International Journal of Digital Crime and Forensics
Volume 11 • Issue 3 • July-September 2019

Cloud-ElGamal and Fast Cloud-RSA Homomorphic Schemes for Protecting Data Confidentiality in Cloud Computing

Khalid El Makkaoui, LAVETE Laboratory, FST, Univ Hassan 1, Settat, Morocco
Abderrahim Beni-Hssane, Department of Computer Science, Faculty of Sciences, Chouaïb Doukkali University, El Jadida, Morocco
Abdellah Ezzati, LAVETE Laboratory, FST, Univ Hassan 1, Settat, Morocco

ABSTRACT

Homomorphic encryption (HE) is an encryption form that offers a third-party with the ability to carry out computations on encrypted data. This property can be considered as a great solution to get over some obstacles limiting the wide-spread adoption of cloud computing (CC) services. Since CC environments are threatened by insider/outsider security attacks and since CC consumers often access to CC services using resource-limited devices, the HE schemes need to be promoted at security level and at running time to work effectively. For this reason, at EMENA-TSSL'16 and at WINCOM'16, the authors respectively boosted the RSA and ElGamal cryptosystems at security level, Cloud-RSA and Cloud-ElGamal. At SCAMS'17 and at EUSPN'17, the authors then suggested two fast variants of the Cloud-RSA scheme. All proposed schemes support the multiplicative homomorphism (MH) over the integers. The aim of this article is to compare the Cloud-ElGamal scheme with the Cloud-RSA schemes. This article first briefly presents the HE schemes and analyzes their security. This article then implements the schemes, compare and discuss their efficiency.

KEYWORDS

Cloud Computing (CC), Cloud-ElGamal Scheme, Cloud-RSA Scheme, Confidentiality, Homomorphic Encryption (HE), Multiplicative Homomorphism (MH)

INTRODUCTION

Cloud computing (CC) provides computing resources (e.g., storage, networks, servers, applications, etc.) as on-demand services through Internet (Armbrust et al., 2009; Hu et al., 2016). Lately, there has been a growing adoption of CC services. This is due to a number of privileges provided by CC providers including reducing costs, high scalability and elasticity of services, and powerful computations (Kiraz, 2016; Zhou, 2018). Nevertheless, the worry to see sensitive data being processed in clear within CC environment is still the main barrier restricting the wide-spread adoption of CC services (El Makkaoui et al., 2016c; Alam et al., 2017; Gao et al., 2018).

DOI: 10.4018/IJDCF.2019070105

The utilization of some techniques which can carry out computations on encrypted data appears to be an efficient way to address this barrier and to build trust in CC solutions (El Makkaoui et al., 2016c; Yan et al., 2017). In fact, scientists have stressed a promising technique, homomorphic encryption (HE), which offers a third-party having the capability to execute operations on encrypted data (i.e., can ensure the confidentiality of both storage and processing of outsourced data into CC environment) (Yi et al., 2014). The homomorphic property makes HE schemes useful in a number of privacy preserving applications e.g., electronic voting and digital healthcare (Liu and Ke, 2018).

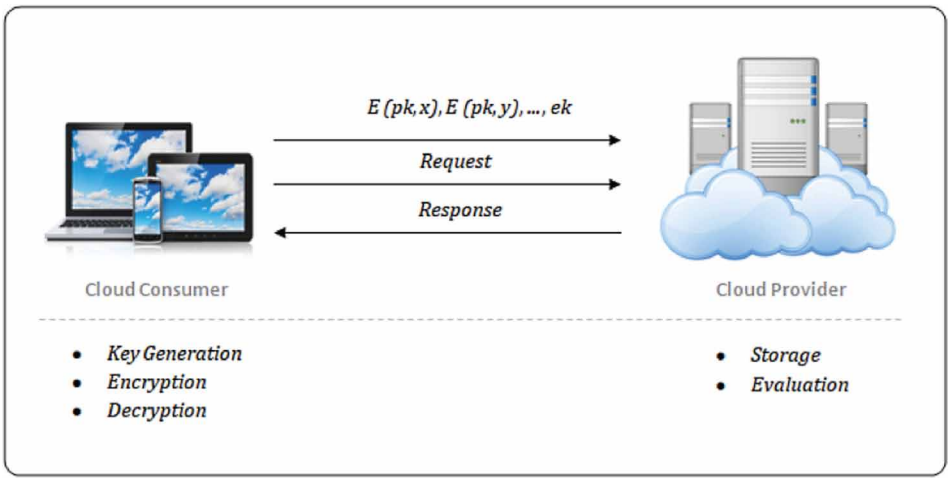
The HE's concept was introduced in 1978 by Rivest et al. (Rivest et al., 1978a). Since that time, many encryption schemes support homomorphic properties have been suggested. The HE schemes can be divided into two classes: somewhat and full homomorphic encryption. The schemes of the first class support one homomorphic property (usually additive homomorphism or multiplicative homomorphism) or even more but in a limited way e.g., (El Makkaoui et al., 2017a, 2017b, 2017c; El Makkaoui et al., 2016a, 2016b). While the full homomorphic schemes support many homomorphic properties simultaneously e.g., (Gentry, 2009; Van Dijk et al., 2010).

