Alabama Journal of Mathematics
41 (2017)
ISSN 2373-0404

Novel Cryptosystems Based on Two Different Mod Values

Zekeriya Y. Karatas
Department of Mathematics
Tuskegee University

Emin Aygün
Department of Mathematics
Erciyes University

Erkam Lüy
Department of Mathematics
Erciyes University

Huseyin Ergin
Department of Computer Science
University of Alabama

Bilal Gonen
School of Information Technology
University of Cincinnati

In this article, we construct two new cryptosystems which use two different mod values. Our main purpose is to create fast and secure schemes by using two different mod values. We do encryption according to mod N and decryption according to mod N_1 where N is a multiple of N_1. We examine homomorphic properties and security of these encryption schemes. At the final part, we implement the schemes and measure their time complexities.

Introduction

It is very well known that making any arbitrary operation on encrypted data is very important for privacy. This fact implies the concept of homomorphism in cryptosystems. The idea of privacy homomorphism was first introduced in 1978 by Rivest, Adleman and Dertouzos in Rivest, Adleman, and Dertouzos (1978). In 1982, S. Goldwasser and S. Micali constructed Goldwasser-Micali cryptosystem in Goldwasser and Micali (1982), and later a generalization of this system, Pailler cryptosystem, was given in Pailler (1999) in 1999. This cryptosystem is homomorphic with respect to addition. The famous RSA and El-Gamal cryptosystems are homomorphic with respect to multiplication, Silverberg (2013).

None of the cryptosystems mentioned above satisfy the feature of being homomorphic with respect to two operations. They are homomorphic with respect to a single operation, only addition or only multiplication. Up to 2009, it was not known whether a cryptosystem which is homomorphic to both addition and multiplication exists. We call such schemes Fully Homomorphic Encryption(FHE) Schemes.

In 2009, with a breakthrough result, Gentry introduced first FHE Scheme in Gentry (2009). After Gentry's paper, many cryptographers studied this system to improve it and make it more practical and more secure.

Additionally, we know that encryption schemes which are homomorphic with respect to addition have some applications in real life. Nowadays, homomorphic cryptosystems are very useful especially in electronic voting and cloud computing. So, we introduce a cryptosystem to contribute to electronic voting and cloud computing with a new idea with more security.

In this paper, we suggest two new symmetric key encryption schemes, one is multiplicative homomorphic, and the other is additional homomorphic. Our main purpose is to construct secure encryption schemes against plaintext - key recovery attack via using different mod values. We use two different mod values to deceive any attacks. We do encryption with respect to mod N and decryption according to mod N_1 with $N_1|N$. We purpose that even if attacker receive the secret key k, still more work should be done in order to break the system.

The rest of the paper is organized as follows. In Section 1, we present the details of Scheme 1. Section 2 depicts the details of Scheme 2. In Section 3, we show how we implement both schemes and measure their time complexities. Last section concludes the study.

Corresponding Author Email: zkaratas@mytu.tuskegee.edu

Scheme 1

The algorithm is as follows.
Keygen:

1. Choose $2r$ prime numbers p_i and q_i, where $1 \le i \le r$, such that for each i, p_i and q_i are distinct primes.

2. Compute $f_i = p_i q_i$, for $1 \le i \le r$.

3. Compute $N_1 = \text{lcm}(f_1, \ldots, f_r)$.

4. Compute $d = \Phi(N_1)$ where Φ denotes Euler Phi function.

5. Pick a secret key k such that $\gcd(k, d) = 1$.

6. Compute $N = k^2\left(\prod_{i=1}^{r} f_i\right)$.

7. Send public key $\{N\}$, and secret key $\{k, N_1\}$.

Encryption:

1. Take the output.

2. Determine your plaintext $M \in Z_{N_1}$.

3. Compute $C \equiv M^k \pmod{N}$.

4. Send C as the ciphertext of M.

Decryption:

1. Compute inverse l of k according to mod d.

2. Compute $C^l \pmod{N_1}$.

As it can be seen, our scheme has a public key and a private key which are shared. Hence our scheme is a symmetric crypotosystem.

