An Efficient Homomorphic Encryption Protocol for Multi-User Systems

Liangliang Xiao
University of Texas at Dallas
xll052000@utdallas.edu

Osbert Bastani
Harvard University
obastani@fas.harvard.edu

I-Ling Yen
University of Texas at Dallas
ilyen@utdallas.edu

Abstract. The homomorphic encryption problem has been an open one for three decades. Recently, Gentry has proposed a full solution. Subsequent works have made improvements on it. However, the time complexities of these algorithms are still too high for practical use. For example, Gentry's homomorphic encryption scheme takes more than 900 seconds to add two 32 bit numbers, and more than 67000 seconds to multiply them. In this paper, we develop a non-circuit based symmetric-key homomorphic encryption scheme. It is proven that the security of our encryption scheme is equivalent to the large integer factorization problem, and it can withstand an attack with up to $m \ln \text{poly}(\lambda)$ chosen plaintexts for any predetermined m, where λ is the security parameter. Multiplication, encryption, and decryption are almost linear in $m\lambda$, and addition is linear in $m\lambda$. Performance analyses show that our algorithm runs multiplication in 108 milliseconds and addition in a tenth of a millisecond for $\lambda = 1024$ and $m = 16$.

We further consider practical multiple-user data-centric applications. Existing homomorphic encryption schemes only consider one master key. To allow multiple users to retrieve data from a server, all users need to have the same key. In this paper, we propose to transform the master encryption key into different user keys and develop a protocol to support correct and secure communication between the users and the server using different user keys. In order to prevent collusion between some user and the server to derive the master key, one or more key agents can be added to mediate the interaction.

Keywords: Homomorphic encryption, one-wayness security, large integer factorization, data centers, cloud.

1 Introduction

It has been a common practice for companies to outsource their online business logics to Web hosting service providers for over a decade. Generally, databases as well as the business logics of a company are hosted by a third party to save the IT management time and cost. The cloud computing further pushes forward this paradigm. There are many cloud based data centers which store a very large amount of data from different sources and support data-centric computation. Security can be a major concern for such data centers when the data they host are sensitive. A data center may be attacked and compromised. Also, there are the potential of insider attacks. If there is a change in management, such as reorganization or buyout [12], the potential threat increases due to the additional exposure to multiple management personnel and the unestablished policies regarding the handling of critical information in such situations.

The security problems with the outsourced databases can be solved if the critical data are encrypted. Naturally it leads to the problem of how the data center can perform computation on encrypted data [18]. Homomorphic encryption is a promising solution for this problem. Over the last three decades, the problem of homomorphic encryption has been studied extensively. The main works on homomorphic encryption algorithms are circuit based. They develop the encryption schemes and computation algorithms considering a single bit plaintext. All operations on various operand types can then be achieved by constructing the corresponding circuits. In [1], an encryption scheme based on elliptic curve is proposed, which allows computation on the ciphertexts directly if the computation involves at most one multiplication with any number of additions. In [19], a homomorphic encryption scheme has been constructed. This scheme doubles the ciphertext for each binary operation. Since the operations are bit-by-bit, the computation time and space overhead for arithmetic operations on general l-bit numbers (e.g., $l = 32$ or 64) can be excessive. In [8], Gentry designs a full homomorphic encryption scheme based on the mathematical object ideal lattices and uses the bootstrapping technique. It is semantically secure and the security of the scheme is based on the splitkey distinguishing problem. However the computational complexity of the scheme is $\tilde{O}(\lambda^6)$ for evaluating a gate over two bits, where $\boldsymbol{\lambda}$ is the security parameter and $\tilde{O}(g(x)) = O(g(x)\log^k g(x))$ for some k and. Dijk et al. conceptually simplify Gentry's construction by using a different hard problem, the approximate-GCD problem over integers [23]. [22] improves the security analysis, reducing the computational complexity of [8] from $\tilde{O}(\lambda^6)$ to $\tilde{O}(\lambda^3)$, and the computational complexity of [23] from $\tilde{O}(\lambda^{17})$ to $\tilde{O}(\lambda^{7.25})$. In [20], a different full homomorphic encryption scheme is constructed based on the elementary theory of algebraic number fields. However, all of these existing homomorphic encryption schemes have high time complexities [22].

Another approach to the homomorphic encryption problem is non-circuit based. The idea is to construct homomorphic encryption algorithms with plaintexts over a finite domain such as finite field. Compared to circuit-based approaches, this approach can be more efficient since it does not require additional circuit computation overhead. In [6], the homomorphic encryption algorithm called Polly Cracker is proposed. It encrypts a plaintext over a field by adding a random polynomial that vanishes under operations using the technique of Gröbner bases. Following Fellows and Koblitz's work, many non-circuit based homomorphic encryption algorithms using Gröbner bases have been proposed [13] [14] [16]; however, they have all either been broken [2] [5] [7] [11] [21] or lack conclusive security evidence. Hence, the problem of constructing a non-circuit based homomorphic encryption scheme remains open.

In this paper, we construct a non-circuit based encryption scheme that is homomorphic in both addition and multiplication. We downgrade the security requirement to achieve efficiency. Specifically, our encryption algorithm encrypts plaintexts over Z_N into the matrix ring $M_4(Z_N)$ by applying a similarity transformation by the key $k \in M_4(Z_N)$ to a diagonal matrix with two entries equal to the plaintext. Here N is the product of $2m$ prime numbers and is of size mn bits. Decryption is performed by applying the inverse similarity transformation. Addition and multiplication are matrix addition and matrix multiplication. Although the algorithm is not semantically secure, we have proven that when facing an adversary with up to $m \ln \mathrm{poly}(\lambda)$ chosen plaintext and ciphertext pairs, the security of our algorithm is equivalent to the large integer factorization problem. Here, m is any predetermined constant that is polynomial in the security parameter λ. Note that the security of the commonly used RSA encryption is no harder than the large integer factorization problem [17]. Thus, our homomorphic encryption scheme can be used in applications where semantic security is not required and one-wayness security is sufficient.

In our scheme, addition has time complexity $O(m\lambda)$. Multiplication, encryption, and decryption have time complexity $O(m\lambda \log m\lambda \log\log m\lambda)$. For relatively small m, multiplication and addition operations run very quickly. We implement our homomorphic encryption scheme and study its performance on a personal computer. When we choose m so that our scheme can withstand an attack with at least 1000 chosen plaintext and ciphertext pairs, the algorithm runs addition in only tenth of a millisecond and runs multiplication in 108 milliseconds. In contrast, Gentry's homomorphic encryption scheme [8] requires more than 900 seconds to add two 32-bit integers and more than 67000 seconds to multiply two 32-bit integers (based on the performance data given in [9]). As can be seen, our algorithm has real world applicability, especially when the large-scale plaintext attack is not an issue. Also, our algorithm is simple and elegant and can be easily implemented in practice.

Our encryption scheme is symmetric-key based while most of the existing homomorphic encryption schemes are public-key based. In public-key based schemes, the user can encrypt the data using the public key and send them to the data center for storage without needing to know the private key (the homomorphic encryption key). However, in almost all applications, it is necessary for the user to not only write to the data, but also retrieve the data from the data center. In order to decrypt the encrypted data retrieved from the data center, one solution is to let all the users have the private key. But this will severely weaken the security of the overall system. Another potential solution is to let different users use different keys, but it will prevent the data center from performing computations on the ciphertext encrypted using different keys. Currently, there is no solution to this problem in existing homomorphic encryption schemes.

We consider multi-user systems and propose a protocol to allow our symmetric-key homomorphic encryption scheme to be used in such systems. The data in the data center are encrypted using homomorphic encryption with a "master key" k. Different keys are assigned to different users. Let k_i denote the key held by user C$_i$. Correspondingly, the data center holds a matching key k'_i, where $k_i \cdot k'_i = k$. When sending a request to the data center, C$_i$ encrypts the secret data d in the request using key k_i, and sends $E(d, k_i)$ with the request to the data center. The data center transforms the encryption key from k_i to k by the similarity transform on the cipher: ${k'_i}^{-1} \cdot E(d, k_i) \cdot k'_i = E(d, k_i \cdot k'_i) = E(d, k)$ using the matching key k'_i. Similarly, when the data center sends a response to user C$_i$, the cipher $E(d, k)$ is transformed into $k'_i \cdot E(d, k) \cdot {k'_i}^{-1} = E\left(d, k_i \cdot {k'_i}^{-1}\right) = E(d, k_i)$ and sent with the response to C$_i$. C$_i$ decrypts it with the key k_i to obtain *d*. To avoid collusion of the user and the data center to reconstruct the master key, one or more agents in between C$_i$ and the data center using the same key transformation technique can be introduced to enhance system security.

The rest of the paper is organized as follows. In Section 2, we prove a few preliminary lemmas related to our algorithm and define the concept of the homomorphic encryption scheme. In Section 3 we construct the homomorphic encryption scheme, prove its correctness and security. In Section 4 we extend the encryption scheme to multi-user systems by considering multiple user keys and establishing the corresponding request/response communication protocol. In Section 5 we analyze the real world performance of our algorithm. Section 6 concludes this paper. All proofs in the paper are relegated to the Appendix due to space limitation.