A consumer-provider model of CC employing homomorphic schemes can be represented in Figure 1, where:

- **Key Generation:** The CC consumer produces the pair key: a private key (*pk*) and an evaluation key (*ek*);
- **Encryption:** The CC consumer encrypts plaintexts using pk. He/she then sends the ciphertexts and the evaluation key ek to the selected CC provider;
- **Storage:** The ciphertexts and ek are stored in the CC environment;
- **Request:** The CC consumer requests the CC provider to execute operations on encrypted data;
- **Evaluation:** The processing server performs the requested operations using *ek*;
- **Response:** The CC provider returns to the CC consumer the processed encrypted results;
- **Decryption:** The CC consumer decrypts the results utilizing *pk*.

Since outsourced encrypted data under HE schemes can be stored for a long-time, usually by the same key, in CC environments and since the majority of CC consumers access to CC services utilizing resource-limited devices (e.g., smart-phone, tablet, etc.), these HE schemes need to be promoted in terms of security level and of running time to work effectively.

Figure 1. CC consumer-provider model employing HE schemes



For this reason, in (El Makkaoui et al., 2017c), we boosted the RSA cryptosystem (Rivest et al., 1978b) at security level, Cloud-RSA, to work efficiently in CC environments. The Cloud-RSA provides the MH over the integers and withstands more confidentiality attacks. The scheme is based on the generation of an RSA modulus $n = pq$ which is the product of two distinct primes and the generation of two integers e and d satisfying $d \equiv e^{-1} \pmod{(p-1)(q-1)}$. The modulus ($n$) is the evaluation key and the triple (n, e, d) is the private key. The encryption scheme maintains the same form of the RSA encryption and decryption. The encryption and decryption process are done utilizing the private key. To get a fast Cloud-RSA decryption, in (El Makkaoui et al., 2017a), we proposed a variant of the Cloud-RSA scheme, MultiPrime Cloud-RSA. The proposed variant uses an RSA modulus n formed of $k \geq 2$ distinct primes and employs Chinese remaindering (Katz et al., 1996) to decrypt. In order to get an extra decryption speed-up, in (El Makkaoui et al., 2017b), we suggested a new variant of the Cloud-RSA, MultiPower Cloud-RSA. The MultiPower Cloud-RSA scheme uses a modulus $n = p^r q^s$ for $r \geq 2$ and $s \geq 1$, and employs Hensel lifting (Yun, 1974) and Chinese remaindering to decrypt.

For similar reasons, in (El Makkaoui et al., 2016a), we boosted the ElGamal cryptosystem ElGamal (1985) at security level, Cloud-ElGamal, to work effectively in CC environments. The Cloud-ElGamal offers also the MH over the integers, it is semantically secure, and it withstands more confidentiality attacks.

In this work, we implement and compare the Cloud-ElGamal scheme with the Cloud-RSA schemes. The implementation will be carried out in a smart-phone.

We begin this article with related work. We then present the proposed HE schemes. We describe the algorithms of the encryption schemes and analyze their security. We then implement the schemes, compare and discuss their efficiency.

RELATED WORK

The HE's concept was introduced by Rivest et al. (1978a). Since that time, many tries have been made to find practical HE schemes.

Rivest et al. (1978b) created the first practical encryption scheme, called RSA, which supports the multiplicative homomorphism (MH) over the integers. The RSA is secure under the difficulty of factoring large composite integers and of the e^{th} root problem.

ElGamal (1985) proposed a new encryption scheme which is secure under the problem of discrete logarithm over finite fields. ElGamal's scheme offers the MH and is semantically secure.

Paillier (1999) suggested two HE cryptosystems to support the additive homomorphism (AH) on the integers. Paillier's encryption schemes achieve semantic security based on the Decisional Composite Residuosity.

Gentry (2009) proposed the first FHE scheme. Its security depends upon the euclidean lattices. The Gentry's scheme encrypts plaintexts encoded as {0, 1} by adding randoms, whilst decryption is made up of getting rid of the randoms. The running time depends linearly on the number of operations carried out. Nevertheless, the Gentry's homomorphic scheme is impractical for several applications since running time and ciphertexts size increase greatly as one increases the level of security.

van Dijk et al. (2010) suggested a new FHE scheme, known as DGHV scheme. Instead of Euclidean lattices, the DGHV's operations are executed on the integers. The homomorphic scheme encrypts plaintexts in {0,1} and accomplishes semantic security based on the approximate greatest common divisor problem. Nevertheless, the DGHV's public-key is in $\tilde{O}(\lambda^{10})$ which is too large for practical applications.

Wang (2016) suggested a FHE scheme which is based upon octonion algebra over finite rings Z_n. The HE properties are obtained via regular matrix functions. The homomorphic scheme is just secure within the weak ciphertext-only security model. Moreover, it is insecure against enemy who has access to sufficiently many linearly impartial ciphertexts with knowing session randomness and plaintexts.

In El Makkaoui et al. (2016a), we boosted ElGamal's encryption scheme at security level by suggesting a new variant of the encryption scheme, called Cloud-ElGamal. The suggested scheme supports the MH over the integers, is semantically secure and is resistant to confidentiality attacks.

