Security Analysis of Homomorphic Encryption Scheme for Cloud Computing: Known-Plaintext Attack

Mikhail Babenko[1], Nikolay Chervyakov[2]
North-Caucasus Federal University
Stavropol, Russia
[1]mgbabenko@ncfu.ru, [2]ncherviakov@ncfu.ru

Andrei Tchernykh[3]
CICESE Research Center
Ensenada, BC, México
[3]chernykh@cicese.mx

Nikolay Kucherov[4], Maxim Deryabin[5]
North-Caucasus Federal University
Stavropol, Russia
[4]nkucherov@ncfu.ru, [5]maderiabin@ncfu.ru

Gleb Radchenko[6]
South Ural State University
Chelyabinsk, Russia
[6]gleb.radchenko@susu.ru

Philippe OA Navaux[7]
Federal University of Rio Grande do Sul
Porto Alegre, Brazil
[7]navaux@inf.ufrgs.br

Viktor Svyatkin[8]
North-Caucasus Federal University
Stavropol, Russia
[8]qiwi3011@mail.ru

Abstract— **We consider cryptosystems for homomorphic encryption schemes based on the Residue Number System (RNS) and Secret Sharing Schemes. One of their disadvantages is that they are directly related to data redundancy, and hence, increasing the size of the storage. To minimize it, homophonic encryption can be combined with the arithmetic coding known as Chinese remainder theorem. We describe a new method of cryptanalysis based on a property of RNS and theory of numbers. We prove that an attacker needs only $\boldsymbol{n \cdot \lceil \log_2 \log_2 (k \cdot p_n) \rceil}$arbitrary generated input files that form the "known-plaintext", where $\mathbf{p_i}$ is moduli RNS, to calculate the secret key required to decrypt the entire data.**

Keywords— known-plaintext attack; homomorphic encryption; cloud computing

I. Introduction

Cloud computing along with technical advantages have disadvantages. The limitation of cloud technologies is that cloud servers can not perform computations over encrypted data without first decrypting it. Homomorphic encryption allows computations over encrypted data without their preliminary decryption. When processing confidential data using cryptographic methods to protect information from untrusted servers, it is possible to transfer keys to decrypt data only to trusted servers [1-5].

To solve the problem of security, modern algorithms for symmetric and asymmetric encryption are not suitable, because they do not allow the security of information processed in the clouds. Homomorphic encryption allows confidentiality of the information and results of the computations. In the implementation of computational algorithms using homomorphic ciphers, there is a problem associated with the conversion of the algorithm into the set of operations supported by a homomorphic cipher. Efficient use and implementation of homomorphic ciphers become important.

Homomorphic encryption has the following applications:

- Cloud computing
- Electronic voting
- Secure information search
- Secure wireless decentralized communication networks
- Outsourcing services for smart cards
- Feedback systems
- Obfuscation for software security

Homomorphic ciphers can be:

- Partially homomorphic
- Fully homomorphic

Partially homomorphic cryptosystems are cryptosystems that are homomorphic with respect to only one algebraic operation (either addition or multiplication).

A cryptosystem that supports both addition and multiplication is called fully homomorphic. Fully homomorphic cryptographic systems allow processing of data in encrypted form.

The homomorphic property of various cryptographic systems can be used to create secure voting systems, hash functions that are resistant to collisions, private information of search engines and makes possible widespread use of public cloud computing, ensuring the confidentiality of processed data.

Gentry, 2010 [6] proposed a fully homomorphic scheme of the form $c = pq + m$, where c is the encrypted text, m is the original message, q is a random number, p is the key. This encryption function is homomorphic with respect to addition, subtraction and multiplication.

From here we assume that m is the residue from division of c by p. In other words, the encryption function is inverse with respect to the rest of the operation.

