systems, Networks Security, Cybercrime law and Mobile Networks.