In El Makkaoui et al. (2016b), we boosted the main Paillier's scheme (Paillier, 1999) at security level to work effectively in cloud environments, we refer to as Cloud-Paillier. The Cloud-Paillier addresses an exception of the Paillier's encryption scheme, we showed that one cannot eliminate random integers padding in the encryption process if the integers are not relatively prime to the Paillier's modulus. The Cloud-Paillier supports the AH over the integers and it is resistant to confidentiality attacks. It accomplishes semantic security under double trapdoor assumptions: the difficulty of factoring large composite integers and the hard decisional composite residuosity assumption.

In (El Makkaoui et al., 2017c), we boosted the RSA cryptosystem (Rivest et al., 1978b) at security level, called Cloud-RSA, to work efficiently in cloud environments. The Cloud-RSA provides the MH over the integers and withstands more confidentiality attacks. To get a fast Cloud-RSA decryption, in El Makkaoui et al. (2017a), we suggested a variant of the Cloud-RSA scheme, called MultiPrime Cloud-RSA. The proposed variant slightly modifies the form of the Cloud-RSA modulus and employs Chinese remaindering (Katz et al., 1996) to decrypt. In order to get an extra decryption speed-up, in El Makkaoui et al. (2017b), we suggested a new variant of the Cloud-RSA, we refer to as MultiPower Cloud-RSA. The MultiPower Cloud-RSA scheme modifies the form of the Cloud-RSA modulus, employs Hensel lifting (Yun, 1974) and Chinese remaindering to decrypt.

In this paper, we focus on the Cloud-ElGamal, Cloud-RSA, MultiPower Cloud-RSA and MultiPower Cloud-RSA schemes. We implement and compare, in terms of encryption and decryption running time, the Cloud-ElGamal scheme with the Cloud-RSA schemes.

PROPOSED HE SCHEMES

In this section, we describe the proposed HE schemes, namely Cloud-RSA scheme (El Makkaoui et al., 2017c), fast Cloud-RSA's variants (El Makkaoui et al., 2017a, 2017b) and Cloud-ElGamal scheme (El Makkaoui et al., 2016a).

Cloud-RSA Scheme

We present the constituent algorithms of the Cloud-RSA scheme. We then analyze its security level.

Algorithms

The Cloud-RSA encryption scheme is based on the generation of a modulus n formed of two distinct prime numbers p and q, and on the generation of two integers e and d satisfying $d \equiv e^{-1} \, (mod \, (p-1)(q-1))$. The modulus n is made public and represents the evaluation key, and the integers e and d are made secret and represent the private key. The Cloud-RSA key generation process is highlighted in Algorithm 1.

The Cloud-RSA scheme encrypts plaintexts under the private key as explained in Algorithm 2. Z_n denotes the ring of integers modulo n and its complete residue class is $\{0, 1, \ldots, n-1\}$.

Algorithm 1. Cloud-RSA key generation

INPUT: Two large distinct primes p and q of the same bits long.
OUTPUT: An evaluation key $ek = (n)$ and a private key $pk = (n, e, d)$.
1: Set $n \leftarrow pq$ and $\phi(n) \leftarrow (p-1)(q-1)$.
2: Pick an integer e that is relatively prime to $\phi(n)$.
3: Set $d \leftarrow e^{-1} \pmod{\phi(n)}$.
4: Return (n) and (n, e, d).

Algorithm 2. Cloud-RSA encryption

INPUT: pk and plaintext $m \in \mathbb{Z}_n$.
OUTPUT: $c = E(pk, m)$.
1: Compute $c \equiv m^e \pmod{n}$.
2: Return c.

The selected CC provider performs the MH over the ciphertexts using the evaluation key (as outlined in Algorithm 3).

The Cloud-RSA scheme decrypts the ciphertexts under the private key as highlighted in Algorithm 4.

Security

The security of the Cloud-RSA scheme depends upon the hardness of factoring a composite integer of the form $n = pq$ and discovering the decryption exponent d when the encryption exponent e is also private.

The prime factors of n must be cautiously chosen to prevent the speediest factorization attacks, namely the Elliptic Curve Method (ECM) (Lenstra, 1987) and the Number Field Sieve (NFS) (Lenstra et al., 1993).

Even if an attacker succeeds in factorizing the modulus n, he/she must discover two exponents e and d that satisfies $ed \equiv 1 \pmod{\varphi(n)}$ and that the decrypted data is semantically proper.

Fast Variants of Cloud-RSA

In this part, we present two fast variants of the Cloud-RSA scheme. The first variant, called MultiPrime Cloud-RSA (El Makkaoui et al., 2017a), uses a modulus n formed of $k \geq 2$ distinct prime numbers and employs the Chinese remaindering to decrypt. The second fast variant, called MultiPower Cloud-RSA (El Makkaoui et al., 2017b), uses a modulus $n = p^r q^s$ for $r \geq 2$ and $s \geq 1$, and utilizes Hensel lifting and Chinese remaindering to decrypt.

MultiPrime Cloud-RSA Scheme

We present the key generation and decryption algorithms of the MultiPrime Cloud-RSA encryption scheme. We then analyze the security level of the scheme.

Algorithm 3. Cloud-RSA evaluation

INPUT: ek, and ciphertexts $c_1, \ldots, c_j \in \mathbb{Z}_n$.
OUTPUT: C as the result.
1: Compute $C \equiv c_1 \times \ldots \times c_j \pmod{n}$.
2: Return C.