One of the approaches to constructing fully homomorphic ciphers with adjustable redundancy is the secret sharing scheme based on residue number system (RNS) [1]. In [7] schemes of homomorphic encryption of two types are proposed:

confidentiality of data and confidentiality of moduli. Homomorphic encryption with data confidentiality uses the same approach as Asmuth-Bloom secret sharing scheme [8]. With random noise, Homomorphic encryption with data confidentiality imposes restrictions on the size of data and reduces the range of encrypted information. Hence, it is difficult to implement arithmetic operations efficiently.

Homomorphic encryption with the confidentiality of moduli is inapplicable to problems where there is modular exponentiation or a large number of modular multiplications of different numbers. Examples of this kind of problems are systems of linear equations over a prime field with the Gaussian method [9], implementation of encryption systems RSA [10], Diffie Helfman [11] and El-Gamal [12] over a prime field.

When using homomorphic encryption to ensure the security of information in the cloud computing, the probability of distortion of one or more chunks of the result is high.

II. Residue number system and its properties

Residue number system is a basic structure of modular arithmetic. Numerous studies reveal the capabilities of this system in the field of increasing the efficiency of data processing and information security [1]. The special to RNS number representation makes it possible in some cases to execute operations in parallel, which leads to greater efficiency with respect to the positional representation.

Roughly speaking, residue number system is a non-positional modular representation of numbers. Each digit in RNS is the remainder of division by a certain number, called the modulus. All the moduli for each digit form RNS moduli set. The uniqueness of a number representation in RNS is only guaranteed in case of pairwise coprime moduli. Let $\{p_1, p_2, \dots, p_n\}$ – be an RNS moduli set. It defines a unique RNS. If a number A is represented in a positional number system then it can be represented in RNS as follows:

$$S \to (s_1, s_2, \dots, s_n), \text{ where } s_i = |S|_{p_i} \text{ for } i = 1,2,\dots,n.$$

Note that the number S should belong to the interval $S \in [0, P)$, where $P = p_1 \cdot p_2 \cdot \dots \cdot p_n$ is called RNS dynamic range.

Advantages of RNS hold for so called modular operations that do not require carries between digits. Examples of these operations are multiplication, addition, subtraction and exponentiation. Operations $C = A + B \bmod P$ and $D = A \cdot B \bmod P$ for numbers $A \to (a_1, a_2, \dots, a_n)$ and $B \to (b_1, b_2, \dots, b_n)$ represented in RNS are defined as follows:

$$C = A + B \bmod P \to (|a_1 + b_1|_{p_1}, \dots, |a_n + b_n|_{p_n}),$$

$$D = A \cdot B \bmod P \to (|a_1 \cdot b_1|_{p_1}, \dots, |a_n \cdot b_n|_{p_n}),$$

Modular computations are carry-free. It allows to execute them in parallel concerning each digit, which increases the speed of the entire algorithm. However, for the division, magnitude comparison, sign and some other operations in RNS, which are required in practice, this property does not hold. They are called non-modular operations that are positional.

RNS is often used for data sharing and error correction. Its natural parallelism is an efficient tool for distributed information representation, which can be used to build threshold data access structures. Popular examples of such structures are Asmuth-Bloom and Mignotte secret sharing schemes. Moreover, due to operand size reduction in RNS w.r.t. to the original numbers, these schemes can be more efficient in terms of the required resources (e.g. memory and power consumption) compared with other approaches, such as Shamir and Blackley schemes.

One of the most important RNS applications is error correction. To ensure error correction, it is required to add redundant (control) moduli to RNS moduli set. Thus, all channels transmit and process information, while during error detection and localization wrong data is eliminated. This is more efficient than if a separate error correction mechanism is applied.

Error correction properties of RNS are one of its main advantages over other ways of data representation. Thus, linear or (potentially) parallel systems can become fault-tolerant if control moduli are added.

III. Secret Sharing Scheme

Secret sharing scheme (SSS) is a method where an administrator distributes shares (shadows) to different participants such that only authorized subsets of participants will be able to reconstruct the secret. This technique was originally focused on problems of secure information storage. However, secret sharing schemes have found numerous other applications in cryptography and distributed computing. The size of the shares is exponential in the number of participants, which is a big drawback of this technique.