2 Preliminaries

2.1 The Ring Z_N

In our homomorphic encryption scheme, all computations are performed in the ring of integers Z_N. Let λ denote the security parameter and let $\mathrm{poly}(\lambda)$ denote some fixed polynomial in λ. We construct N as follows. First, $2m$ prime numbers p_i and q_i of size $\lambda/2$ bits are chosen, where $m \ln \mathrm{poly}(\lambda)$ is the number of plaintext attacks which the algorithm can withstand. Then let $f_i = p_i q_i$ and $N = \prod_{i=1}^{m} f_i$. In the following, we show that such an N can be constructed in polynomial time. Specifically, we show that it is possible to find $2m$ prime numbers of $\lambda/2$ bits for some given m and λ. Note that m is required to be polynomial in λ to ensure that there are enough primes of length $\lambda/2$ bits.

Lemma 1: Given m and λ, where $m = O(\text{poly}(\lambda))$ (more precisely, $m \ll \sqrt{2}^{\lambda-4}$), it is possible to obtain $2m$ prime numbers of length $\lambda/2$ bits in polynomial time.

Theoretically, for $\lambda = 1024$, m is only bounded by $m \ll \sqrt{2}^{1024-4} = 2^{510}$. Therefore, for practical purposes, m can be chosen to be arbitrarily large.

Next, we show that factoring any of the f_i is infeasible if large integer factorization (factoring numbers of the form $f = pq$ for large primes p and q) is infeasible.

Lemma 2: Suppose that large integer factorization is infeasible. Let $f_i = p_i q_i$ for large primes p_i and q_i, where $1 \leq i \leq m$. Then given $F = \{f_i\}_{i=1}^{m}$, it is infeasible to factor any of the $f_i \in F$, i.e. there does not exist a PPT (probabilistic polynomial time) algorithm that can return a factor of f_i for any $1 \leq i \leq m$ with nonnegligible probability.

2.2 Matrices over Z_N

In the algorithm, we need to randomly choose an invertible matrix $k \in GL_4(Z_N)$. Lemma 3 demonstrates that an arbitrary matrix over Z_N is likely to be invertible, so that it is possible to choose k in polynomial time. For convenience, in the next lemma, we define $N = \prod_{i=1}^{m} p_i$, since our results do not depend on the prime factors of N coming in pairs.

Lemma 3: Let $N = \prod_{i=1}^{m} p_i$, where the p_i are prime and $m = O(\text{poly}(\lambda))$. A random matrix $T \in M_4(Z_N)$ is invertible with a high probability.

2.3 Definition of the Homomorphic Encryption Scheme

A homomorphic encryption scheme allows both addition and multiplication to be performed directly on encrypted data. Let (K, E, D) denote an encryption scheme, where E and D are the encryption and decryption algorithms and K is the corresponding key generation algorithm. In Definition 1, we define a homomorphic encryption scheme (K, E, D, F).

Definition 1: (K, E, D, F) is a homomorphic encryption scheme if K, E, and D form an encryption scheme (K, E, D) with E and D being the encryption and decryption algorithms, respectively, and K being the corresponding key generation algorithm. F is a polynomial time algorithm such that $K(F(E(o_1, k), \dots, E(o_l, k), f(x_1, \dots, x_l)),\ k) = f(o_1, \dots, o_l)$ for any plaintexts $o_1, \dots, o_l$ and any polynomial function $f(x_1, \dots, x_l)$. Here k is the encryption key in (K, E, D), $E(o_i, k)$ are the original ciphertexts, and $F\big(E(o_1, k), \dots, E(o_m, k),\ f(x_1, \dots, x_m)\big)$ is the computed ciphertext.

3 The Homomorphic Encryption Scheme

In this section, we present a fully homomorphic encryption scheme (K, E, D, F) , and prove that the algorithm is secure under the assumption that it is infeasible to factor large integers of the form $f = pq$ for large primes p and q. In Subsection 3.1 we introduce the idea of the design. In Subsection 3.2, we discuss the encryption, decryption, and key generation algorithms (K, E, D). Subsection 3.3 proves the security of (K, E, D) and Subsection 3.4 derives time complexity of the encryption and decryption schemes. Subsection 3.5 gives the computation algorithm F and proves the correctness of F. The complexity of F is derived in Subsection 3.6.

3.1 Design Concept

We first discuss the design idea of our homomorphic encryption scheme. We start from the Rabin's encryption algorithm. Given a plaintext x, the encryption algorithm is

$$E(x) = x^2 \bmod N$$

where $N = f = pq$. Although Rabin's encryption algorithm is homomorphic in multiplication, it is not homomorphic in addition. Thus, we can try to generalize the ciphertext domain from Z_N to ring of matrices over Z_N. In particular, consider the encryption algorithm

$$E_1(x) = \begin{pmatrix} x & 0 \\ 0 & r \end{pmatrix} \bmod N.$$

The homomorphic properties in addition and multiplication can be easily verified. However, since x is the eigenvalue of the eigenvector $\overrightarrow{v_{1,0}} = (1,0)^t$, the adversary can easily reverse x given the ciphertext by solving the linear equation system $E_1(x)\,\overrightarrow{v_{1,0}} = x\,\overrightarrow{v_{1,0}}$.

To cope with the problem above, we apply a randomly selected similarity transform k to $\begin{pmatrix} x & 0 \\ 0 & r \end{pmatrix}$, which is the ciphertext in E_1 and we call it the pre-transformed cipher from now. Note that the eigenvector corresponding to x is also transformed by k. As a result, the encryption algorithm becomes

$$E_2(x, k) = k^{-1} \begin{pmatrix} x & 0 \\ 0 & r \end{pmatrix} k \bmod N$$

where k is a randomly selected 2×2 invertible matrix. We give some informal reasoning as to why such an algorithm should be secure. Note that the similarity transformation transforms the eigenvector of x from $\overrightarrow{v_{1,0}}$ to $k^{-1}\,\overrightarrow{v_{1,0}}$. Since the adversary does not know the key k, he/she does not know the transformed eigenvector, so he/she cannot establish the linear equation system to obtain the plaintext. Also, although the adversary can derive the characteristic equation

$$\det\bigl(zI - E_2(x,k)\bigr) = 0 \bmod N \Leftrightarrow z^2 - (x + r)z + xr = 0 \bmod N,$$

it is infeasible for him/her to solve the quadratic equation, because this is equivalent to the factorization of N (the security of Rabin's encryption algorithm is also based on the infeasibility of solving quadratic equation in Z_N).

Unfortunately, the encryption algorithm E_2 cannot resist the chosen plaintext attack. Suppose that the adversary gets the plaintext and ciphertext pair

$$\left(x, E_2(x,k) = k^{-1} \begin{pmatrix} x & 0 \\ 0 & r \end{pmatrix} k\right)$$

Then the adversary can establish the equation $(E_2(x,k) - xI)\,\vec{v} = 0$, and derive the transformed eigenvector because

$$(E_2(x,k) - xI)\ \vec{v} = 0 \quad \Rightarrow k^{-1}\left(\begin{pmatrix} x & 0 \\ 0 & r \end{pmatrix} - xI\right) k\vec{v} = 0 \ \Rightarrow \begin{pmatrix} 0 & 0 \\ 0 & r - x \end{pmatrix} k\vec{v} = 0$$

$$\Rightarrow k\vec{v} = \overrightarrow{v_{1,0}} \quad \Rightarrow \vec{v} = k^{-1}\,\overrightarrow{v_{1,0}}.$$

Consequently, the adversary can solve for y, if given an additional ciphertext $E(y,k) = k^{-1} \begin{pmatrix} y & 0 \\ 0 & r \end{pmatrix} k \bmod N$, by solving the linear equation system $E(y,k)\,\vec{v} = y\,\vec{v}$.

One remedy is to use different keys to encrypt different plaintexts, but then the homomorphic properties in addition and multiplication is lost. Also, increasing the number of primes in N cannot improve security since the attack does not depend on the number of primes in N.

To improve the encryption algorithm so that it can withstand the chosen plaintext attack, we associate the eigenvalue x with two eigenvectors $\overrightarrow{v_1}$ and $\overrightarrow{v_2}$ instead of one. We choose the same $\overrightarrow{v_1}$ for all plaintexts so that the homomorphic properties in addition and multiplication are guaranteed. On the other hand, $\overrightarrow{v_2}$ cannot be the same for all plaintexts; otherwise, any linear combination of $\overrightarrow{v_1}$ and $\overrightarrow{v_2}$ is an eigenvector of all plaintexts, becoming the same as E_2. Instead, we randomly choose $\overrightarrow{v_2}$ following a probability distribution D (to be given). Note that since x is associated with two eigenvectors and there are choices in $\overrightarrow{v_2}$, we need to work in matrices of a higher dimension. For example, we can have $\overrightarrow{v_1} = (1,0,0,0)$ and $\overrightarrow{v_2}$ can be randomly chosen between $(0,1,0,0)^t$ and $(0,0,1,0)^t$ following D.

When encrypting a plaintext, we construct the pre-transformed ciphertext C with $\overrightarrow{v_1}$ and $\overrightarrow{v_2}$ as eigenvectors of C corresponding to eigenvalue x. Then we perform a similarity transformation with key k. Let E_3 denote this encryption scheme.