Algorithm 4. Cloud-RSA decryption

INPUT: pk and a ciphertext C.
OUTPUT: $D(pk, C) = m_1 \times \ldots \times m_j$
1: Compute $D(pk, C) = C^d \pmod{n} \equiv m_1 \times \ldots \times m_j$.
2: Return $D(pk, C)$.

Algorithms

The MultiPrime Cloud-RSA scheme is based on the generation of a modulus n= $p_1 \ldots p_k$ for $k \geq 2$, where the primes $p_1, \ldots, p_k$ are distinct. And it is based on the generation of an integer e and k integers d_i, for $i = 1, \ldots, k$, satisfying $d_i \equiv e^{-1} \pmod{(p_i - 1)}$. The modulus n is made public and represents the evaluation key, and the integers e and d_i are made private. To obtain an extra decryption speed-up, we pre-compute the Chinese remaindering coefficients and store it as a part of the private key. The MultiPrime Cloud-RSA key generation process is highlighted in Algorithm 5.

The MultiPrime Cloud-RSA scheme keeps the same form of the Cloud-RSA encryption algorithm (Algorithm 2).

The CC provider carries out the MH over the ciphertexts using the evaluation key (as illustrated in Algorithm 3).

The CC consumer decrypts the return encrypted result under the MultiPrime Cloud-RSA private key employing the Chinese remaindering (as highlighted in Algorithm 6).

Security

The safety of the MultiPrime Cloud-RSA scheme depends upon the trouble of both factoring the modulus = $p_1 \ldots p_k$, for $k \geq 2$ large and distinct primes, and discovering the decryption exponent d when the encryption exponent e is also private. The speediest factorization algorithm, NFS (Lenstra et al., 1993), is unable to capitalize on the structure of the modulus n . Nonetheless, one has to ensure that the prime factors of n do not fall in the abilities of the ECM algorithm (Lenstra, 1987).

Algorithm 5. MultiPrime Cloud-RSA key generation

INPUT: k distinct primes $p_1, \ldots, p_k$ of the same bit numbers.
OUTPUT: An evaluation key $ek = (n)$ and a private key $pk = (n, p_i, e, d_i, k_i)$ for $i = 1, \ldots, k$.

1: Set $n \leftarrow \prod_{i=1}^{k} p_i$ and $\phi(n) \leftarrow \prod_{i=1}^{k} (p_i - 1)$.
2: Pick an integer e such that $gcd(e, \phi(n)) = 1$.
for i=1 **to** k **do**
  3: Set $d_i \leftarrow e^{-1} \pmod{p_i - 1}$.
  4: Set $k_i \leftarrow \frac{n}{p_i}(\frac{n}{p_i})^{-1}$, where $k_i \equiv 1 \pmod{p_i}$.
end for
5: Return (n) and (n, p_i, e, d_i, k_i).

Algorithm 6. MultiPrime Cloud-RSA decryption

INPUT: $pk = (n, p_i, e, d_i, k_i)$ and ciphertexts $C = c_1, \ldots, c_j$.
OUTPUT: $D(pk, C) = m_1 \times \ldots \times m_j$
for i=1 **to** k **do**
  1: Set $c_i \leftarrow C \pmod{p_i}$.
  2: Set $x_i \leftarrow {c_i}^{d_i} \pmod{p_i}$.
end for
3: Compute $D(pk, C) = \sum_{i=1}^{k} x_i \times k_i \pmod{n}$.
4: Return $D(pk, C)$.

Even if an attacker succeeds in factorizing n, to reveal the decryption exponent, he/she must discover two exponents e and d that satisfies $ed \equiv 1 \pmod{\varphi(n)}$ and that the decrypted data is semantically correct.

MultiPower Cloud-RSA Scheme

We present the key generation and decryption algorithms of the MultiPower Cloud-RSA encryption scheme. We then analyze its security level.

Algorithms

The MultiPower Cloud-RSA scheme is based on the generation of a modulus $n = p^r q^s$ for $r \geq 2$ and $s \geq 1$, where the primes p and q are distinct. It is also based on the generation of three integers e, d_p and d_q satisfying the equations: $d_p \equiv e^{-1} \pmod{p-1}$ and $d_q \equiv e^{-1} \pmod{p-1}$. The modulus n is made public and represents the evaluation key. The integers e, d_p, and d_q are made private. To obtain an additional decryption speed-up, we pre-compute the Chinese remaindering coefficients and store it as a part of the private key. The MultiPower Cloud-RSA key generation process is outlined in Algorithm 7.

The MultiPower Cloud-RSA scheme keeps the same form of the Cloud-RSA and the MultiPrime Cloud-RSA algorithms (as pointed out in Algorithm 2).

The CC provider executes the MH over the ciphertexts using the evaluation key (as illustrated in Algorithm 3).

The CC consumer decrypts the return encrypted result under the MultiPower Cloud-RSA private key employing the Hensel lifting and the Chinese remaindering (as highlighted in Algorithm 8).

Security

The safety of the MultiPower Cloud-RSA scheme depends upon the hardness of factoring the modulus $n = p^r q^s$ for $r \geq 2$ and $s \geq 1$, and of the discovering the decryption exponent d when the encryption exponent e is also private.