SSS is used in many secure protocols, e.g. they allow building secure distributed storage systems [1]. However, if SSS is not homomorphic, computation over encrypted data is impossible or complex. For the construction of a fully homomorphic encryption, [5] propose schemes based on RNS. The RNS allows building fully homomorphic ciphers due to the properties of the parallelism of arithmetic operations, which improve performance and develop a homomorphic encryption that is asymptotically perfect and balanced, depending on the task.

IV. Known-Plaintext Attack

An attempt to break a specific cipher using cryptanalysis methods is called a cryptographic attack on this cipher. The cryptographic attack, as a result of which the cipher was broken, is called break of the cipher.

Main methods of cryptanalysis are [13]:

- Known-ciphertext attack
- Known-plaintext attack
- Chosen-plaintext attack
- Adaptive chosen-plaintext attack

Now, we consider these attacks in details.

A. Known-ciphertext attack

WEP – is a security algorithm for wireless networks.

Frame format for WEP is:

1. Open part;
1. Initialization vector (24 bits);
2. Empty space (6 bits);
3. Key id (2 bits);
2. Encrypted part
1. Data;
2. Control sum (32 bits).

The data is ciphered with RC4. To attack WEP, it is necessary to eavesdrop and analyze the transmitted data. All the attacks are based on the weakness of RC4. There are three main types of the attacks:

Fluhrer, Mantin and Shamir attack [14] - is based on the use of weak initialization vectors. With a weak initialization vector, an attacker knowing the first byte of the keystream and the first m bytes of the key can derive the $(m + 1)$ -th byte of the key due to a weakness in the PRNG used to generate the keystream. Since the first byte of the plaintext is defined by SNAP, an attacker can obtain the first byte of the keystream.

KoreK attack [15] uses 17 different attacks, which help to determine $K[l]$, if preceding keystream bytes and first two ciphertext words are known.

Tevsa-Weinman-Pyshkin attack [16] uses the injection of ARP queries into the wireless network. It is the most efficient attack; it only requires several ten thousands of frames.

B. Chosen-plaintext attack

The possibilities of chosen-plaintext attacks are quite rich. For example, an attacker could bribe someone who encrypts the selected message. The following situation is also possible: the attacker sends a message to the ambassador of a certain country, and he sends it to his homeland in the encrypted form [13].

A chosen-plaintext attack is an active attack on a cryptosystem. Such attacks violate the integrity and confidentiality of the transmitted information.

An attacker receives the ciphertext $C = E(P, K)$ from the user, where P – is the chosen by the cryptanalyst plain-text; C – is the ciphertext; E – is the encryption function. The aim of the attacker is to find the plaintext $P' = P$. Since the attacker has the access to the cipher block E, he can cipher his messages P_i and analyze the obtained ciphertexts $C_i = E(P_i, K)$. As a result, the attacker finds the message P' and sends it to the user.

There are two types of Chosen-plaintext attacks:

- batch chosen-plaintext attack
- adaptive chosen-plaintext attack (CPA2)

In known plaintext attack, the cryptanalyst has plaintext and the corresponding ciphertext. This type of attack is more powerful than the known ciphertext attacks because more information is known about the cryptosystem.

Chosen plaintext attack is a more powerful type of attack than known plaintext attack. The ability to choose plaintexts provides more options for breaking the system key. It is also true that if a cryptosystem is vulnerable to known plaintext attack, then it is also vulnerable to chosen plaintext attack [17].