Now, the adversary has to derive the transformed eigenvector $k^{-1}\overrightarrow{v_1}$ in order to compromise the encryption scheme. Suppose that the adversary gets a single plaintext and ciphertext pair. Then he/she cannot derive $k^{-1}\overrightarrow{v_1}$ because any linear combination of $k^{-1}\overrightarrow{v_1}$ and $k^{-1}\overrightarrow{v_2}$ is an eigenvector of C with eigenvalue x.

Next consider the case where the adversary gets two pairs of plaintexts and ciphertexts. If $\overrightarrow{v_2} = \overrightarrow{v_1}$, then, same as above, the adversary cannot derive $\overrightarrow{v_1}$. If $\overrightarrow{v_2} \neq \overrightarrow{v_1}$, then, the probability of success with which the adversary can derive $k^{-1}\overrightarrow{v_1}$ after an attack with m' chosen plaintexts follows a probabilistic distributions related to D. Without loss of generality, let $p_{m'}$ denote the probability for the adversary to derive $\overrightarrow{v_1}$ after an attack with m' chosen plaintext and ciphertext pairs. To improve the strength of encryption, we want to apply E_3 multiple times. More precisely, let $N = \prod_{i=1}^{m} f_i$, where $f_i = p_i q_i$. For each i, we apply E_3 to encrypt the plaintext x over Z_{f_i}. Then, the ciphertext over Z_N is obtained by applying the Chinese remainder theorem to the individual encryption results over Z_{f_i}, for all i. Let E_4 denote this new encryption scheme.

Now, the adversary has to derive the corresponding eigenvectors over all Z_{f_i} in order to reverse the plaintext over Z_N. The probability for the adversary to derive all corresponding eigenvectors over all Z_{f_i} after an attack with m' chosen plaintexts decreases to $p_{m'}^{m}$. Hence, by carefully selecting parameters D and m, E_4 can resist the chosen plaintext attack with a predetermined number m' of plaintexts.

The discussion above is to give a high level concept of our design of a non-circuit based homomorphic encryption scheme. We will formally prove the security of the encryption scheme by reduction to the large integer factorization problem. In particular we will show that if there exists a PPT algorithm that can reverse the ciphertext with nonnegligible probability after the chosen plaintext attack with m' plaintexts, then there exists a PPT algorithm to factor N.

3.2 Encryption and Decryption Algorithms

To formally set up the encryption algorithm following the concept for constructing E_4, we first pick a random 4×4 invertible matrix k as the key for our encryption scheme. This can be done efficiently as proven in Lemma 3. We then construct the diagonal matrix $\mathrm{diag}(x, a, b, c)$, where a, b, and c, the solutions to a sets of linear congruences depending on x and a random value $r \in Z_N$, are computed using the Chinese Remainder Theorem. The corresponding ciphertext C is the similarity transform of this matrix by k, $C = k^{-1}\mathrm{diag}(x, a, b, c)k$. The encryption algorithm is formally presented as follows.

Given $\{f_i\}_{i=1}^{m}$ with $N = \prod_{i=1}^{m} f_i$, and a plaintext $x \in Z_N$, we encrypt x into a ciphertext $C \in M_4(Z_N)$ as follows:

1. Choose a random value $r \in Z_N$.
2. Define a set of numbers a_i , b_i, and c_i, for $1 \leq i \leq m$, as follows. For each i, exactly one of a_i , b_i and c_i, is equal to x. Let $a_i = x$ with probability $1 - \frac{1}{m+1}$, $b_i = x$ with probability $\frac{1}{2(m+1)}$, and $c_i = x$ with probability $\frac{1}{2(m+1)}$. Set the other two values equal to r. This way, for each i, one of the values a , b, and c, equals x and the other two equal r.
3. By the Chinese Remainder Theorem, let a, b, c, be the solution to the set of simultaneous congruences $a = a_i \bmod f_i$, $b = b_i \bmod f_i$, and $c = c_i \bmod f_i$, for $1 \leq i \leq m$.
4. Let $C = k^{-1}\mathrm{diag}(x, a, b, c,)k$.

Given ciphertext C and key k, the decryption algorithm compute the plaintext $x = (kCk^{-1})_{00}$.
The correctness of the encryption scheme is proven in Lemma 4.

Lemma 4: The encryption scheme (K, E, D) is correct.

The following is an example of our encryption and decryption algorithms. Let $m = 2$, and consider the values $f_1 = 3 \times 5 = 15$ and $f_2 = 2 \times 7 = 14$, so that $N = 210$. We randomly choose key

$$k = \begin{pmatrix} 17 & 44 & 169 & 126 \\ 91 & 121 & 84 & 85 \\ 85 & 71 & 119 & 25 \\ 0 & 85 & 201 & 44 \end{pmatrix} \text{ with inverse } k^{-1} = \begin{pmatrix} 35 & 31 & 29 & 0 \\ 44 & 57 & 113 & 29 \\ 74 & 157 & 37 & 194 \\ 59 & 27 & 152 & 103 \end{pmatrix}.$$

We encrypt the plaintext $42 \in Z_{210}$ using our encryption algorithm. First we randomly choose $r = 91 \in Z_{210}$. Next, we let $a_1 = x = 42$ and $b_1 = c_1 = r = 91$, and let $a_2 = c_2 = r = 91$ and $b_2 = x = 42$. Then we use the Chinese Remainder Theorem to solve for the values a , b , and c where $a = 42 \bmod 15$ and $a = 91 \bmod 14$, and where $b = 91 \bmod 15$ and $b = 42 \bmod 14$, and where $c = 91 \bmod 15$ and $c = 91 \bmod 14$, and get $a = 147$, $b = 196$, and $c = 91$. Finally we construct our ciphertext,

$$C = k^{-1}\mathrm{diag}(42, 147, 196, 91)k$$
$$= \begin{pmatrix} 35 & 31 & 29 & 0 \\ 44 & 57 & 113 & 29 \\ 74 & 157 & 37 & 194 \\ 59 & 27 & 152 & 103 \end{pmatrix} \cdot \begin{pmatrix} 42 & 0 & 0 & 0 \\ 0 & 147 & 0 & 0 \\ 0 & 0 & 196 & 0 \\ 0 & 0 & 0 & 91 \end{pmatrix} \cdot \begin{pmatrix} 17 & 44 & 169 & 126 \\ 91 & 121 & 84 & 85 \\ 85 & 71 & 119 & 25 \\ 0 & 85 & 201 & 44 \end{pmatrix} = \begin{pmatrix} 77 & 91 & 154 & 35 \\ 35 & 84 & 49 & 189 \\ 175 & 133 & 140 & 119 \\ 35 & 98 & 49 & 175 \end{pmatrix}$$

Then we decrypt our ciphertext and extract the plaintext x,

$$x = (kCk^{-1})_{00}$$
$$= \left(\begin{pmatrix} 17 & 44 & 169 & 126 \\ 91 & 121 & 84 & 85 \\ 85 & 71 & 119 & 25 \\ 0 & 85 & 201 & 44 \end{pmatrix} \cdot \begin{pmatrix} 77 & 91 & 154 & 35 \\ 35 & 84 & 49 & 189 \\ 175 & 133 & 140 & 119 \\ 35 & 98 & 49 & 175 \end{pmatrix} \cdot \begin{pmatrix} 35 & 31 & 29 & 0 \\ 44 & 57 & 113 & 29 \\ 74 & 157 & 37 & 194 \\ 59 & 27 & 152 & 103 \end{pmatrix} \right)_{00}$$
$$= \begin{pmatrix} 42 & 0 & 0 & 0 \\ 0 & 147 & 0 & 0 \\ 0 & 0 & 196 & 0 \\ 0 & 0 & 0 & 91 \end{pmatrix}_{00} = 42.$$

3.3 Security of the Encryption Scheme

We proceed with the proof of security by reductions. In Lemma 6 we establish the existence of matrices k_i which will be used in the proof of security because, as will be shown in Lemma 7, given the ciphertext C, an adversary cannot distinguish between $E(x, k)$ and $E(y, k_i k)$. Lemma 8 demonstrates that this fact implies that, if a PPT algorithm exists to extract the plaintext x, with nonnegligible probability, from the ciphertext C without the key k, then a PPT algorithm exists to factor $f_i \in F$, for some arbitrarily chosen $F = \{f_i\}_{i=1}^{m}$, for some i, with nonnegligible probability. Then, by Lemma 2, there would exist a PPT algorithm for large integer factorization with nonnegligible probability. Lemmas 9 and 10 prove that, given $m' \leq m$ plaintext and ciphertext pairs $\{(x_l, C_l)\}_{l=1}^{m'}$, if a PPT algorithm exists to find the plaintext x given the corresponding ciphertext C with nonnegligible probability, then a PPT algorithm exists to factor $f_i \in F = \{f_i\}_{i=1}^{m'}$, implying that there exists a PPT algorithm for large integer factorization with nonnegligible probability. Finally, Theorem 1 proves the same result with the weaker condition $m' \leq m \ln \mathrm{poly}(\lambda)$. In other words, if given less than $m \ln \mathrm{poly}(\lambda)$ plaintext ciphertext pairs, then decryption of a ciphertext without the key is at least as hard as the well known integer factorization problem.