The prime factors of n must be thoroughly chosen to prevent the speediest factorization attacks: the ECM (Lenstra, 1987) and NFS (Lenstra et al., 1993). In addition, the integers r and s should not be very large (Coron et al., 2016).

Even if an attacker succeeds in factorizing n, to reveal the decryption exponent, he/she must discover two exponents e and d that satisfies $ed \equiv 1 \pmod{\lambda(n)}$ and that the decrypted data is semantically correct.

Cloud-ElGamal Scheme

Now, we present the constituent algorithms of the Cloud-ElGamal encryption scheme. We then analyze the security level of the scheme.

Algorithm 7. MultiPower Cloud-RSA key generation

INPUT: Two distinct primes p and q each $[n/r + s]$ bit lengths.
OUTPUT: An evaluation key $ek = (n)$ and a private key $pk = (n, p, q, r, s, e, d_p, d_q, k_p, k_q)$.
1: Set $n \leftarrow pq$ and $\lambda(n) \leftarrow lcm(p-1, q-1)$.
2: Pick an integer e such that $gcd(e, \lambda(n)) = 1$.
3: Set $d_p \leftarrow e^{-1} \pmod{p-1}$ and $d_q \leftarrow e^{-1} \pmod{q-1}$.
4: Set $k_p \leftarrow \frac{n}{p^r}(\frac{n}{p^r})^{-1}$ and $k_q \leftarrow \frac{n}{q^s}(\frac{n}{q^s})^{-1}$, where $k_p \equiv 1 \pmod{p^r}$ and $k_q \equiv 1 \pmod{q^s}$.
5: Return (n) and $(n, p, q, r, s, e, d_p, d_q, k_p, k_q)$.

Algorithm 8. MultiPower Cloud-RSA decryption

INPUT: $pk = (n, p, q, r, s, e, d_p, d_q, k_p, k_q)$ and ciphertext $C = c_1, \ldots, c_j$.
OUTPUT: $D(pk, C) = m_1 \times \ldots \times m_j$
1: Set $C_0 \leftarrow C \pmod p$, $C_p \leftarrow C \pmod{p^r}$, $C_0^{'} \leftarrow C \pmod q$ and $C_q \leftarrow C \pmod{q^s}$.
2: Set $M_0 \leftarrow C_0^{d_p} \pmod p$ and $M_0^{'} \leftarrow C_0^{'d_q} \pmod q$.
3: Set $K_0 \leftarrow M_0$ and $K_0^{'} \leftarrow M_0^{'}$.
for i=1 **to** r-1 **do**
4: $E_i \leftarrow {K_{i-1}}^e \pmod{p^{i+1}}$.
5: $F_i \leftarrow C_p - E_i \pmod{p^{i+1}}$.
6: $G_i \leftarrow F_i / p^i$.
7: $M_i \leftarrow G_i \times (e{K_0}^{e-1})^{-1} \pmod p$.
8: $K_i \leftarrow K_{i-1} + p^i M_i$.
end for
9: Set $M_p \leftarrow K_{r-1}$.
for j=1 **to** s-1 **do**
10: $E_j^{'} \leftarrow {K_{j-1}^{'}}^e \pmod{q^{j+1}}$.
11: $F_j^{'} \leftarrow C_q - E_j^{'} \pmod{q^{j+1}}$.
12: $G_j^{'} \leftarrow F_j^{'} / q^j$.
13: $M_j^{'} \leftarrow G_j^{'} \times (e{K_0^{'}}^{e-1})^{-1} \pmod q$.
14: $K_j^{'} \leftarrow K_{j-1}^{'} + q^j M_j^{'}$.
end for
15: Set $M_q \leftarrow K_{s-1}^{'}$.
16: Set $D(pk, C) \leftarrow (M_p \times k_p) + (M_q \times k_q) \pmod n$.
17: Return $D(pk, C)$.

Algorithms

The Cloud-ElGamal scheme is based on the generation of a prime p and on the generation of a generator *g* of the multiplicative group *Zp**. The prime *p* is made public and represents the evaluation key. The private key is composed of the generator *g*, an integer $2 \leq a \leq p - 2$ and $\beta \equiv g^{a} \pmod p$. The Cloud-ElGamal key generation process is highlighted in Algorithm 9.

The Cloud-ElGamal scheme encrypts plaintexts under the private key as described in Algorithm 10.

The selected CC provider carries out the MH over the ciphertexts using the evaluation key (*ek*) (as illustrated in Algorithm 11).

The Cloud-ElGamal scheme decrypts the ciphertexts under the private key (*pk*) as highlighted in Algorithm 12.

Security

The security of the Cloud-ElGamal scheme relies upon the hardness of taking out discrete logarithms over finite groups. Recall that a group Z_p^*, *p* is prime, can be outfitted with more than one generator.

Algorithm 9. Cloud-ElGamal key generation

INPUT: A large prime p.
OUTPUT: An evaluation key $ek = (p)$ and a private key $pk = (p, g, a, \beta)$.
1: Find a generator g of the multiplicative group $\mathbb{Z}_p{}^*$.
2: Pick randomly an integer a such that $2 \leq a \leq p - 2$.
3: Set $\beta \leftarrow g^a \pmod p$.
4: Return (p) and (p, g, a, β).