C. Adaptive chosen-plaintext attack

The adaptive chosen-plaintext attack often uses the methods of differential cryptanalysis, which are described in the works of Shamir, and also linear cryptanalysis. The general method of such attacks is as follows:

Differential cryptanalysis algorithm:

1. A pair of plaintexts with a predefined difference is chosen
2. Plaintext is ciphered
3. Corresponding ciphertexts also have some difference
4. Many pairs plaintext – ciphertext are collected
5. Differences in plaintexts and corresponding ciphertexts are compared
6. Based on the collected data the assumption about the key is made

Linear cryptanalysis algorithm:

The goal of the cryptanalyst is to obtain the following one-round expression:

$$P_{i_1} \oplus P_{i_2} \oplus \dots \oplus P_{i_a} \oplus C_{j_1} \oplus C_{j_2} \oplus \dots \oplus C_{j_b} = K_{k_1} \oplus \dots \oplus K_{k_c},$$

where P_i is the i-th bit of the plaintext, C_j is the j-th bit of the ciphertext, K_k is the k-th bit of the keystream. This expression is called a linear approximation. For a random plaintext, ciphertext and keystream the probability of this message is roughly $1/2$. This probability can be increased, if special algorithms are used.

Approximation with a given condition on the probability can be found quite easily. The problem arises when the method is extended to full-round encryption. An exact solution to this problem is not realistic since the cryptanalyst will have to analyze all possible versions of plaintexts and ciphertexts. But after some assumptions, it becomes possible to use the piling-up lemma.

The adaptive chosen-plaintext attack is an improved version of the chosen -plaintext attack [18]. The advantage of this type of attack is the fact that the cryptanalyst can choose a new plaintext based on the available results, whereas in the case of chosen plaintext attack all plaintexts are chosen before the beginning of the attack.

V. Mignotte Secret Sharing Scheme Cryptanalysis

For the analysis of Mignotte secret sharing schemes, we use the statistical properties of the cipher text similar to the approach in [19]. We note that to solve collusion problem in cloud computing, RNS moduli set $\{p_1, p_2, \ldots, p_n\}$ becomes the secret key. Without loss of generality, let us assume that

$$p_1 < p_2 < \cdots < p_n .$$

Let the malicious user know m different pairs «plaintext/ciphertext». We denote the set of plaintexts $\{S_1, S_2, \ldots, S_m\}$ and the ciphertext: $S_i \to (s_{i,1}, s_{i,2}, \ldots, s_{i,n})$.

Theorem 1. If $m \geq n \cdot \lceil \log_2 \log_2 (k \cdot p_n) \rceil$, then the malicious user can unambiguously determine the secret key in the Mignotte secret sharing scheme.

Proof

In the Mignotte scheme the malicious user has the following information:

$$S_i \equiv s_{i,j} \bmod p_j, \tag{1}$$

where $i = \overline{1,m}$ and $j = \overline{1,n}$.

From congruence (1) it follows that for all $i = \overline{1,m}$ and $j = \overline{1,n}$ it holds that: $p_j | (S_i - s_{i,j})$, hence:

$$p_j | \gcd(S_1 - s_{1,j}, S_2 - s_{2,j}, \ldots, S_m - s_{m,j}), \tag{2}$$

where $j = \overline{1,n}$.

According to fundamental theorem of arithmetic $S_1 - s_{1,j}$ can be uniquely represented as the product of prime numbers up to the order of the factors so $S_1 - s_{1,j} = m_{j,1} \cdot m_{j,2} \cdot \ldots \cdot m_{j,L_j}$.

From expression (2), it follows that for all $j = \overline{1,n}$ modulus $p_j = m_{j,k_1} \cdot \ldots \cdot m_{j,k_{l_j}}$, where $1 \leq l_j \leq L_j$. In the worst case scenario $L_j = \lfloor \log_2 (S_1 - s_{1,j}) \rfloor$ and $l_j = \lfloor \log_2 p_j \rfloor$.

Let $S_2 = m_{j,1} \cdot m_{j,2} \cdot \ldots \cdot m_{j,\left\lfloor \frac{L_j}{2} \right\rfloor}$ and $S_3 = m_{j,\left\lfloor \frac{L_j}{2} \right\rfloor + 1} \cdot \ldots \cdot m_{j,L_j}$. We look at three cases:

Case 1. If $s_{2,j} = 0$ and $s_{3,j} \neq 0$, then $p_j | S_2$.

Case 2. If $s_{2,j} \neq 0$ and $s_{3,j} = 0$, then $p_j | S_3$.