Lemma 5: For $1 \leq i \leq N$, there exists a unique element $k_i \in GL_4(Z_N)$ so that $k_i = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \bmod p_i$, $k_i = I \bmod q_i$, and $k_i = I \bmod f_j$ for $j \neq i$ where I is the identity matrix in $GL_4(Z_N)$. Additionally, $k_i = k_i^{-1}$.

Lemma 6: Given plaintext x, key k and random element r, there exist y and s such that $E(x, k) = E(y, k_i k)$. Additionally, $y - x$ divides q_i, and $y - x$ does not divide p_i with probability $\frac{1}{m+1}\left(1 - \frac{1}{p_i}\right)$.

Lemma 7: If a PPT algorithm $A_d(C)$ exists that, given $C = E(x, k)$, returns x with probability p, then there exists a PPT algorithm A_f to return a factor f_i for some i with probability $p' = \frac{p}{m+1}\left(1 - \frac{1}{p_i}\right)$.

Lemma 8: Let m' be the number of plaintext and ciphertext pairs the adversary has access to. If for some m' there exists an algorithm $A_d\left(C = E(x,k),\ \{(x_l, C_l = E(x_l,k))\}_{l=1}^{m'}\right)$ such that, given m' chosen plaintext and ciphertext pairs (x_l, C_l) and a ciphertext C, returns x with some probability p, then there exists a PPT algorithm A_f using A_d as an oracle to factor f_i for some i with probability

$$p' = p\left(1 - \frac{1}{p_i}\right)\left(1 - \left(1 - \frac{1}{m+1}\left(1 - \frac{1}{m+1}\right)^{m'}\right)^{m}\right).$$

Lemma 9: Assuming that the probability to factor an λ bit integer in polynomial time is negligible, the encryption scheme is secure for $m' \le m$.

We now prove the security of our homomorphic encryption algorithm.

Theorem 1: The bound of m' in Lemma 10 can be weakened to $m' \le m \ln \mathrm{poly}(\lambda)$, where $\mathrm{poly}(\lambda)$ denotes some fixed polynomial in λ.

We have shown that, under an attack with $m \ln \mathrm{poly}(\lambda)$ chosen plaintext and ciphertext pairs, our encryption scheme reduces to the large integer factorization problem, under an attack with $m \ln \mathrm{poly}(\lambda)$ chosen plaintext and ciphertext pairs. Note that there is no constraint on the chosen plaintexts. In particular, the adversary can choose a plaintext multiple times.

3.4 Complexity of the Encryption and Decryption Algorithms

We need to choose $2m$ primes in the encryption scheme. As shown in Lemma 1 this takes polynomial time. Note that the primes can be precomputed. The decryption algorithm involves only two matrix multiplications, which, as shown later in Section 3.6, takes $O(m\lambda \log m\lambda \log\log m\lambda)$ time. The encryption algorithm requires both two matrix multiplications and also an algorithm to solve the m linear congruences that define the values a, b, and c. It takes time $O(m\lambda)$ to construct the solution to these linear congruences, so the overall complexity for encryption is also $O(m\lambda \log m\lambda \log\log m\lambda)$.

3.5 Computation Algorithms

Multiplication and addition of encrypted elements is simply normal matrix multiplication and addition, respectively.

Lemma 10: The multiplication and addition algorithms are correct.

3.6 Complexity of the Computation Algorithms

We now consider the complexity of our multiplication and addition algorithms. First consider the size of the integers in the ring Z_N. The value N is the product of m numbers of length λ bits, so it is approximately an $m\lambda$ bit number. There exist efficient algorithms for multiplication of b bit integers with complexity $O(b \log b \log\log b)$. For $b = m\lambda$ this becomes $O(m\lambda \log m\lambda \log\log m\lambda)$. In our case matrix multiplication involves 64 multiplications and 64 additions. Since addition can be done in linear time, the algorithm is dominated by multiplication and thus has complexity $O(m\lambda \log m\lambda \log\log m\lambda)$. Addition is linear and thus has complexity $O(m\lambda)$.

4 Homomorphic Encryption in Multi-user Systems

The homomorphic encryption scheme discussed in Section 3 cannot be securely used in practical systems. To allow computation on encrypted data, the data in the data center should be encrypted by the same master key. The master key then has to be shared by multiple users who need to access the data in the data center. A user may need to encrypt secret data and send them with a request to the data center. Also, the data center may send a response, which contains some encrypted data, to a user and the user needs to decrypt the ciphertexts in the response. Having all users holding the master key can compromise the security of the system, especially if the users are from many different domains.

Our solution is let each user hold a unique user key k_i and use a transformation function to transform the ciphertext encrypted by the user key k_i to the ciphertext encrypted by the master key k. Such a transformation scheme may not always be easy to obtain for some encryption algorithms. In our scheme, data are encrypted into matrix representation and a similarity transform function can be used to achieve the goal. We develop the corresponding communication protocol for sending secret data between the users and the data center using individual user keys and then use similarity transform to convert the user keys to the master encryption key. In Subsection 4.1, we define a model for the multi-user systems. In Subsection 4.2, we present the protocols for the user to send requests and receive responses while using her own user key to encrypt and decrypted the data in the request and the response, respectively.

4.1 System Model

We consider a simplified system model where the data center consists of a single server hosting a database and a set of users accessing the data in the database. For security assurance, a key agent is added in between the server and the users. Let DB denote the database server, KA denote the key agent, and $C = \{C_i \mid i \geq 1\}$ denote the set of users. User C_i holds a user key k_i, KA holds the first matching key of k_i, denoted as $k_i{}'$. DB holds the second matching key of k_i, denoted as $k_i{}''$, where $k = k_i \cdot k_i' \cdot k_i''$. The keys are generated and distributed by a trusted party TP at the system initialization time.

Each entity DB, KA, or a user may be malicious or may be compromised by an external attacker (we will refer to this as malicious in general) and will act like a passive attacker to compromise some critical information. One or more users may collude with DB or KA to acquire critical information. Since DB and KA are generally better protected, the probability that both of them are malicious is extremely small. Thus we assume that it is not possible to have malicious DB, KA, and at least a user to collude. Specifically, we assume the adversary structure

$$AS = \{\{DB\}, \{KA\}, C_A, \{DB, KA\}, \{DB\}\cup C_A, \{KA\}\cup C_A \mid C_A \subset C\}.$$

We also assume that an adversary can access all the information its role allows.

The data hosted by the DB may have different criticality levels and may be protected in different ways. We only consider the data that should be protected during computation time and encrypt them using our homomorphic encryption scheme. Additional protection can be added by encrypting these data using a conventional encryption scheme, such as AES, when they are in memory or disk and decrypt them in CPU. Also, we assume that the communications between any two entities are via secure channels (e.g. messages are properly encrypted and communication keys are properly established). The adversary cannot know the content of the communication unless it compromises at least one entity. (In our protocol, we do not discuss the additional protection mechanisms but only consider the steps relevant to our homomorphic encryption.)

4.2 The Multi-User Access Protocol

Here, we first introduce the similarity transformation function, which plays the central role in the construction of the protocols, and discuss its property. Let φ be the similarity transformation function

$$\varphi(C, k') = k'^{-1} \cdot C \cdot k'$$

where $C = E(x, k)$ is the ciphertext, and k, k' are encryption keys. Then φ can transform the encryption key from k to $k \cdot k'$ based on the following lemma.

Lemma 11: If $C = E(x, k)$, then $\varphi(C, k') = E(x, k \cdot k')$.

The process of the system consists of two phases: the system initialization phase, and the request-response phase. At the system initialization time, a trusted party TP generates and distributes the keys to the DB, KA, and users, then TP exits the system and destroys all the key related knowledge. In the request-response phase, the user C_i sends a request to DB, and DB processes it, and sends a response to C_i.

Key generation and distribution. TP generates the master key k by the method discussed in Section 3.2. Then TP generates many key triples k_i, k'_i, k''_i . It uses the same method (Section 3.2) to randomly generate user key k_i and the first matching key k'_i. Then, it computes the second matching key $k''_i = k'^{-1}_i \cdot k_i^{-1} \cdot k$. TP sends k_i to user C$_i$, k'_i to KA, and k''_i to DB. In a static system, TP exits the system after key initialization and distribution. In a dynamic system where new users may join the system dynamically, a key manager KM is also introduced to manage user keys. TP sends the list of unused user keys to KM to be distributed to new users later. TP exits after key initialization and distribution. Note that matching keys can be associated using their indices.

Request-response protocol. The main issue in the request-response protocol is how to encrypt the data to be sent with the request and how to decrypt the data in the response. The pseudo code for the request-response protocol is given in Figure 1. In the protocol, the critical data in the request is encrypted (Line 1), and the encryption key is then transformed (Lines 2 and 3) by the KA and DB, respectively. Then, in Line 4, the DB processes the request and generates the response (with an encrypted data in the response). The encryption key of the critical data in the response is transformed (Lines 5 and 6) by the DB and KA, respectively. Finally, the user decrypts and gets the result (Line 7).