Algorithm 10. Cloud-ElGamal encryption

INPUT: pk and plaintext $m \in \mathbb{Z}_p$.
OUTPUT: $c = E(pk, m)$.
1: Pick a random integer $b \in \{2, \ldots, p-2\}$.
2: Set $c_1 \leftarrow g^b \pmod p$ and $c_2 \leftarrow m \cdot \beta^b \pmod p$.
2: Return $E(pk, m) = (c_1, c_2)$.

Algorithm 11. Cloud-ElGamal evaluation

INPUT: ek, and ciphertexts $(c_{11}, c_{12}), \ldots, (c_{j1}, c_{j2}) \in \mathbb{Z}_p$.
OUTPUT: C as the result.
1: Compute

$$\begin{aligned} C = (C_1, C_2) &= (\prod_{i=1}^{j} c_{i1} \pmod p, \prod_{i=1}^{j} c_{i2} \pmod p) \\ &= (\prod_{i=1}^{j} g^{r_i} \pmod p, \prod_{i=1}^{j} m_i \cdot \beta^{r_i} \pmod p) \\ &= (g^{r_1+\ldots+r_j} \pmod p, (m_1 \times \ldots \times m_j)\beta^{r_1+\ldots+r_j} \pmod p) \\ &\equiv E(pk, m_1 \times \ldots \times m_j) \end{aligned}$$

2: Return C.

Algorithm 12. Cloud-ElGamal decryption

INPUT: pk and a ciphertext $C = (C_1, C_2)$.
OUTPUT: $D(pk, C) = m_1 \times \ldots \times m_j$
1: Compute $D(pk, C) = ({C_1}^{-a})C_2 \pmod p$.
2: Return $D(pk, C)$.

Since in the proposed scheme the selected generator g of Z_p^* and $\beta \equiv g^a \pmod p$ are made private and the scheme is probabilistic; therefore, no polynomial-time algorithm can recover the decryption key a from the ciphertexts or from the prime p. Hence, the Cloud-ElGamal scheme is semantically secure.

The ElGamal scheme (ElGamal, 1985) can be broken if an adversary succeeds in recovering the private key from the public key. Indeed, many polynomial-time algorithms (e.g., baby-step giant-step (Coron et al., 2005) and Index Calculus (Adleman, 1979)) can easily recover the private key when utilizing weak prime numbers. So, to achieve a high security level, the prime p of the Cloud-ElGamal scheme should not be too weak.

SIMULATION AND PERFORMANCE EVALUATION

In this section, we implement the above HE schemes. We then compare and discuss the efficiency of the schemes.

Simulation Results

The proposed HE schemes, namely the Cloud-RSA, the MultiPrime Cloud-RSA, the MultiPower Cloud-RSA, and the Cloud-ElGamal scheme have been implemented. The performance evaluation at encryption and decryption running time has been measured applying Python programming language. The implementation has been performed in a Smart-phone [CPU 600 @ 1,9 GHz and RAM @ 2 Go].

The evaluation and the private keys of the schemes have been generated utilizing PARI/GP system (Batut et al. 2000). We have set three moduli $n_1 = pq$, $n_2 = p_1p_2p_3$ and $n_3 = p_1^2p_2$ each of 1024-bit lengths. n_1, n_2 and n_3 represent the Cloud-RSA modulus, the MultiPrime Cloud-RSA modulus and the MultiPower Cloud-RSA modulus, respectively. The Cloud-RSA scheme and its variants share a common small and secure exponent e which is relatively prime to $\varphi(n_1)$, $\varphi(n_2)$ and $\lambda(n_3)$. We have also set a prime p of 1024-bit lengths. We have then generated a generator g of the multiplicative group Z_p^* over which the Cloud-ElGamal is defined.

Our implementation has offered the following tables, Table 1 and Table 2. The Cloud-RSA encryption time is as fast as its variants, since the schemes share a common encryption exponent. Table 1 depicts the required time to encrypt a quantity of data (from 100 to 1000 bits) under the Cloud-RSA and the Cloud-ElGamal schemes. Table 2 depicts the required time to decrypt the corresponding encrypted data of the plain data (from 100 to 1000 bits) under the HE schemes.

Table 1. Encryption time performance

Plaintext Bit Size	Cloud-RSA	Cloud-ElGamal
100 bits	5.09 ms	756.99 ms
200 bits	5.75 ms	741.67 ms
300 bits	6.24 ms	765.60 ms
400 bits	7.31 ms	754.78 ms
500 bits	8.05 ms	797.58 ms
600 bits	8.87 ms	749.40 ms
700 bits	9.70 ms	794.13 ms
800 bits	10.65 ms	788.37 ms
900 bits	11.05 ms	727.19 ms
1000 bits	12.24 ms	743.06 ms

Table 2. Decryption time performance

Plaintext Bit Size	Cloud-RSA	MultiPrime Cloud-RSA	MultiPower Cloud-RSA	Cloud-ElGamal
100 bits	452.09 ms	80.28 ms	63.42 ms	448.30 ms
200 bits	450.26 ms	80.52 ms	61.66 ms	451.20 ms
300 bits	455.12 ms	81.16 ms	65.17 ms	462.63 ms
400 bits	453.48 ms	79.83 ms	62.80 ms	469.70 ms
500 bits	457.81 ms	82.62 ms	64.45 ms	458.02 ms
600 bits	445.59 ms	78.40 ms	63.78 ms	446.72 ms
700 bits	459.54 ms	81.81 ms	64.76 ms	437.57 ms
800 bits	448.47 ms	79.92 ms	66.12 ms	471.86 ms
900 bits	464.40ms	83.88 ms	64.64 ms	458.13 ms
1000 bits	461.83 ms	82.34 ms	63.11 ms	443.64 ms