Case 3. If $(s_{2,j} \neq 0) \& (S_2 \neq s_{2,j})$ and $(s_{3,j} \neq 0)) \& (S_3 \neq s_{3,j})$, then $p_j = \gcd(S_2, s_{2,j}) \cdot \gcd(S_3, s_{3,j})$ otherwise $S_2 < p_j$ and $S_3 < p_j$

In case 1 and 2, we divide the interval in half $\left[1, \left\lfloor \frac{L_j}{2} \right\rfloor\right]$ and $\left[\left\lfloor \frac{L_j}{2} \right\rfloor + 1, L_j\right]$ and then repeat the procedure. In case 3 the interval $\left[\left\lfloor \frac{L_j}{2} \right\rfloor + 1, L_j\right]$ is divided in half and $S_4 = S_2 \cdot S_4'$ и $S_5 = S_2 \cdot S_5'$ is computed and the algorithm is repeated.

In the worst case, we need $\lceil \log_2 L_j \rceil$ pairs «plaintext/ciphertext» to compute one modulus, hence to compute n moduli we need $\sum_{j=1}^{n} \lceil \log_2 L_j \rceil \leq n \cdot \lceil \log_2 \log_2 S \rceil$. Since $S < k \cdot p_n$, then $\sum_{j=1}^{n} \lceil \log_2 L_j \rceil \leq n \cdot \lceil \log_2 \log_2 (k \cdot p_n) \rceil$.

Theorem is proved.

The properties of known plaintext attack allow us to compute the secret key and gain the full access to the data.

VI. Conclusions

From Theorem 1, it follows that the schemes proposed in [20-24] do not resist plaintext attack. However, we should note that the proposed algorithm uses factorization algorithm, which is computationally complex. Thus in case of a big k computational complexity of secret key computation increases dramatically.

Acknowledgment

The work is partially supported by Russian Federation President Grant SP-1215.2016, MK-6294.2018.9, and Russian Foundation for Basic Research (RFBR) 18-07-01224.

References

[1] N. Chervyakov, M. Babenko, A. Tchernykh, N. Kucherov, V. Miranda-López, J. M. Cortés-Mendoza, "AR-RRNS: Configurable reliable distributed data storage systems for Internet of Things to ensure security," *Future Generation Computer Systems,* 2017. DOI: 10.1016/j.future.2017.09.061

[2] M. Akkal, P. Siy, "A new Mixed Radix Conversion algorithm MRC-II,' *Journal of Systems Architecture*, vol. 53, no. 9, pp. 577-586, 2007. DOI: 10.1016/j.sysarc.2006.12.006

[3] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, M. Zaharia, "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50-58, 2010. DOI: 10.1145/1721654.1721672

[4] S. Andrew, S. M. Van, Distributed systems: principles and paradigms, Prentice Hall. Upper Saddle River (NJ); 2007.

[5] C. A. Asmuth, J. Bloom, "A Modular Approach to Key Safeguarding," *IEEE Transactions on Information Theory*, vol. 29, no. 2, pp. 208-210, 1983. DOI: 10.1109/TIT.1983.1056651

[6] C. Gentry, "Computing arbitrary functions of encrypted data," *Communications of the ACM*, vol. 53, no. 3, pp. 97-105, 2010. DOI: 10.1145/1666420.1666444

[7] N. I. Chervyakov, M. G. Babenko, N. N. Kucherov, A. I. Garianina, "The effective neural network implementation of the secret sharing scheme with the use of matrix projections on FPGA," *Lecture Notes in Computer Science*, vol. 9142, pp. 3-10, 2015. DOI: 10.1007/978-3-319-20469-7_1

[8] L. Bai, "A reliable (k, n) image secret sharing scheme," in *2nd IEEE International Symposium on Dependable Autonomic and Secure Computing*, 2006, pp. 31–36. DOI: 10.1109/DASC.2006.11

[9] N.I. Chervyakov, M.G. Babenko, P.A. Lyakhov, I.N. Lavrinenko, "An Approximate Method for Comparing Modular Numbers and its Application to the Division of Numbers in Residue Number Systems," *Cybernetics and Systems Analysis*, vol. 50, no. 6, pp. 977–984, 2014. DOI: 10.1007/s10559-014-9689-2

[10] C.A. Gayoso, C. Gonzalez, L. Arnone, M. Rabini, M. J. Castineira, "Pseudorandom number generator based on the residue number system and its FPGA implementation," in *EAMTA*, 2013, pp. 9–14.

