Journal of Artificial Intelligence and Engineering Applications

Website: https://ioinformatic.org/

15th June 2024. Vol. 3. No. 3; e-ISSN : 2808-4519

Cryptanalysis of RSA Using Algebraic and Lattice Methods

Faisal Amir Harahap[1]*, Yusfrizal[2], Mutiara Sovina[3], Ivi Lazuly[4]

[1, 2, 3, 4] Universitas Potensi Utama

faisalamirharahap5@gmail.com[1], yusfrizal80@gmail.com[2], mutiarasovina@gmail.com[3], ivilazuli2@gmail.com[4]*

Abstract

This paper applies tools from the geometry of numbers to solve several problems in cryptanalysis. We use algebraic techniques to cryptanalyze several public key cryptosystems. This paper focuses on RSA and RSA-like schemes, and use tools from the theory of integer lattices to get our results. We believe that this field is still underexplored, and that much more work can be done utilizing connections between lattices and cryptography. This paper studies the security of the RSA public key cryptosystem under partial key exposure. We show that for short public exponent RSA, given a quarter of the bits of the private key an adversary can recover the entire private key. Similar results (though not as strong) are obtained for larger values of the public exponent e. Our results point out the danger of partial key exposure in the RSA public key cryptosystem. This paper shows that if the secret exponent d used in the RSA public key cryptosystem is less than $N^{0.292}$, then the system is insecure. This is the first improvement over an old result of Wiener showing that when d is less than $N^{0.25}$ the RSA system is insecure.

Keywords: *Algebraic Techniques, Cryptosystems, Lattices, RSA*

1. Introduction

Cryptology is the study of secret codes. In speaking of cryptology, we discuss two main branches: cryptography is concerned with the writing of messages in secret code and the creation of these methods, while cryptanalysis is concerned with reading encrypted messages by breaking secret codes [1]. Ancient times saw many examples of cryptography. Over three thousand years ago, Egyptian scribes made use of hieroglyphic transformations to obscure the meaning of written messages. Mesopotamian and Babylonian scribes employed similar techniques to render cuneiform tablets unreadable to the uninitiated. The Greeks employed cryptography (and the closely related steganography, which is concerned with concealing the existence of communication rather than content) for military and tactical purposes [2]. Even the Kama sutra of ancient India touches on cryptography, listing secret writing as one of the sixty-four fundamental arts. Hundreds of other examples occur in ancient civilizations; in short, cryptography has appeared more or less spontaneously in every culture in which literacy has become widespread. Cryptanalysis, on the other hand, took considerably longer to develop as a rigorous subject of study. The earliest surviving descriptions of systematic methods of code breaking come from fifteenth century, in the Arabic encyclopedia. It gives the first known written description of the technique of frequency analysis, where the frequencies of letters and letter groupings of a language are used to unravel secret codes [3].

A method of securing communication is called a cryptosystem. The sender encrypts (or enciphers) a message using an encryption algorithm together with a secret key [4]. This produces a cipher text which is sent to the recipient. The recipient, who also possesses a key, receives the cipher text and decrypts (or deciphers) using the key to recover the original message, called the plaintext. The concept of public key cryptography circulated in the research community for some time before the first practical proposal for such a scheme was made. The RSA public key cryptosystem, named after inventors Ron Rivest, Adi Shamir, and Len Adleman, was introduced in Martin Gardner's column on Mathematical Games in Scientific American [5]. The RSA cryptosystem has survived over forty years of study by cryptanalysts in the public sector, and it is the most widely used public key cryptosystem in the world. It is used, among other places, in the SET protocol for secure credit card transactions and the SSL protocol for secure communication on the Internet [6].

Studying and using methods for breaking cryptosystems is an essential step in the development of new designs for more secure cryptosystems [7]. By learning how things break, we learn how to make them stronger. It is in this spirit that this research is written. This work uses mathematical tools to study the RSA public key cryptosystem and several variants. We use tools from numerical algebra and the geometry of numbers to get our results. Algebraic cryptanalysis has proven to be one of the most effective methods in the study of public key cryptosystems [8].

This research applies tools from the geometry of numbers to solve several problems in cryptanalysis. We use algebraic techniques to cryptanalyze several public key cryptosystems. We focus on RSA and RSA-like schemes, and use tools from the theory of integer lattices to get our results [9]. We believe that this field is still underexplored, and that much more work can be done utilizing connections between lattices and cryptography.