Comparison and Discussion

The simulation results show that the Cloud-RSA encryption process is about 89.68 times faster than that of the Cloud-ElGamal; whereas, the Cloud-ElGamal decryption time is as fast as that of the Cloud-RSA scheme. Figure 1 shows that the Cloud-RSA encryption time raises with the increase in data bit-size, and the Cloud-ElGamal encryption time is approximately stable since the time depends on bit-size of the random integers.

The MultiPrime Cloud-RSA gets a decryption speed-up by factor of 5.61 over the Cloud-RSA encryption scheme. The MultiPower Cloud-RSA can get a faster decryption; it gets an experimental decryption speed-up factor of 7.10 over the Cloud-RSA. The Chinese remaindering and the Hensel lifting steps take negligible time compared to the full exponentiation of the Cloud-RSA decryption. The decryption time of the schemes is approximately stable since the different ciphertexts have about 1024−bits long.

Counter to the Cloud-RSA scheme and its fast variants, the Cloud-ElGamal scheme achieves semantic security. However, the Cloud-ElGamal ciphertexts are two times larger than the corresponding plaintexts.

CONCLUSION

In this paper, we presented four encryption schemes support HE over the integers, namely the Cloud-RSA, the MultiPrime Cloud-RSA, the MultiPower Cloud-RSA, and the Cloud-ElGamal schemes. The schemes resist more the confidentiality attacks. The Cloud-RSA scheme and its variants, MultiPrime Cloud-RSA and MultiPower Cloud-RSA, are secure under the hardness to factor large composite integers and to discover the decryption exponents when the encryption exponents are private; whereas, the Cloud-ElGamal scheme is secure under the hardness of taking out discrete logarithms over finite groups when the selected generator of the group is private.

The Cloud-RSA's variants are designed to get a faster decryption. The simulation results show that the MultiPower Cloud-RSA scheme offers a large decryption speed-up over the three proposed HE schemes while providing a recommended security level.

ACKNOWLEDGMENT

We would like to acknowledge professor Chafik Boufrioua for the proofreading of this work and the anonymous reviewers for their useful comments.

REFERENCES

Adleman, L. 1979. A subexponential algorithm for the discrete logarithm problem with applications to cryptography. In *20th Annual Symposium on Foundations of Computer Science* (pp. 55–60). IEEE. doi:10.1109/SFCS.1979.2

Alam, M., Emmanuel, N., Khan, T., Xiang, Y., & Hassan, H. (2017). Garbled role-based access control in the cloud. *Journal of Ambient Intelligence and Humanized Computing*, 1–14.

Batut, C., Belabas, K., Bernardi, D., Cohen, H., & Olivier, M. (2000). *User's Guide to PARI-GP*. Université de Bordeaux I.

Cheon, J. H., Coron, J. S., Kim, J., Lee, M. S., Lepoint, T., Tibouchi, M., & Yun, A. 2013. Batch fully homomorphic encryption over the integers. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques* (pp. 315–335). Springer, Berlin, Heidelberg. doi:10.1007/978-3-642-38348-9_20

Coron, J. S., Faugere, J. C., Renault, G., & Zeitoun, R. (2016). Factoring N = prqs for Large r and s. In *Cryptographers' Track at the RSA Conference* (pp. 448–464). Cham: Springer.

Coron, J. S., Lefranc, D., & Poupard, G. 2005. A new baby-step giant-step algorithm and some applications to cryptanalysis. In *International Workshop on Cryptographic Hardware and Embedded Systems* (pp. 47–60). Springer. doi:10.1007/11545262_4

El Makkaoui, K., Beni-Hssane, A., & Ezzati, A. (2016). Cloud-ElGamal: An efficient homomorphic encryption scheme. In *2016 International Conference on Wireless Networks and Mobile Communications (WINCOM),* (pp. 63–66). IEEE.

El Makkaoui, K., Beni-Hssane, A., & Ezzati, A. 2017. MultiPrime Cloud-RSA Scheme to Promote Data Confidentiality in the Cloud Environment. In *Proceedings of the Mediterranean Symposium on Smart City Applications* (pp. 445–452). Cham: Springer.