Theorem 1. *The algorithm given above works.*

Proof. Since $\gcd(k, d) = 1$, we can find l such that $kl \equiv 1 \pmod{\Phi(N_1)}$. Hence, there exist an integer b such that $kl = 1 + b\Phi(N_1)$. By construction, $N_1 = w_1 w_2 \dots w_r$ where w_i's are distinct prime numbers. So, $\Phi(N_1) = \Phi(w_1)\Phi(w_2)\dots\Phi(w_r)$ since Euler Phi function is multiplicative.

Let w_i be one of primes in decomposition of N_1. Assume first that $\gcd(M, w_i) = 1$. Then, by Euler's Theorem $M^{\Phi(w_i)} \equiv 1 \pmod{w_i}$. Hence, by raising both sides of this congruence to the power $b\Phi(w_1)\Phi(w_2)\dots\Phi(w_{i-1})\Phi(w_{i+1})\dots\Phi(w_r)$ and then multiplying both sides by M yields $M^{1+b\Phi(w_1)\Phi(w_2)\dots\Phi(w_r)} \equiv M^{kl} \equiv M \pmod{w_i}$.

Next, suppose that $\gcd(M, w_i) = w_i$. Hence, we have $M \equiv 0 \pmod{w_i}$, which implies $M^{kl} \equiv M \equiv 0 \pmod{w_i}$. Thus, we again get $M^{kl} \equiv M \pmod{w_i}$.

Hence, for every i, we have $M^{kl} \equiv M \pmod{w_i}$. So, we have the following system of equations.

$$\begin{aligned} M^{kl} &\equiv M \pmod{w_1} \\ M^{kl} &\equiv M \pmod{w_2} \\ &\vdots \\ M^{kl} &\equiv M \pmod{w_r} \end{aligned}$$

By Chinese Remainder Theorem, this system has a unique solution $M^{kl} \equiv M \pmod{N_1}$. Now, note that $C \equiv M^k \pmod{N}$, $N_1 | N$ and $M \in Z_{N_1}$, hence $C \equiv M^k \pmod{N_1}$ as well. Thus, we get $C^l \equiv M^{kl} \equiv M \pmod{N_1}$ which implies that the algorithm works.

□

Homomorphic Property of Scheme 1

Theorem 2. *The Scheme 1 is homomorphic with respect to multiplication.*

Proof. Assume that $C_1 \equiv M_1^k \pmod{N}$ and $C_2 \equiv M_2^k \pmod{N}$, where $M_i \in Z_{N_1}$ for $i = 1, 2$. We clearly have $C_1 C_2 = M_1^k M_2^k = (M_1 M_2)^k \pmod{N}$. So, if we decrypt the ciphertext $C_1 C_2$ with $k^{-1} \equiv l \pmod{\Phi(N_1)}$ as in our algorithm, we obtain $M_1 M_2 \pmod{N_1}$ which shows that Scheme 1 is homomorphic with respect to multiplication and completes the proof.

□

Now, we can go over the security of Scheme 1.

Security of Scheme 1

(1) A Ciphertext Only Attack

There is no way to find out M from $C \pmod{N}$. Because C does not give any information about plaintext M since k and N_1 are private.

(2) A Known Plaintext Attack

Assume that the attacker knows one correct ciphertext and one correct corresponding plaintext, C_1 and M_1, respectively. Also we know that the attacker knows the mod value N because N is public. So, the attacker can construct the congruence

$$C_1 \equiv M_1^k \pmod{N} \tag{1}$$

The attacker wants to obtain the secret key k, hence, the attacker must solve the congruence (1). But this problem is a well-known problem which called the Discrete Logarithm Problem (DLP). However, solving DLP is computationally infeasible since N is composite, (see Section 3.8, Menezes, van Oorschot, and Vanstone (1996)).

However, we actually can increase the security with a new idea. In this method, even if the attacker solves the DLP and gets the secret key k, the attacker could not be able to obtain the inverse of it which is used in decryption algorithm.