1. *C$_i$* prepares a request q with a sensitive data d. It encrypts d with k_i, obtaining $E(d, k_i)$, and sends q with $E(d, k_i)$ to KA.
2. KA computes $\varphi(E(d, k_i), k'_i) = E(d, k_i \cdot k'_i)$ and sends the updated request q to DB.
3. DB computes $\varphi(E(d, k_i \cdot k'_i), k''_i) = E(d, k_i \cdot k'_i \cdot k''_i) = E(d, k)$.
4. DB processes q with $E(d, k)$ and generates response *r* with an encrypted data $E(d', k)$.
5. DB computes $\varphi\left(E(d', k), k''^{-1}_i\right) = E(d', k_i \cdot k'_i)$ and sends $E(d', k_i \cdot k_i')$ with *r* to KA.
6. KA further computes $\varphi\left(E(d', k_i \cdot k_i'), k'^{-1}_i\right) = E(d', k_i)$ and sends $E(d', k_i)$ with *r* to C$_i$.
7. *C$_i$* receives the response and decrypts $E(d', k_i)$ with k_i and gets d'.

Figure 1. Request processing protocol.

Theorem 2: Suppose that the adversary attacks the multi-user system with the adversary structure AS. The system is secure if the adversary collects less than $m \ln \text{poly}(\lambda)$ plaintext-ciphertext pairs.

5 Performance of Our Homomorphic Encryption Scheme

We implement our algorithm and evaluate its execution time. The large integer multiplication and addition were implemented using the GNU Multiple Precision (GMP) Arithmetic Library [10]. In Figure 2, we give the number of milliseconds required to perform addition and multiplication of encrypted data (4×4 matrices over the ring Z_N). The computations were performed on a 2.16GHz Intel Core 2 Duo Processor. The security parameter considered was $\lambda = 1024$. The data was gathered from running 10000 additions and multiplications of randomly selected numbers of length $m\lambda$ bits.

As can be seen from Figure 2, for a small enough m, the algorithm is very efficient. For example, for $\lambda = 1024$ and $m = 16$, the algorithm runs multiplication in only 108 milliseconds and runs addition in a tenth of a millisecond. For such λ and m, we can choose $\text{poly}(\lambda) = \lambda^{10} = 2^{100}$, which translates into 1109 chosen plaintexts in an attack that our algorithm can withstand. This makes the algorithm practical for real world implementation where large scale plaintext attacks are not an issue.

For the purpose of comparison, we estimate the computation time of Gentry's homomorphic encryption scheme [8]. In [9], the performance of the primitive operations has been studied: The bootstrapping (re-crypt) time is 6 seconds, which dominates the time of the operations. For *l*-bit numbers, the addition circuit needs 5**l* gates and the multiplication circuit needs 11*l^2 gates [15]. Thus, Gentry's homomorphic encryption scheme needs 5*32*6 (**> 900**) seconds to add two 32 bit numbers, and 11*32^2*6 (**> 67000**) seconds to multiply two 32 bit numbers. This is far slower than our homomorphic encryption scheme (0.1 milliseconds to add two 32 bit numbers and 108 milliseconds to multiply two 32 bit numbers).

We also study the performance of our request and response protocols for multi-user data processing systems. Assume that the size of the secret data in the request and response is the same as the security parameter. To factor in the communication cost, we simulate the case where the user is at UTD, the KA is at ASU, and the DB is at UCLA. The performance results for $\lambda = 1024$ and $m = 16$ are shown in Table 1 and the execution time are measured in milliseconds.

Operation	**Process Description**	**Performance**
Encryption	Encrypt one data item	**215** ms
Decryption	Decrypt one data item	**34** ms
Transform	Local one-time transformation	**215** ms
The user sends a request to DB	User encrypts the data and sends the request to KA, KA transforms the embedded data and sends the request to DB, DB further transforms the data in the request.	**807** ms (**85** ms if the data in the request is not encrypted and is sent from user to DB directly)
DB sends a response to the user	DB transforms the data in the response and sends the response to KA, KA transforms the data and sends the response to the user, the user decrypts the data	**806** ms (**77** ms if the data in the response is not encrypted and is sent from DB to user directly)

Table 1. The performance of the communication protocol

As can be seen, by using the communication protocol with our homomorphic encryption algorithm, the performance for sending a request and receiving a response is degraded by approximately 10 times from the case where encryption is not used, but it is still a reasonable cost to achieve the desired security.

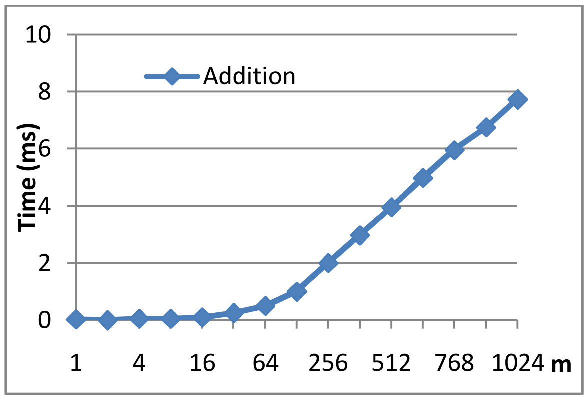
6 Conclusion

We have presented a novel non-circuit based homomorphic encryption algorithm. Our scheme is fully homomorphic, but it is not semantic secure. The security of our homomorphic encryption scheme is equivalent to the well known large integer factorization problem (which is also the security basis for RSA), but it requires a chosen bound on the plaintext attack. Even though Gentry's and the subsequent solutions are semantic secure, their time complexity is too high for practical use. Also, its circuit based approach suffers from a significant overhead. Our scheme yields a very practical time complexity for encryption, decryption, and computation on ciphertexts. Specifically, to withstand a chosen plaintext attack with over 1000 plaintexts, the algorithm runs addition in only tenth of a millisecond and multiplication in hundred milliseconds. In contrast, Gentry's algorithm requires more than 900 seconds for addition and more than 67000 seconds for multiplication.

Due to the realistic performance and the simplicity of our algorithm, it can be used in real world applications with moderate performance requirements and where the large-scale plaintext attack is unlikely. For example, in large scale key management systems, such as some SCADA [3] systems, it is necessary to use many distributed key servers to avoid overloading. In this case, some key servers may not be fully trusted or may be compromised by internal or external attackers. Untrustable key servers pose very severe consequences. Our homomorphic encryption algorithm can be used to confidentially generate, validate, and update communication and authentication keys without knowing the actual information of these keys. Also, in mobile agent based systems, the agents may use homomorphic encryption algorithms to protect their data and securely perform transactions on a remote host.

Our homomorphic encryption algorithm is symmetric-key based while most of the existing algorithms are public key based. The only advantage of the public key homomorphic encryption schemes is the possibility of encrypting data without needing to know the private key, i.e., so that many clients can issue the requests to the encrypted database. However, in almost all applications, it is necessary, but not secure, for the client to know the private key in order to read back and decrypt the data in the response. Our request-response communication protocol for our symmetric-key homomorphic encryption scheme can secure the request and response processes in multi-user systems.

Our novel approach to secure computation, encrypting data into matrices over finite integer rings with eigenvalue equal to the plaintext, will potentially lead to even more efficient homomorphic encryption algorithms. We plan on continuing to use this approach to improve the efficiency of our scheme, especially for systems requiring protection against the chosen plaintext attack with a large number of plaintext and ciphertext pairs, and to investigate possible further improvements in the security of our approach.



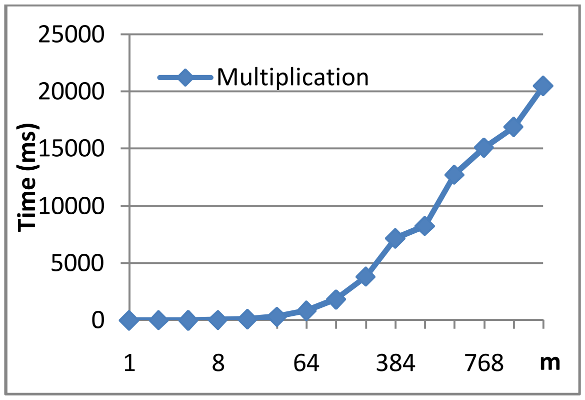


Figure 2. The size of m against the speed of addition and multiplication of two encrypted data (in milliseconds).

7 Bibliography

[1] G. Bebek. “Anti-tamper database research: Inference control techniques”, *Technical Report EECS 433 Final Report*, Case Western Reserve University, 2002.

[2] S. Bulygin, T. Rai, “Countering Chosen-Ciphertext Attacks against Noncommutative Polly Cracker Cryptosystems”, *Special Semester on Gröbner Bases*, Linz, Austria, May 2006.

[3] D. Choi, H. Kim, D. Won, and S. Kim, “Advanced Key Management Architecture for Secure SCADA Communications”, *IEEE Transactions on Power Delivery*, Vol. 24, No. 3, pp. 1154-1163, 2009.

[4] R. Dingledine; N. Mathewson, P. Syverson. "Tor: The Second-Generation Onion Router", *Proceedings of the 13th USENIX Security Symposium*, August 2004.

[5] R. Endsuleit, W. Geiselmann, R. Steinwandt, “Attacking a Polynomial-Based Cryptosystem: Polly Cracker”, *International Journal of Information Security*, Vol. 1, No. 3, pp. 143-148, 2002.