2. Research Methodology

These are the stages carried out in this research:

1. Preparation, this preparation stage is the beginning of the research process that will be carried out, the preparation carried out is determining the background of the problem, and this is done by looking for problems and obstacles that occur. Formulate what problems have occurred and what the resolution process will be. Providing problem boundaries, this is done to provide limitations to this research, namely starting from the data used, variables, applications or systems built and the output that will be produced. Determine the objectives, namely what results will be achieved from this research process [10].
2. Theoretical Study, at this stage a theoretical study will be carried out on the existing problem. The study was carried out to determine the concepts that will be used in the research, especially RSA Cryptosystem. Data Collection, this stage is intended to collect supporting data obtained from Android malware detection. For this reason, several methods were used to collect data, namely Observation. Data Analysis, at this stage, supporting data will be analyzed [11]. Testing and Implementation, at this stage data variables and data implementation will be tested as well as system program preparation, namely by preparing the data to be analyzed. Determine the parameters for data testing using a RSA Cryptosystem. Classifying the results of the system work process, namely the results of RSA Cryptosystem [10].
3. Final stage, this stage is the stage of drawing conclusions and suggestions that can be made in preparing this research. With the conclusions, the results of the research carried out will be known and it is hoped that with suggestions there will be improvements and benefits for others. This conclusion is to answer what is the problem formulation based on the analysis that has been carried out in the previous stages [12].

2.1. Representations of Lattices

In order to discuss the running times of algorithms operating on lattices we must describe the representation of lattices given as input to these algorithms. Suppose $u_1, \ldots, u_w$ are vectors in Z^n. The running time of an algorithm with input $(u_1, \ldots, u_w)$ is parameterized by w, n, and by the largest element of the u_i, defined by $u_{largest} := max_{i,j} u_{ij}$. For lattices in Q_n, the situation is slightly more complex. Suppose we are given the vectors $u_1, \ldots, u_w$ in Q^n. There is some least integer D such that $Du_1, \ldots, Du_w$ are all in Z^n. Then we define the largest element by $u_{largest} := max_{i,j} (Du_{ij})$.

Much of the work described in this thesis was inspired by the seminal work of Coppersmith for finding small solutions to polynomial congruence's. We use this very effective technique as a starting point for many of our results. In subsequent chapters we will apply and extend this technique to solve a number of cryptanalytic problems, and discuss subtleties in its implementation and use. In this section we will introduce the general Coppersmith approach and provide a few simple examples [13]. We use a simplified version due to Howgrave-Graham. We note that the generic result is in some sense "blind" to the actual polynomial being used (it takes into account only the degree, but not the coefficients), and that there may be a more optimal choice of polynomials $g_{i,k}$ to include in the lattice to solve a particular problem. In this paper, we will see an example that improves over this generic result by taking into account an optimized set of polynomials.

2.2. Public Key Cryptography

A public key (or asymmetric) cryptosystem is a method for securing communication between parties who have never met before. More precisely, a public key cryptosystem is described by the following:

1. a set M of plaintexts (or messages), and a set C of cipher texts;
2. a set K_p of public keys, and a set K_s of secret keys;
3. a key generation algorithm key-gen : $Z \rightarrow K_p \times K_s$;
4. an encryption algorithm $E : K_p \times M \rightarrow C$; and,
5. a decryption algorithm $D : K_s \times C \rightarrow M$.

The key generation, encryption, and decryption algorithms can be randomized and should run in expected time polynomial in the length of their inputs [14]. For all (K_p,K_s) output by key-gen and all messages $M \in M$ we must have that $D(K_s,E(K_p,M)) = M$. The input to the key generation algorithm is called the security parameter. The hope is that as the security parameter increases, the resources required to break the cryptosystem using the resulting keys should increase more rapidly than the resources required to use it. Ideally, the running time of a break should be a (sub-) exponential function of n, while the running time of key-gen, E, and D should be some (small) polynomial in n.

Suppose Alice is a user of a public key cryptosystem. To initialize she chooses a security parameter n and computes (K_p,K_s) := key-gen(n). When another user Bob wishes to send a message to Alice securely, he obtains Alice's public key K_p and computes the cipher text $C := E(K_p,M)$. He sends cipher text C is sent to Alice, who upon obtaining it computes the original message $M = D(K_s,C)$. The security requirements of a cryptosystem can be defined in many ways. In general, when defining a security goal it is important to state what resources are available to an attacker and what success criteria the attacker must fulfill. A very basic requirement is that it should not be possible to derive the secret key from the public key efficiently; indeed, it is considered the most devastating cryptanalytic break to compute K_s from K_p in (say) time polynomial in the security parameter. There are many issues that arise in determining good notions of security, and we do not try to address them all here. There are many good surveys on the subject.