To increase the security, we suggest to use two different mod values for misleading the attacker. While decrypting a ciphertext, we use l which is inverse of k with respect to mod $\Phi(N_1)$. So, we examine what happens if the attacker gets the secret key k and tries to compute its inverse with respect to mod $\Phi(N)$. Note here that the attacker does not know N_1, hence the attacker can not compute $\Phi(N_1)$.

Assume that attacker gets k and wants to compute its inverse. Hence the attacker must compute $\Phi(N)$. But for computing $\Phi(N)$, the attacker must factor N. This problem is one of the well-known difficult problems which is the Large Integer Factorization Problem. Assume next that the attacker factors N and computes $\Phi(N)$. So, now the aim for the attacker will be obtaining the inverse of k in mod $\Phi(N)$. But it is a well-known fact that k has an inverse in mod $\Phi(N)$ if and only if $\gcd(k, \Phi(N)) = 1$.

But we know that $N = k^2 \prod_{i=1}^{r} f_i$ where $f_i = p_i q_i$ with p_i and q_i are distinct primes for each i. So, $N = k^2 p_1 q_1 p_2 q_2 \dots p_r q_r$ and hence $\Phi(N) = \Phi(k^2 p_1 q_1 p_2 q_2 \dots p_r q_r)$. Assume that $k = h_1^{t_1} h_2^{t_2} \dots h_y^{t_y}$ where h_j's are distinct primes for $1 \le j \le y$.

Firstly, if $h_j \in \{p_1, q_1, p_2, q_2, \dots p_r, q_r\}$ for every j, then obviously $\gcd(k, \Phi(N)) > 1$.

Assume now that $h_{j_0} \notin \{p_1, q_1, p_2, q_2, \dots p_r, q_r\}$ for some j_0. Hence, $\Phi\left(h_{j_0}^{2t_{j_0}}\right) = \left(h_{j_0}^{2t_{j_0}-1}\right)(h_{j_0} - 1)$ divides $\Phi(N)$, hence we get $\gcd(k, \Phi(N)) > 1$ in this case as well.

So, the attacker will not be able to obtain the inverse of k by using N, which definitely misleads the attacker and increases the security.

In summary, the security is very strong since the attacker must solve two computationally infeasible problems DLP and Large Integer Factorization Problem, and guess the smaller mod value N_1.

(3) A Chosen Chiphertext Attack

Let C^* be the chosen chiphertext of the attacker, and M^* be the corresponding plaintext. Then the attacker must solve $(M^*)^k \equiv C^* \pmod{N}$. But this analysis is already given in part (2).

Before starting Scheme 2, we want to give a simple example of Scheme 1.

Example 1.

Keygen:

1. Let $r = 3$ and $p_1 = 2$, $p_2 = 3$, $p_3 = 5$, $q_1 = 3$, $q_2 = 5$, $q_3 = 7$.
2. Compute $f_1 = 6$, $f_2 = 15$ and $f_3 = 35$.
3. Compute $\mathrm{lcm}(6, 15, 35) = 210$.
4. Compute $\Phi(210) = 48$.
5. Let secret key be $k = 5$ since $\gcd(5, 48) = 1$.
6. Compute $N = 5^2 \prod_{i=1}^{3} f_i = 5^2 \times 6 \times 15 \times 35 = 78750$.
7. Output: Public Key $\{N = 78750\}$ and Secret Key $\{k = 5, 210\}$.

Encryption:

1. Take the output.
2. Let our message be $M = 20 \in Z_{210}$.
3. Compute ciphertext as $C \equiv 20^5 \equiv 50000 \pmod{78750}$.
4. Send $C = 50000$ as ciphertext of $M = 20$.

Decryption:

1. Compute inverse of k, $l = 29$ with respect to (mod 48).
2. Compute $M = 50000^{29} \equiv 20 \pmod{210}$.

Note that $\Phi(78750) = 18000$ and $\gcd(k = 5, \Phi(N) = 18000) = 5$ so k does not have inverse with respect to mod N. So the attacker can not obtain k^{-1} with respect to mod N and mod N_1.