[6] M. Fellows and N. Koblitz, “Combinatorial Cryptosystems Galore!”, *Finite fields: theory, applications, and algorithms*, volume 168 of Contemp. Math., pp 51-61, 1994.

[7] W. Geiselmann, R. Steinwandt, “Cryptanalysis of Polly Cracker”, *IEEE Transactions on Information Theory 48(11)*, pp. 2990-2991, 2002.

[8] C. Gentry, “Fully homomorphic encryption using ideal lattices”, *41st ACM Symposium on Theory of Computing (STOC)*, 2009.

[9] C. Gentry, “A working implementation of fully homomorphic encryption”, http://eurocrypt2010rump.cr.yp.to/9854ad3cab48983f7c2c5a2258e27717.pdf.

[10] GMU MP library, http://gmplib.org/.

[11] D. Hofheinz, R. Steinwandt, “A Differential Attack on Polly Cracker”, *Proceedings of IEEE International Symposium on Information Theory*, pp. 211, 2002.

[12] M. Kane and D. Kawamoto, “Oracle buys PeopleSoft for $10 billion”, http://news.cnet.com/Oracle-buys-PeopleSoft-for-10-billion/2100-1001_3-5488298.html.

[13] F. Levy-dit-Vehel, L. Perret, “A Polly Cracker System Based on Satisfiability”, *Progress in Computer Science and Applied Logic*, pp 177-192, 2004.

[14] L. Ly, “Polly Two – A New Algebraic Polynomial-Based Public-Key Scheme”, *Applicable Algebra in Engineering, Communication and Computing*, 17, pp. 267-283, 2006.

[15] M. Morris Mano, Digital Design, Prentice Hall; 3 edition, August 1, 2001.

[16] T. Rai, “Infinite Gröbner bases and Noncommutative Polly Cracker Cryptosystems”, *PhD Thesis*, Virginia Polytechnique Institute and State Univ, 2004.

[17] R. Rivest, A. Shamir, L. Adleman, “A Method for Obtaining Digital Signatures and Public-Key Cryptosystems”, *Communications of the ACM 21 (2)*, pp. 120–126, 1978.

[18] R.L. Rivest, L. Adleman and M.L. Dertouzos, “On data banks and privacy homomorphisms”, *Foundations of Secure Computation*, eds. R. A. Demillo et al., Academic Press, 1978, pp. 167-179.

[19] T. Sander, A. Young, M. Yung, “Non-Interactive CryptoComputing For NC1”, *FOCS* 1999, pp. 554-567.

[20] N. P. Smart and F. Vercauteren, “Fully homomorphic encryption with relatively small key and ciphertext sizes”, Accepted to PKC 2010. Available at http://eprint.iacr.org/2009/571.

[21] R. Steinwandt, “A Ciphertext-Only Attack on Polly Two”, preprint, 2006.

[22] D. Stehle and R. Steinfeld, “Faster Fully Homomorphic Encryption”, *Cryptology ePrint Archive: Report 2010/299*, 2010.

[23] M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, “Fully homomorphic encryption over the integers”, Accepted to Eurocrypt 2010. Available at http://eprint.iacr.org/2009/616.

8 Appendix

Lemma 1: Given m and λ, where $m = O(\text{poly}(\lambda))$ (more precisely, $m \ll \sqrt{2}^{\lambda-4}$), it is possible to obtain $2m$ prime numbers of length $\lambda/2$ bits in polynomial time.

Proof. By the Prime Number Theorem, there are approximately $x/\ln x$ prime numbers $p \le x$. Consider primes of length k bits. There are

$$\frac{2^k}{\ln 2^k} - \frac{2^{k-1}}{\ln 2^{k-1}} \approx \frac{1}{\ln 2}\left(\frac{2^k}{k} - \frac{2^{k-1}}{k-1}\right) = \frac{1}{\ln 2}\frac{(k-1)2^k - k2^{k-1}}{k(k-1)}$$

$$= \frac{1}{\ln 2}\frac{2k2^{k-1} - 2^k - k2^{k-1}}{k(k-1)} = \frac{1}{\ln 2}\frac{k2^{k-1} - 2^k}{k(k-1)} = \frac{1}{\ln 2}\frac{k-2}{k(k-1)}2^{k-1}$$

such primes. Since we need to find $2m$ primes, at any point there are at least $\frac{1}{\ln 2}\frac{k-2}{k(k-1)}2^{k-1} - 2m$ primes left of length k bits. Note that there are $2^k - 2^{k-1} = 2^{k-1}$ integers of length k bits, so at any point the probability that a random number chosen is prime is at least $\frac{\frac{1}{\ln 2}\frac{k-2}{k(k-1)}2^{k-1} - 2m}{2^{k-1}} = \frac{1}{\ln 2}\frac{k-2}{k(k-1)} - \frac{2m}{2^{k-1}}$. For $k = \frac{\lambda}{2}$ this becomes $\frac{1}{\ln 2}\frac{\frac{\lambda}{2}-2}{\frac{\lambda}{2}\left(\frac{\lambda}{2}-1\right)} - \frac{2m}{2^{\frac{\lambda}{2}-1}} = \frac{1}{\ln 2}\frac{2\lambda - 8}{\lambda(\lambda-2)} - \frac{m}{\sqrt{2}^{\lambda-4}}$. If m is polynomial in λ (i.e. $m \ll \sqrt{2}^{\lambda-4}$), then this probability is nonnegligible. Since it is possible to check whether a number is prime in polynomial time, each of the primes can be found in polynomial time. Since m is polynomial in λ, the number of primes we must find is polynomial in λ, so the time complexity of the algorithm is polynomial in λ. ■

Lemma 2: Suppose that large integer factorization is infeasible. Let $f_i = p_i q_i$ for large primes p_i and q_i, where $1 \le i \le m$. Then given $F = \{f_i\}_{i=1}^m$, it is infeasible to factor any of the $f_i \in F$, i.e. there does not exist a PPT (probabilistic polynomial time) algorithm that can return a factor of f_i for any $1 \le i \le m$ with nonnegligible probability.

Proof. Assume that a PPT algorithm A_1 exists that, given a set of large integers $F = \{f_i\}_{i=1}^m$, can randomly factor one of the f_i with some probability p'. Let A_2 be an algorithm to factor some large integer $f = pq$ for primes p and q as follows. First construct the set $F = \{f_1, \dots, f_{m-1}, f_m = f\}$ where f_i is random for $1 \le i \le m-1$, and then run A_1 on this set F, and return the result. Note that A_1 successfully factors one of the f_i with probability p', and the probability that $i = m$ is $\frac{1}{m}$. Thus A_2 is successful with nonnegligible probability $\frac{p'}{m}$. Clearly A_2 is a PPT algorithm, completing the proof. ■

Lemma 3: Let $N = \prod_{i=1}^m p_i$, where the p_i are prime and $m = O(\text{poly}(\lambda))$. A random matrix $T \in M_4(Z_N)$ is invertible with a high probability.

Proof. Note that an $l \times l$ matrix $T' = (\alpha_1, \dots, \alpha_l)^t$ over a field Z_p , where t denotes the transpose of a matrix and $\alpha_i \in (Z_p)^l$, is not invertible if and only if $\exists c_i \in Z_p$ and $c_{i_0} \ne 0$ for some i_0, s.t. $\sum_{i=1}^l c_i \alpha_i = 0$. The condition is equivalent to $\alpha_{i_0} = \sum_{i \ne i_0} c_i' \alpha_i$, where $c_i' = -c_{i_0} c_i$. Since T' has l^2 entries, totally there are p^{l^2} possibilities for T'. Since $i \ne i_0$, there are a total of p^{l-1} possibilities for c_i', and p^{l^2-l} possibilities for α_i. Hence, given T', the probability that $\alpha_{i_0} = \sum_{i \ne i_0} c_i' \alpha_i$ is $\frac{p^{l-1} p^{l^2-l}}{p^{l^2}} = \frac{1}{p}$. Note that $1 \le i_0 \le l$, so T' is not invertible with probability at most $\frac{l}{p}$.