Let n be a security parameter. The key generation algorithm for RSA computes primes p and q approximately n/2 bits in length, so that N := pq is an integer n bit in length [15]. More precisely, p and q are random primes subject to the constraint that N = pq is an n-bit number.

As discussed earlier, a break of the RSA public key cryptosystem can be defined in several ways. Most obviously the scheme is broken if an attacker is able to recover the secret exponent d. Since factorization of the modulus N = pq leads to recovery of the private key d, this is also a total break. All of the attacks presented in subsequent chapters are of this type, and involve either a direct computation of the private key d or one of the factors p of the public modulus N, given the public key information (N, e) alone. We emphasize that our results come from the basic RSA equations; our attacks do not use plaintext/cipher text pairs or signatures, so they hold regardless of any padding schemes used. It is an interesting open question to determine if the attacks presented in this work can be improved if a particular padding is in use, or if the adversary is given access to known or chosen plaintext/cipher text pairs or chosen signatures [16].

2.3. Partial Key Exposure Attacks

Let N = pq be an RSA modulus and let e, d be encryption/decryption exponents, i.e., $ed \cong 1 \bmod \phi(N)$. How many bits of d does an adversary require in order to reconstruct all of d? Surprisingly, we show that for short public exponent RSA, given only a quarter of the least significant bits of d, an adversary can efficiently recover all of d. We obtain similar results, summarized below, for larger values of e as well. Our results show that the RSA public key cryptosystem, and particularly low public exponent RSA, are vulnerable to partial key exposure. We refer to this class of attacks as partial key exposure attacks.
To motivate this problem consider a computer system which has an RSA private key stored on it. An adversary may attempt to attack the system in a variety of ways in order to obtain the private key. Some attacks (e.g. a timing attack) are able to reveal some bits of the key, but may fail to reveal the entire key. Our results show that attacks, such as the timing attack on RSA, need only be carried out until a quarter of the least significant bits of d are exposed. Once these bits are revealed the adversary can efficiently compute all of d. Another scenario where partial key exposure comes up is in the presence of covert channels. Such channels are often slow or have a bounded capacity [16]. Our results show that as long as a fraction of the private exponent bits can be leaked, the remaining bits can be reconstructed.
Let N = pq be an n-bit RSA modulus. The reader is reminded that we view the private exponent d as an n-bit string, so that when referring to the t most significant bits of d we refer to the t leftmost bits of d when viewed canonically as an n-bit string. For instance, it is possible that the t most significant bits of d are all zero, for some t. Similarly, a quarter of the bits of d always refer to n/4 bits.

2.4. Attacks on Short Secret Exponent RSA

If the secret exponent d used in the RSA public key cryptosystem is less than $N^{0.292}$, then the system is insecure. This is the first improvement over an old result of Wiener showing that when d is less than $N^{0.25}$ the RSA system is insecure. To motivate this problem, consider the problem of speeding up RSA signature generation. The most computationally expensive step in signature generation is the operation $m \rightarrow m^d \bmod N$. Thus one is tempted to use a short secret exponent d. Unfortunately, Wiener showed over ten years ago that if one uses $d < N^{0.25}$ then the RSA system can be broken. Since then there have been no improvements to this bound. Verheul and Tilborg showed that as long as $d < N^{0.5}$ it is possible to expose d in less time than an exhaustive search; however, their algorithm requires exponential time as soon as $d > N^{0.25}$. Wiener describes a number of clever techniques for avoiding his attack while still providing fast RSA signature generation. One such suggestion is to use a large value of e. Indeed, Wiener's attack provides no information as soon as $e > N^{1.5}$ [17].

3. Results and Discussion

The first scheme uses unbalanced RSA module, that is, N = pq for p < q. The recommended parameters are chosen to defeat the Wiener attack and the attacks of the previous topic.

Scheme I

1. Fix desired bit-length n for N, bit-length l_d for d, bit-length l_p for p, and security parameter $\gamma > 64$, subject to the following constraints:
   $l_d + l_p > n/3$, and
   $l_d > \gamma + l_p/2$.
   Note that the smaller l_p is, the smaller l_d is allowed to be.
2. Select the prime p of bit length l_p and the prime q of bit length $n-l_p$ uniformly at random. Note that both n and l_p must be large enough so that N = pq cannot be factored efficiently by ECM or NFS.
3. Pick the secret exponent d uniformly at random from l_d-bit integers.
4. If the public exponent e defined by $ed \cong 1 \bmod \phi(N)$ is not larger than $\phi(N)/2$, go to Step 3.