Scheme 2

As we see in previous section, Scheme 1 is homomorphic with respect to multiplication. Now, we want to construct an additional homomorphic encryption scheme similar to Scheme 1 as additional homomorphic encryption schemes can be useful in electronic voting.

The algorithm for Scheme 2 is as follows.

Keygen:

1. Choose $2r$ prime numbers p_i and q_i, where $1 \le i \le r$, such that for each i, p_i and q_i are distinct primes.
2. Compute $f_i = p_i q_i$, for $1 \le i \le r$.
3. Compute $N_1 = \mathrm{lcm}(f_1, \dots, f_r)$.
4. Pick a secret key k such that $\gcd(k, N_1) = 1$.
5. Compute $N = k\left(\prod_{i=1}^{r} f_i\right)$.
6. Send public key $\{N\}$, and secret key $\{k, N_1\}$.

Encryption:

1. Take the output.
2. Determine your plaintext $M \in Z_{N_1}$.
3. Compute $C \equiv kM \pmod{N}$.
4. Send C as the ciphertext of M.

Decryption:

1. Compute inverse l of k with respect to mod N_1.

2. Compute $lC \equiv lkM \equiv M \pmod{N_1}$.

Theorem 3. *The algorithm given above works.*

Proof. It can be easily shown that this algorithm works by using the similar arguments in the proof of Theorem 1. □

Homomorphic Property of Scheme 2

Theorem 4. *The Scheme 2 is homomorphic with respect to addition.*

Proof. Assume that $C_1 \equiv kM_1 \pmod{N}$ and $C_2 \equiv kM_2 \pmod{N}$, then we have $C_1 + C_2 \equiv k(M_1 + M_2) \pmod{N}$. So, if we decrypt the ciphertext $C_1 + C_2$ with $k^{-1} \equiv l \pmod{N_1}$ as in our algorithm, we obtain $M_1 + M_2$, which completes the proof. □

Security of Scheme 2

Different from previous scheme, the attacker does not need to solve the DLP problem. Hence, we can conclude that the scheme is secure as well.

Time Complexity

In this section, we briefly give details of how we implement both schemes. We use Java programming language, as it provides native support for big integers which are especially designed for cryptograhic purposes (theoretically, a BigInteger in java has no size limit, meaning as long as the computer memory allows). The source code is publicly available online[1]. We choose 1024 bit numbers for p and q pairs in the schemes. In order to find very large prime numbers of length 1024, we use probable prime concept (see Kim and Pomerance (1989), Introduction) that is built into Java's BigInteger. We encrypt a very large plain text M, which is 10^{16}, and decrypt again for running and testing the system. Our test computer has Intel Core2Quad 2.83 GHz processor, 8 GB of RAM and Windows 7 running as the operating system.

We measure the individual steps of key generation, encryption and decryption of both schemes with respect to varying r. We repeat the experiment 5 times and compute the average in order to have more accurate results. All the measurements in the tables are in milliseconds. At the bottom row of each table, we also provide the complexity of each step with respect to r. In addition, we only show the measurable steps of schemes. Therefore, we do not show steps that only explain a fact or pass the produced data (e.g. in step 7 of Scheme 1, it only states what the public and secret keys are.).

In the following, we give the full table of measurements for both schemes and illustrate some important results according to the data we collect.

Scheme 1
Table 1 depicts the time measurements of Scheme 1 steps.

As a reference, $r = 1$ means we have one pair of primes $\{p, q\}$, the key generation takes 596 milliseconds. The ecnryption takes 16.4 milliseconds and the decryption takes 35.8 milliseconds.

Scheme 2
Table 2 depicts the time measurements of Scheme 2 steps.

As a reference, having r equals 1, the key generation takes 617.8 milliseconds. The ecnryption takes 0 milliseconds (which has some nanoseconds but is not shown due to precision) and the decryption takes 2.2 milliseconds.