We show that a matrix T over the ring Z_N is invertible if and only if it is invertible over every field Z_{p_i}, where $1 \le i \le m$. Note that if T is invertible over Z_N, then we can construct $T^{-1} \bmod p_i$. Since $TT^{-1} = I$ (the identity matrix), $TT^{-1} = I \bmod p_i$, so $T^{-1} \bmod p_i$ is the inverse of $T \bmod p_i$, so T is invertible over each field Z_{p_i}. Now assume that T is invertible over each field p_i with inverse S_i, and consider the set of linear congruences $S = S_i \bmod p_i$, which has a unique solution over Z_N by the Chinese Remainder

Theorem. Then we have the set of linear congruences $TS = TS_i \bmod p_i = I \bmod p_i$. But I is clearly a solution to these congruences, and by the Chinese Remainder Theorem, this solution is unique over Z_N, so $TS = I$, so T is invertible with inverse S. As shown above, this means that $T \in M_4(Z_N)$ is not invertible with negligible probability

$$1 - \prod_{i=1}^{m}\left(1 - \frac{4}{p_i}\right) \le \sum_{i=1}^{m}\frac{4}{p_i} \le \frac{4m}{\min\limits_{1\le i\le m} p_i}.$$

This completes the proof. ■

Lemma 4: The encryption scheme (K, E, D) is correct.
Proof: Note that $((k^{-1})^{-1}(k^{-1}\text{diag}(x,a,b,c)k)k^{-1})_{00} = \text{diag}(x,a,b,c)_{00} = x$. ■

Lemma 5: For $1 \le i \le N$, there exists a unique element $k_i \in GL_4(Z_N)$ so that $k_i = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \bmod p_i$, $k_i = I \bmod q_i$, and $k_i = I \bmod f_j$ for $j \ne i$ where I is the identity matrix in $GL_4(Z_N)$. Additionally, $k_i = k_i^{-1}$.
Proof. The first claim follows from the Chinese Remainder Theorem. The fact that $k_i = k_i^{-1}$ follows from the fact that

$$k_i^2 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}\begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \bmod p_i = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \bmod p_i = I \bmod p_i$$

and the trivial fact that $I = I^{-1}$. ■

Lemma 6: Given plaintext x, key k and random element r, there exist y and s such that $E(x,k) = E(y, k_i k)$. Additionally, $y - x$ divides q_i, and $y - x$ does not divide p_i with probability $\frac{1}{m+1}\left(1 - \frac{1}{p_i}\right)$.
Proof. Note that $C' = \text{diag}(x,a,b,c)$ satisfies the congruences $C' = \text{diag}(x, a_i, b_i, c_i) \bmod f_i$, so

$$\begin{aligned} k_i C' k_i^{-1} \bmod p_i &= k_i \text{diag}(x, a_i, b_i, c_i) k_i^{-1} \bmod p_i \\ &= \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}\begin{pmatrix} x & 0 & 0 & 0 \\ 0 & a_i & 0 & 0 \\ 0 & 0 & b_i & 0 \\ 0 & 0 & 0 & c_i \end{pmatrix}\begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \bmod p_i \\ &= \text{diag}(a_i, x, c_i, b_i) \bmod p_i \end{aligned}$$

Also, $k_i C' k_i^{-1} \bmod q_i = IC'I \bmod q_i = C' \bmod q_i$ and similarly $k_i C' k_i^{-1} \bmod f_j = C' \bmod f_j$ for $j \ne i$. Let $D' = \text{diag}(y, a', b', c')$. Then the set of congruences $D' = \text{diag}(a_i, x, c_i, b_i) \bmod p_i$, $D' = C' \bmod q_i$, and $D' = C' \bmod f_j$ has a unique solution by the Chinese Remainder Theorem. Note that this solution satisfies $D' = k_i C' k_i^{-1}$, so $k_i^{-1} D' k_i = C'$ and $(k_i k)^{-1} D' (k_i k) = k^{-1} k_i^{-1} D' k_i k = k^{-1} C' k = E(x,k)$. But this means that $E(x,k) = (k_i k)^{-1} D' (k_i k) = E(y, k_i k)$, proving the first claim. Note that additionally $y = x \bmod q_i$ so that $y - x$ divides q_i, but $y = a_i \bmod p_i$. Note that $a_i = r$ (i.e., $a_i \ne x$) with probability $\frac{1}{m+1}$. Additionally, $r \ne x \bmod p_i$ with probability $1 - \frac{1}{p_i}$ since r is chosen uniformly randomly on Z_N, and thus with probability $1 - \frac{1}{p_i}$, $r - x$ does not divide p_i. Thus $y - x$ divides q_i, and $y - x$ does not divide p_i with probability $\frac{1}{m+1}\left(1 - \frac{1}{p_i}\right)$, proving the second claim. ■

Lemma 7: If a PPT algorithm $A_d(C)$ exists that, given $C = E(x,k)$, returns x with probability p, then there exists a PPT algorithm A_f to return a factor f_i for some i with probability $p' = \frac{p}{m+1}\left(1 - \frac{1}{p_i}\right)$.
Proof. Let the algorithm A_f first choose a random plaintext x and a random key k, and construct ciphertext $C = E(x,k)$ using the encryption scheme. Then, A_f runs $A_d(C)$ to obtain value o. Then A_f returns

$\gcd(f_i, o-x)$. Note that A_f is clearly a PPT algorithm assuming that A_d is a PPT algorithm. We also show that A_f is correct with some probability p'. Note that since $E(x,k) = E(y, k_i k)$, A_d will also return y with probability p. Note that $q_i = \gcd(f_i, o-x) = \gcd(f_i, y-x)$ with probability $\frac{1}{m+1}\left(1-\frac{1}{p_i}\right)$ since $y-x$ divides q_i but not p_i with this probability. Since f_i has only factors p_i and q_i, the GCD of this pair of numbers is thus q_i. If p is nonnegligible, then A_f is thus correct with nonnegligible probability $p' = \frac{p}{m+1}\left(1-\frac{1}{p_i}\right)$. Clearly A_f is a PPT algorithm if A_d is a PPT algorithm, completing the proof. ■

Lemma 8: Let m' be the number of plaintext and ciphertext pairs the adversary has access to. If for some m' there exists an algorithm $A_d\left(C = E(x,k), \left\{\left(x_l, C_l = E(x_l,k)\right)\right\}_{l=1}^{m'}\right)$ such that, given m' chosen plaintext and ciphertext pairs (x_l, C_l) and a ciphertext C, returns x with some probability p, then there exists a PPT algorithm A_f using A_d as an oracle to factor f_i for some i with probability

$$p' = p\left(1-\frac{1}{p_i}\right)\left(1-\left(1-\frac{1}{m+1}\left(1-\frac{1}{m+1}\right)^{m'}\right)^m\right).$$

Proof. As before, let the algorithm A_f first choose random plaintexts $x_i \in Z_N$ for $1 \le i \le m'$, an additional random plaintext $x \in Z_N$, and a random key k, and construct ciphertexts $C_i = E(x_i, k)$ and $C = E(x,k)$ using the encryption scheme. Then let A_f run $A_d(C)$ to obtain o. Then let A_f return $\gcd(f_i, o-x)$ as a factor of f_i for some i. We find the probability that A_f succeeds in factoring f_i for some i. Consider the case where, for some i_0, for all of the ciphertexts C_l, $a_{l,i_0} = x_l$, and where for the ciphertext C, $a_{i_0} = r$. Here a_{l,i_0} refers to the a_{i_0} in the encryption process of C_l, and a_{i_0} refers to the a_{i_0} in the encryption process of C. If this is the case, then as seen in the proof of lemma 7, $E(x_l,k) = E\left(x_l, k_{i_0}k\right)$ (since a_{l,i_0} is assumed to equal x_l), so the adversary cannot differentiate k from $k_{i_0}k$. Additionally, as in lemma 7, $E(x,k) = E\left(y, k_{i_0}k\right)$ for some y for which $y-x$ divides q_{i_0}. Also, $y = a_{i_0} \bmod p_{i_0} = r \bmod p_{i_0}$, so $o-x = y-x$ does not divide p_{i_0} with probability $1-\frac{1}{p_{i_0}}$. Since the adversary cannot differentiate k from $k_{i_0}k$, if running $A_d(C)$ returns x with probability p, it must also return y with probability p. Then the probability that $o = y$ is p, and if this happens the probability that $o-x = y-x$ divides q_i and does not divide p_i is $1-\frac{1}{p_{i_0}}$. Thus if i_0 exists then A_f succeeds with probability $p\left(1-\frac{1}{p_i}\right)$.

To find the probability that such an i_0 exists, note that the probability that, for a specific l and i, $a_{l,i} = x_l$ is $1-\frac{1}{m+1}$, so the probability that for all l, $a_{l,i} = x_l$ is $\left(1-\frac{1}{m+1}\right)^{m'}$. The probability that, additionally, $a_i = r$ is $\frac{1}{m+1}\left(1-\frac{1}{m+1}\right)^{m'}$. Then the probability that this does not occur for a given i is $1-\frac{1}{m+1}\left(1-\frac{1}{m+1}\right)^{m'}$, so the probability that this does not occur for any i is $\left(1-\frac{1}{m+1}\left(1-\frac{1}{m+1}\right)^{m'}\right)^m$, and finally the probability that this occurs for some i is thus $1-\left(1-\frac{1}{m+1}\left(1-\frac{1}{m+1}\right)^{m'}\right)^m$.

Finally we integrate the derivations and get

$$p' = p\left(1-\frac{1}{p_i}\right)\left(1-\left(1-\frac{1}{m+1}\left(1-\frac{1}{m+1}\right)^{m'}\right)^m\right). ■$$

Lemma 9: Assuming that the probability to factor an λ bit integer in polynomial time is negligible, the encryption scheme is secure for $m' \le m$.

Proof. As seen the equation of p' in Lemma 9, $\left(1-\frac{1}{p_i}\right)$ is nonnegligible. If $1-\left(1-\frac{1}{m+1}\left(1-\right.\right.$

$\left.\left.\frac{1}{m+1}\right)^{m'}\right)^{m}$ is further nonnegligible, then p is negligible if and only if p' is negligible. Thus, it implies that the encryption scheme is secure. Otherwise, if a PPT adversary can attack the scheme with nonnegligible success probability p, then there will exist a PPT algorithm to factor some integer with nonnegligible success probability p', which contradicts to Lemma 2. Therefore, the encryption scheme is secure if $1-\left(1-\frac{1}{m+1}\left(1-\frac{1}{m+1}\right)^{m'}\right)^{m}$ is nonnegligible.