El Makkaoui, K., Beni-Hssane, A., Ezzati, A., & El-Ansari, A. (2017). Fast Cloud-RSA Scheme for Promoting Data Confidentiality in the Cloud Computing. *Procedia Computer Science*, *113*, 33–40. doi:10.1016/j.procs.2017.08.282

El Makkaoui, K., Ezzati, A., & Beni-Hssane, A. 2016. Securely Adapt a Paillier Encryption Scheme to Protect the Data Confidentiality in the Cloud Environment. In *Proceedings of the International Conference on Big Data and Advanced Wireless Technologies* (p. 57). ACM. doi:10.1145/3010089.3016026

El Makkaoui, K., Ezzati, A., & Beni-Hssane, A. (2017). Cloud-RSA: An Enhanced Homomorphic Encryption Scheme. In *Europe and MENA Cooperation Advances in Information and Communication Technologies* (pp. 471–480). Cham: Springer. doi:10.1007/978-3-319-46568-5_48

El Makkaoui, K., Ezzati, A., Beni-Hssane, A., & Motamed, C. 2016. Cloud security and privacy model for providing secure cloud services. In *2016 2nd International Conference on Cloud Computing Technologies and Applications (CloudTech)* (pp. 81–86). IEEE. doi:10.1109/CloudTech.2016.7847682

ElGamal, T. (1985). A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, *31*(4), 469–472. doi:10.1109/TIT.1985.1057074

Fox, A., Griffith, R., Joseph, A., Katz, R., Konwinski, A., Lee, G., ... & Stoica, I. (2009). *Above the clouds: A berkeley view of cloud computing* (Technical Report UCB/EECS-2009-28). EECS Department, University of California, Berkeley.

Gao, H., Hu, M., Gao, T., & Cheng, R. (2018). Random grid and reversible watermarking-based on verifiable secret sharing for outsourcing images in cloud. *International Journal of Digital Crime and Forensics*, *10*(1), 24–39. doi:10.4018/IJDCF.2018010103

Gentry, C. 2009. Fully homomorphic encryption using ideal lattices. In *41st Annual ACM Symposium on Theory of Computing, STOC'09* (pp. 169–178). doi:10.1145/1536414.1536440

Hu, C., Liu, P., & Guo, S. (2016). Public key encryption secure against related-key attacks and key-leakage attacks from extractable hash proofs. *Journal of Ambient Intelligence and Humanized Computing*, *7*(5), 681–692. doi:10.1007/s12652-015-0329-0

Katz, J., Menezes, A. J., Van Oorschot, P. C., & Vanstone, S. A. (1996). *Handbook of applied cryptography*. CRC press.

Kiraz, M. S. (2016). A comprehensive meta-analysis of cryptographic security mechanisms for cloud computing. *Journal of Ambient Intelligence and Humanized Computing*, *7*(5), 731–760. doi:10.1007/s12652-016-0385-0

Lenstra, A. K., Lenstra, H. W., Manasse, M. S., & Pollard, J. M. (1993). The number field sieve. In *The development of the number field sieve* (pp. 11–42). Springer. doi:10.1007/BFb0091537

Lenstra, H. W. Jr. (1987). Factoring integers with elliptic curves. *Annals of Mathematics*, *126*(3), 649–673. doi:10.2307/1971363

Liu, J., & Ke, L. (2018). New efficient identity based encryption without pairings. *Journal of Ambient Intelligence and Humanized Computing*, 1–10.

Rivest, R. L., Adleman, L., & Dertouzos, M. L. (1978). On data banks and privacy homomorphisms. *Foundations of secure computation, 4*(11), 169–180.

Rivest, R. L., Shamir, A., & Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, *21*(2), 120–126. doi:10.1145/359340.359342

Van Dijk, M., Gentry, C., Halevi, S., & Vaikuntanathan, V. 2010. Fully homomorphic encryption over the integers. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques* (pp. 24–43). Springer.

Yan, X., Lu, Y., Liu, L., Wan, S., Ding, W., & Liu, H. (2017). Exploiting the homomorphic property of visual cryptography. *International Journal of Digital Crime and Forensics*, *9*(2), 45–56. doi:10.4018/IJDCF.2017040105

Yi, X., Paulet, R., & Bertino, E. (2014). *Homomorphic encryption and applications* (Vol. 3). Cham: Springer.

Yun, D. Y. (1974). *The Hensel lemma in algebraic manipulation (No. MAC-TR-138)*. Massachusetts Inst Of Tech., Cambridge Project Mac.

Zhou, L., Yan, W. Q., Shu, Y., & Yu, J. (2018). CVSS: A cloud-based visual surveillance system. *International Journal of Digital Crime and Forensics*, *10*(1), 79–91. doi:10.4018/IJDCF.2018010107

Khalid El Makkaoui received his Master's degree in sciences and techniques: Networks and Systems in 2014. Currently, he's pursuing his PhD in Cloud Computing Security at LAVETE Laboratory, Hassan 1st University, Faculty of Sciences and Techniques of Settat, Morocco.

Abderrahim Beni-Hssane: Is a researcher and a professor at Science Faculty, Chouaib Doukkali University, El Jadida, Morocco, since September 1994.He got his B.Sc. degree in applied mathematics and his Doctorate of High Study Degree in computer science, respectively, in 1992 and 1997 from Mohamed V University, Rabat, Morocco. His research interests focus on performance evaluation in wireless networks, Cryptography, Cloud Computing and Big Data.

Abdellah Ezzati got his PhD in computer science in 1997 from Science Faculty, Rabat, Morocco, and member of the Computer commission in the same Faculty. Now he is a professor in Hassan First University, Morocco. And he is the Head of Bachelor of Computer Science. He has participated in several works as the Palmes project which elaborates a Moroccan Education Certification. His research interests focus on performance evaluation in WSN, VANET and IoT, Security, Cloud Computing, and Big Data.