The main difference between two schemes are in the encryption and decryption schemes. Scheme 1 is homomorphic with respect to multiplication and uses power operation in the encryption and decryption, whereas Scheme 2 is homomorphic with respect to addition and uses multiplication operation in the encryption and decryption. This difference shows itself in the complexities. Scheme 1 has an encryption complexity of $O(r^2)$. However, Scheme 2 has a constant complexity in the encryption. Same situation holds in the decryption. Scheme 1 has an overall complexity of $O(r^3)$. However, Scheme 2 has $O(r^2)$.

Conclusion

In this paper, we introduce two new symmetric key encryption schemes which use two different mod values. Each scheme is homomorphic with respect to a single operation. We also show that these schemes are strongly secure. We finalize by measuring the time complexities. They are very simple and useful for applications like electronic voting.

[1]`https://github.com/hergin/Encryption`

r	**Keygen Steps**						**Encrypt**	**Decrypt Steps**	
	1	**2**	**3**	**4**	**5**	**6**	**3**	**1**	**2**
1	450.8	0.2	0	0	144.8	0.2	16.4	2.4	33.4
2	495	0.2	0.4	0	254	0	61.6	3.2	242
4	1371.8	0	1.2	0.2	371.2	0.2	249	5.2	1863.2
8	2840.8	0	3	0.2	195.8	0.2	963	15.2	14625.4
16	7326	0.2	7.2	1	186.8	1.2	3778.2	51.2	114763.6
32	14452.8	0.2	20.8	4.4	335.6	4.8	14903.8	187.4	905927
64	28102.8	0.2	71.8	18.2	235	18.4	59645	733.2	7203246.4
128	58589.6	0.2	263.2	73.6	193	73	236544.4	2896.2	57730743.2
O()	**O(r)**	**O(r)**	**O(r^2)**	**O(r^2)**	**O(1)**	**O(r^2)**	**O(r^2)**	**O(r^2)**	**O(r^3)**

Table 1
Scheme 1 steps time measurements with complexity

r	**Keygen Steps**					**Encrypt**	**Decrypt Steps**	
	1	**2**	**3**	**4**	**5**	**3**	**1**	**2**
1	482.6	0	0	135.2	0	0	2	0.2
2	949.8	0	0.4	218.8	0	0	1.8	0
4	1650.8	0	1.2	183.8	0.2	0	4.4	0.2
8	3479.6	0	3	322.2	0.2	0	13.6	0.2
16	6619.2	0	7.4	226	1.2	0	46.2	0.2
32	13826	0.2	21	101.2	4.6	0	172.8	0.6
64	27941.2	0.2	72.6	200.8	18.2	0	668.4	0.8
128	52160.4	0.4	268	144.8	73.2	0	2659.6	2
O()	**O(r)**	**O(r)**	**O(r^2)**	**O(1)**	**O(r^2)**	**O(1)**	**O(r^2)**	**O(r)**

Table 2
Scheme 2 steps time measurements with complexity

References

Gentry, C. (2009). A fully homomorphic encryption scheme. *Ph.D. Thesis*. (Stanford University)

Goldwasser, S., & Micali, S. (1982). Probabilistic encryption and how to play mental poker keeping secret all partial information. *Proceedings of the 14th ACM Symposium on Theory of Computing*, 365–377.

Kim, S. H., & Pomerance, C. (1989). The probability that a random probable prime is composite. *Mathematics of Computation*, *53*(188), 721–741.

Menezes, A. J., van Oorschot, P. C., & Vanstone, S. A. (1996).

Menezes, A. J., van Oorschot, P. C., & Vanstone, S. A. (1996). *Handbook of Applied Cryptography*. CRC Press.

Pailler, P. (1999). Public-key cryptosystems based on composite degree residuosity classes. *Advances in Cryptology, EUROCRYPT*, 223–238.

Rivest, R. L., Adleman, L. A., & Dertouzos, M. (1978). On data banks and privacy homomorphisms. *Foundations of Secure Computation*, 169–179.

Silverberg, A. (2013). Fully homomorphic encryption for mathematicians. *sponsored by DARPA under agreement numbers FA8750-11-1-0248 and FA8750- 13-2-0054*.