A choice of parameters suggested by the authors is: p is a 256-bit prime, q is a 768-bit prime, and d is a 192-bit number. Note that 192 is far below the Wiener bound of 256 bits, and the bound of 299 bits achieved in the previous chapter, yet the choice of parameters foils both attacks.

Scheme II

The second scheme selects one of the primes in such a way that allows e and d to be simultaneously small.

1. Fix the bit-length n of N, and set $l_p = (n/2) - 112$ and $l_d = (n/2) + 56$.
2. Select a random prime p of l_p and a random k of 112 bits.
3. Select a random d of l_d bits such that $\gcd(d, k(p-1)) = 1$.
4. Compute u and v such that $du - k(p-1)v = 1$ with $0 < u < k(p-1)$ and $0 < v < d$.
5. If $\gcd(v+1, d) \neq 1$ then return to Step 3.
6. Select a random h of 56 bits until $q = v + hd + 1$ is prime.
7. Set $e := u + hk(p-1)$ and N = pq.

Notice that e and d satisfy the equation $ed = 1 + k\phi(N)$. They both have approximate bit-length (n/2) + 56. The primes p and q have approximate bit-length (n/2) – 112 and (n/2) + 112 respectively. A possible choice of parameters for Scheme (II) might be: p a 400-bit prime, q a 624-bit prime, and e and d are each 568-bit integers.

Scheme III

The third scheme is a mix of the first two schemes, allowing a trade-off between the lengths of e and d. More precisely, the scheme is a parameterized version of scheme (II), where p, k, d and h have bit-lengths l_p, l_k, l_d, and $n - l_p - l_d$, respectively. These parameters may be chosen freely, but to resist various attacks, the following is required:

1. $lp > n/2$;
2. $l_k >> l_p - l_d + 1$;

3. $l_k + l_p > n/3$; and
4. k must withstand an exhaustive search, i.e., $l_k > 64$.

A choice of parameters suggested by the authors is: p is a 256-bit prime, q is a 768-bit prime, e is an 880-bit number, and d is a 256-bit number.

We ran several dozen experiments to test our results when $d > N^{0.25}$. Our experiments were carried out using the LLL implementation available in Victor Shoup's NTL package. In all our experiments LLL produced two independent relations $H_1(x, y)$ and $H_2(x, y)$. In every case, the resultant $H(y) := Res_x(H_1(x, y),H_2(x, y))$ with respect to x was a polynomial of the form $H(y) = (y + p + q)H_0(y)$, with $H_0(y)$ irreducible over Z (similarly for x). Hence, the unique solution (x_0, y_0) was correctly determined in every trial executed. We show the parameters of some attacks executed in Table 1. In each of these tests, d was chosen uniformly at random in the range. The second to last row of the table is especially interesting as it is an example in which our attack breaks RSA with a private key that is 60 bits larger than Wiener's bound.

Table 1: Running times for short secret exponent attack

N	d	δ	m	t	Lattice dim	Running time	Advantage over Wiener's attack
1024 bits	283 bits	0.277	7	3	45	2.5 hours	27 bits
2048 bits	550 bits	0.275	7	3	45	16 hours	50 bits
4096 bits	1060 bits	0.265	5	2	25	3 hours	60 bits
10000 bits	2550 bits	0.255	3	1	11	19 minutes	50 bits

This paper implemented this attack using Victor Shoup's Number Theory Library and the Maple Analytical Computation System. The attack runs very efficiently, and in all instances of Schemes (I) and (III) we tested, it produced algebraically independent polynomials H1(x, y) and H2(x, y). These yielded a resultant $H(y) = (y - p)H0(y)$, where H0(y) is irreducible, exposing the factor p of N in every instance. This strongly suggests that this "heuristic" assumption needed to complete the multivariate modular version of Coppersmith's technique is extremely reliable, and we conjecture that it always holds for suitably bounded lattices of this form. The running times of our attacks are given in Table 2.

Table 2: Running times of attacks on Sun-Yang-Laih schemes

Scheme	Size of n	Size of p	Size of e	Size of d	m	t	a	Lattice rank	Running time
I	1024	256	1024	192	3	1	1	20	40
III	1024	256	880	256	2	3	0	15	9

4. Conclusion

Based on the results of the design, discussion and testing of the system implementation carried out. So it can be concluded as follows:

1. This is similar to the situation with many factoring algorithms, where one cannot prove that they work; instead one gives strong heuristic arguments that explain their running time. In our case, the heuristic "assumption" we make is that the two shortest vectors in an LLL reduced basis give rise to algebraically independent polynomials.
2. We showed that unbalanced RSA actually improves the attacks on short secret exponent by allowing larger exponent. This enabled us to break most of the RSA schemes with short secret exponent from Asiacrypt '99. The attack extends the attack of the previous chapter by using a "trivariate" version of Coppersmith's lattice based technique for finding small roots of low-degree modular polynomial equations. This attack is provable up to the step where the bivariate heuristic assumption is required.
3. These results illustrate once again the fact that one should be very cautious when using RSA with short secret exponent. Again, the best method to enjoy the computational advantage of short secret exponent is the countermeasure proposed by Wiener to use a secret exponent d such that both d mod (p − 1) and d mod (q − 1) are small. Such a d speeds up RSA signature generation when the signature is generated modulo p and q separately and combined using the Chinese Remainder Theorem. Classical attacks do not work since d is likely to be close to $\phi(N)$. It is an open problem whether there is an efficient attack on such secret exponents.

Acknowledgement

Thank you to all those who have helped complete this research.

References

[1] M. F. Brown, "Encryption Systems Resilient against Quantum Cracking." Utica College, 2019.
[2] P. Kestner, "Codes and Secret Messages: From the Ancient Origins of Steganography and Cryptography and their Relevance to Today," in *The Art of Cyber Warfare: Strategic and Tactical Approaches for Attack and Defense in the Digital Age*, Springer, 2024, pp. 59–111.
[3] E. Göransson, G. V. M. Haverling, S. O'Sullivan, O. Merisalo, I. Ventura, and P. A. Stokes, "Textual traditions." De Gruyter, 2020.
[4] A. A. Mohammed and H. A. Anwer, "A New Method Encryption and Decryption.," *Webology*, vol. 18, no. 1, 2021.
[5] F. Rubin, *Secret Key Cryptography: Ciphers, from Simple to Unbreakable*. Simon and Schuster, 2022.
[6] D. D. Kumar, J. D. Mukharzee, C. V. D. Reddy, and S. M. Rajagopal, "Safe and Secure Communication Using SSL/TLS," in *2024 International Conference on Emerging Smart Computing and Informatics (ESCI)*, 2024, pp. 1–6.
[7] O. A. Alzubi, J. A. Alzubi, O. Dorgham, and M. Alsayyed, "Cryptosystem design based on Hermitian curves for IoT security," *J. Supercomput.*, vol. 76, no. 11, pp. 8566–8589, 2020.
[8] A. Bakhtiyor, K. Zarif, A. Orif, and B. Ilkhom, "Algebraic Cryptanalysis of O'zDSt 1105: 2009 Encryption Algorithm," in *2020 International Conference on Information Science and Communications Technologies (ICISCT)*, 2020, pp. 1–7.
[9] C. Lee, A. Pellet-Mary, D. Stehlé, and A. Wallet, "An LLL algorithm for module lattices," in *International Conference on the Theory and*

Application of Cryptology and Information Security, 2019, pp. 59–90.
[10] D. R. Hancock, B. Algozzine, and J. H. Lim, "Doing case study research: A practical guide for beginning researchers," 2021.
[11] M. Mumtaz and L. Ping, "Forty years of attacks on the RSA cryptosystem: A brief survey," *J. Discret. Math. Sci. Cryptogr.*, vol. 22, no. 1, pp. 9–29, 2019.
[12] V. K. Chauhan, K. Dahiya, and A. Sharma, "Problem formulations and solvers in linear SVM: a review," *Artif. Intell. Rev.*, vol. 52, no. 2, pp. 803–855, 2019.
[13] A. May, "Lattice-based integer factorisation: an introduction to coppersmith's method," *Comput. Cryptogr. Algorithmic Asp. Cryptol.*, pp. 78–105, 2021.
[14] Z. K. Abdalrdha, I. H. Al-Qinani, and F. N. Abbas, "Subject review: key generation in different cryptography algorithm," *Int J Sci Res Sci Eng Technol*, vol. 6, no. 5, pp. 230–240, 2019.
[15] G.-C. Kim, S.-C. Li, and H.-C. Hwang, "Fast rebalanced RSA signature scheme with typical prime generation," *Theor. Comput. Sci.*, vol. 830, pp. 1–19, 2020.
[16] S. Varghese and S. M. C. Vigila, "A Novel Method for Mapping Plaintext Characters to Elliptic Curve Affine points over Prime Field and Pseudorandom Number Generation," *Int. J. Comput. Inf. Syst. Ind. Manag. Appl.*, vol. 12, p. 8, 2020.
[17] J. M. Almira, *Norbert Wiener: A Mathematician Among Engineers*. World Scientific, 2022.