Note that $\frac{d}{d\alpha}\left(1-\frac{1}{\alpha}\right)^{\alpha}=\alpha\left(1-\frac{1}{\alpha}\right)^{\alpha-1}\alpha^{-2}=\frac{\left(1-\frac{1}{\alpha}\right)^{\alpha-1}}{\alpha}>0$ for $\alpha>0$. Thus the function is monotonically increasing, so on Z^{+} it achieves its minimum at $\alpha=1$, where it takes the value $\left(1-\frac{1}{1+1}\right)^{1+1}=\frac{1}{4}$. Additionally, since $\lim_{\alpha\to\infty}\left(1-\frac{1}{\alpha}\right)^{\alpha}=\frac{1}{e}$ and since $\left(1-\frac{1}{\alpha}\right)^{\alpha}$ is monotonically increasing, $\left(1-\frac{1}{\alpha}\right)^{\alpha}\leq\frac{1}{e}$. Then we obtain

$$1-\left(1-\frac{1}{m+1}\left(1-\frac{1}{m+1}\right)^{m'}\right)^{m}\geq 1-\left(1-\frac{1}{m+1}\left(1-\frac{1}{m+1}\right)^{m+1}\right)^{m}\geq 1-\left(1-\frac{1}{4(m+1)}\right)^{m}$$

$$=1-\left(\left(1-\frac{1}{4(m+1)}\right)^{4(m+1)}\right)^{\frac{m}{4(m+1)}}$$

$$\geq 1-\left(\frac{1}{e}\right)^{\frac{m}{4(m+1)}}\geq 1-\left(\frac{1}{2}\right)^{\frac{1}{4(1+1)}}=1-\left(\frac{1}{2}\right)^{\frac{1}{8}}\geq 1-.92=.08 \qquad \blacksquare$$

Theorem 1: The bound of m' in Lemma 10 can be weakened to $m'\leq m\ln\mathrm{poly}(\lambda)$, where $\mathrm{poly}(\lambda)$ denotes some fixed polynomial in λ.

Proof. As shown in Lemma 10, the encryption scheme is secure if $1-\left(1-\frac{1}{m+1}\left(1-\frac{1}{m+1}\right)^{m'}\right)^{m}$ is nonnegligible. In other words, we require that $1-\left(1-\frac{1}{m+1}\left(1-\frac{1}{m+1}\right)^{m'}\right)^{m}=\frac{1}{\mathrm{poly}(\lambda)}$ for some fixed polynomial in λ. Then

$$m'=\frac{\ln\left((m+1)\left(1-\left(1-\frac{1}{\mathrm{poly}(\lambda)}\right)^{\frac{1}{m}}\right)\right)}{\ln\left(1-\frac{1}{m+1}\right)}$$

Before estimating the lower bound of m', we first derive two inequalities. Note that $\frac{d}{d\alpha}\left(\ln(1-\alpha)+\frac{\alpha}{1-\alpha}\right)=\frac{\alpha}{(1-\alpha)^2}>0$ for $0<\alpha<1$, and $\left(\ln(1-\alpha)+\frac{\alpha}{1-\alpha}\right)|_{\alpha=0}=0$.
Therefore $\ln(1-\alpha)>-\frac{\alpha}{1-\alpha}$ for $0<\alpha<1$. Also, note that $\frac{d}{d\alpha}(\alpha-1+e^{-\alpha})=1-e^{-\alpha}>0$ for $0<\alpha<1$, and $(\alpha-1+e^{-\alpha})|_{\alpha=0}=0$.
Therefore $\alpha>1-e^{-\alpha}$ for $0<\alpha<1$. Thus,

$$\ln\left(1-\frac{1}{\mathrm{poly}(\lambda)}\right)>-\frac{\frac{1}{\mathrm{poly}(\lambda)}}{1-\frac{1}{\mathrm{poly}(\lambda)}}=-\frac{1}{\mathrm{poly}(\lambda)-1}$$

$$\to\ln\left(1-\frac{1}{\mathrm{poly}(\lambda)}\right)^{\frac{1}{m}}=\frac{1}{m}\ln\left(1-\frac{1}{\mathrm{poly}(\lambda)}\right)>-\frac{1}{m(\mathrm{poly}(\lambda)-1)}\quad\to\left(1-\frac{1}{\mathrm{poly}(\lambda)}\right)^{\frac{1}{m}}>e^{-\frac{1}{m(\mathrm{poly}(\lambda)-1)}}$$

$$\to 1-\left(1-\frac{1}{\mathrm{poly}(\lambda)}\right)^{\frac{1}{m}}<1-e^{-\frac{1}{m(\mathrm{poly}(\lambda)-1)}}<\frac{1}{m(\mathrm{poly}(\lambda)-1)}$$

$$\to \ln\left((m+1)\left(1-\left(1-\frac{1}{\mathrm{poly}(\lambda)}\right)^{\frac{1}{m}}\right)\right) < \ln\frac{m+1}{m(\mathrm{poly}(\lambda)-1)} = -\ln\left(\frac{m(\mathrm{poly}(\lambda)-1)}{m+1}\right) = -\ln \mathrm{poly}(\lambda) \quad (*)$$

The last equation in $(*)$ is obtained by replacing $\mathrm{poly}(\lambda)$ with the polynomial $\mathrm{poly}(\lambda)+1+\frac{\mathrm{poly}(\lambda)}{m}$. Also, we have

$$\ln\left(1-\frac{1}{m+1}\right) > -\frac{\frac{1}{m+1}}{1-\frac{1}{m+1}} = -\frac{1}{m} \to \frac{1}{\ln\left(1-\frac{1}{m+1}\right)} < -m \quad (**)$$

Hence, by multiplying $(*)$ and $(**)$, we get

$$m' = \frac{\ln\left((m+1)\left(1-\left(1-\frac{1}{\mathrm{poly}(\lambda)}\right)^{\frac{1}{m}}\right)\right)}{\ln\left(1-\frac{1}{m+1}\right)} > m\ln \mathrm{poly}(\lambda)\,. \quad \blacksquare$$

Lemma 10: The multiplication and addition algorithms are correct.

Proof. First we show that addition is correct. Note that

$$\begin{aligned} E(x,k)+E(y,k) &= k^{-1}\mathrm{diag}(x,a,b,c)k + k^{-1}\mathrm{diag}(y,a',b',c')k \\ &= k^{-1}\big(\mathrm{diag}(x,a,b,c)+\mathrm{diag}(y,a',b',c')\big)k \\ &= k^{-1}\mathrm{diag}(x+y,a+a',b+b',c+c')k = E(x+y,k) \end{aligned}$$

Next we show that multiplication is correct. Note that

$$\begin{aligned} E(x,k)E(y,k) &= k^{-1}\mathrm{diag}(x,a,b,c)kk^{-1}\mathrm{diag}(y,a',b',c')k = k^{-1}\mathrm{diag}(x,a,b,c)\mathrm{diag}(y,a',b',c')k \\ &= k^{-1}\mathrm{diag}(xy,aa',bb',cc')k = E(xy,k). \quad \blacksquare \end{aligned}$$

Lemma 11: If $C = E(x,k)$, then $\varphi(C,k') = E(x,k\cdot k')$.

Proof. $\varphi(C,k') = \varphi(E(x,k),k') = k'^{-1}\cdot E(x,k)\cdot k' = k'^{-1}\cdot k^{-1}\mathrm{diag}(x,a,b,c,)k\cdot k'$
$= (k\cdot k')^{-1}\cdot \mathrm{diag}(x,a,b,c,)\cdot(k\cdot k') = E(x,k\cdot k')$. ■

Theorem 2: Suppose that the adversary attacks the multi-user system with the adversary structure AS. The system is secure if the adversary collects less than $m\ln \mathrm{poly}(\lambda)$ plaintext-ciphertext pairs.

Proof. We consider three compromising situations with respect to the adversary structure AS.

Case 1. The adversary compromises DB and KA. Then k''_i and k'_i are compromised, but the master key k and the key k_i of the user C_i are intact. Therefore the adversary can neither reverse the ciphertext encrypted by k stored on DB, nor the ciphertext encrypted by k_i sent from C_i to KA.

Case 2. The adversary compromises DB and C_A. Then k''_i are compromised, k_j are compromised if $C_j \in C_A$, but the master key k and the key $k_{j'}$ of the user $C_{i'}$ are intact if $C_{j'} \notin C_A$. Therefore the adversary can neither reverse the ciphertext encrypted by k stored on DB, nor the ciphertext encrypted by $k_{j'}\cdot k'_{j'}$ sent from KA to DB.

Case 3. The adversary compromises KA and C_A. Then k'_i are compromised, k_j are compromised if $C_j \in C_A$, but the key $k_{j'}$ of the user $C_{i'}$ are intact if $C_{j'} \notin C_A$. Therefore the adversary cannot reverse the ciphertext encrypted by $k_{j'}$ sent from $C_{j'}$ to KA, nor the ciphertext encrypted by $k_{j'}\cdot k'_{j'}$ sent from DB to KA.

Thus it completes the proof. ■