[11] G. Blakley, "Safeguarding crypographic keys," in *Proceedings of the AFIPS,* 1979, vol. 48, pp. 313-317.

[12] N. I. Chervyakov, M. G. Babenko, V. A. Kuchukov, "Research of effective methods of conversion from positional notation to RNS on FPGA," in *EIConRus*, 2017, pp. 277-281. DOI: 10.1109/EIConRus.2017.7910546

[13] B. Schneier, Applied cryptography: protocols, algorithms, and source code in C, Wiley & Sons, 2007.

[14] S. Fluhrer, I. Mantin, A. Shamir, "Weaknesses in the key scheduling algorithm of RC4," in *SAC*, 2001, vol. 2259, pp. 1-24. DOI: 10.1007/3-540-45537-X_1

[15] E. Tews, M. Beck, "Practical attacks against WEP and WPA," in *Proceedings of the second ACM conference on Wireless network security*, 2009, pp. 79-86. DOI: 10.1145/1514274.1514286

[16] E. Tews, R. P. Weinmann, A. Pyshkin, "Breaking 104 bit WEP in less than 60 seconds," *Information Security Applications*, pp. 188-202, 2007. DOI: 10.1007/978-3-540-77535-5_14

[17] N. Ferguson, B. Schneier, Practical cryptography, New York : Wiley, 2003.

[18] X. Peng, P. Zhang, H. Wei, B. Yu, “Known-plaintext attack on optical encryption based on double random phase keys,” Optics Letters, vol. 31, no. 8, pp. 1044-1046, 2006. DOI: 10.1364/OL.31.001044

[19] M. Matsui, A. Yamagishi,.”A new method for known plaintext attack of FEAL cipher,” in *Workshop on the Theory and Application of Cryptographic Techniques*, 1992, pp. 81-91. DOI: 10.1007/3-540-47555-9_7

[20] A. Tchernykh, U. Schwiegelsohn, E. G. Talbi, M. Babenko, “Towards understanding uncertainty in cloud computing with risks of confidentiality, integrity, and availability,” *Journal of Computational Science*, 2016. DOI: 10.1016/j.jocs.2016.11.011

[21] N. I. Chervyakov, P. A. Lyakhov, M. G. Babenko, A. I Garyanina,., I. N. Lavrinenko, A. V. Lavrinenko, M. A. Deryabin, “An efficient method of error correction in fault-tolerant modular neurocomputer,”. *Neurocomputing*, vol. 205, pp. 32-44, 2016. DOI: 10.1016/j.neucom.2016.03.041

[22] N. Chervyakov, M. Babenko, M. Deryabin, A. Garianina, “Development of Information Security's Theoretical Aspects in Cloud Technology with the Use of Threshold Structures,” in *EnT*, 2014, pp. 38-42. DOI: 10.1109/EnT.2014.19

[23] N. Chervyakov, M. Babenko, A. Tchenykh, I. Dvoryaninova, N. Kucherov, “Towards reliable low cost distributed storage in multi-clouds,” in *SIBCON*, 2017, pp. 1-6. DOI: 10.1109/SIBCON.2017.7998476

[24] V. Miranda-López, A. Tchernykh, J. M. Cortés-Mendoza, M. Babenko, G. Radchenko, S. Nesmachnow, Z. Du, “Experimental Analysis of Secret Sharing Schemes for Cloud Storage Based on RNS,” *Communications in Computer and Information Science*, 796, pp. 370-383, 2017. DOI: 10.1007/978-3-319-73353-1_26