A Survey Report On Partially Homomorphic Encryption Techniques In Cloud Computing

1. S. Ramachandram 2. R.Sridevi 3. P. Srivani

1. *Professor &Vice Principal, Oucollege Of Engineering,Dept Of Cse,Hyd*
2. *Associate Professor, Jntuh, Dept Of Cse,Hyd*
3. *Research Scholar & Asso.Prof., Det Of Cse, Hitam, Hyd*

Abstract: Homomorphic encryption is a form of encryption which allows specific types of computations to be carried out on cipher text and obtain an encrypted result which decrypted matches the result of operations performed on the plaintext. As cloud computing provides different services, homomorphic encryption techniques can be used to achieve security. In this paper , We presented the partially homomorphic encryption techniques.

Keywords: Homomorphic, ElGamal, Pailler, RSA, Goldwasser–Micali , Benaloh, Okamoto Uchiyama cryptosystem, Naccache–Stern cryptosystem, RNS.

1. INTRODUCTION

Security is a desirable feature in modern system architectures. Homomorphic encryption would allow the chaining together of different services without exposing the data to each of those services. For example a chain of different services from different companies could 1) calculate the tax 2) the currency exchange rate 3) shipping, on a transaction without exposing the unencrypted data to each of those services. Homomorphic encryption schemes are malleable by design. The homomorphic property of various cryptosystems can be used to create secure voting systems, collision-resistant hash functions, private information retrieval schemes and enable widespread use of cloud computing by ensuring the confidentiality of processed data.

There are several efficient, partially homomorphic cryptosystems. Although a cryptosystem which is unintentionally homomorphic can be subject to attacks on this basis, if treated carefully, homomorphism can also be used to perform computations securely. Section 2 describes about the partially homomorphic encryption techniques.

2. Partially Homomorphic Encryption Techniques

RSA

In cryptography, RSA[1] is an asymmetric encryption system. If the RSA public key is modulus m and exponent e, then the encryption of a message x is given by $\mathcal{E}(x) = x^e \bmod m$.

The homomorphic property is then

$$\mathcal{E}(x_1) \cdot \mathcal{E}(x_2) = x_1^e x_2^e \bmod m = (x_1 x_2)^e \bmod m = \mathcal{E}(x_1 \cdot x_2)$$

ElGamal Encryption

In cryptography, the ElGamal encryption system[2] is an asymmetric encryption algorithm for public key cryptography which is based on the Diffie Helman exchange . It was described by Taher ElGamal in 1984. ElGamal encryption is used in the free GNU privacy Guard software, recent versions of PGP, and other cryptosystems. The Digital Signature Algorithm is a variant of ElGamal Signature Algorithm, which should not be confused with ElGamal encryption.

ElGamal encryption can be defined over any cyclic Group G. Its security depends upon the difficulty of a certain problem in G related to computing discrete logarithms.

Key Generation

The key generator works as follows:

- Alice generates an efficient description of a multiplicative cyclic group G of order q with generator g.
- Alice chooses a random x from $\{1, \ldots, q-1\}$.
- Alice computes $h = g^x$.
- Alice publishes h, along with the description of G, q, g, as her public key. Alice retains x as her private key which must be kept secret.

Encryption

The encryption algorithm works as follows: to encrypt a message m to Alice under her public key (G, q, g, h),

- Bob chooses a random y from $\{1, \ldots, q-1\}$, then calculates $c_1 = g^y$.
- Bob calculates the shared secret $s = h^y$.
- Bob converts his secret message m into an element m' of G.
- Bob calculates $c_2 = m' \cdot s$.
- Bob sends the cipher text $(c_1, c_2) = (g^y, m' \cdot h^y) = (g^y, m' \cdot (g^x)^y)$ to Alice.

Note that one can easily find h^y if one knows m'. Therefore, a new y is generated for every message to improve security. For this reason, y is also called an ephemeral key.

Decryption

The decryption algorithm works as follows: to decrypt a cipher ext (c_1, c_2) with her private key x,

- Alice calculates the shared secret $s = c_1^x$
- and then computes $m' = c_2 \cdot s^{-1}$ which she then converts back into the plaintext message m, where s^{-1} is inverse of s In the group G. (E.g. modular multiplicative inverse if G is a subgroup of a multiplication group of integers modulo n).

The decryption algorithm produces the intended message, since

$$c_2 \cdot s^{-1} = m' \cdot h^y \cdot (g^{xy})^{-1} = m' \cdot g^{xy} \cdot g^{-xy} = m'.$$

Efficiency

ElGamal encryption is probabilistic meaning that a single plain text can be encrypted to many possible cipher texts, with the consequence that a general ElGamal encryption produces a 2:1

expansion in size from plaintext to cipher text.

Encryption under ElGamal requires two exponentiation. However, these exponentiations are independent of the message and can be computed ahead of time if need be. Decryption only requires one exponentiation.

The division by $s$can be avoided by using an alternative method for decryption. To decrypt a cipher text $(c_1, c_2)$with Alice's private key x,

- Alice calculates $s' = c_1^{q-x} = g^{(q-x)y}$.

$s'$is the inverse of s. This is a consequence of Langranges Theorem, because

$$s \cdot s' = g^{xy} \cdot g^{y(q-x)} = (g^q)^y = 1^y = 1.$$

- Alice then computes $m' = c_2 \cdot s'$, which she then converts back into the plaintext message m.

The decryption algorithm produces the intended message, since

$$c_2 \cdot s' = m' \cdot s \cdot s' = m' \cdot 1 = m'.$$

In the ElGamal cryptosystem, in a group G, if the public key is (G, q, g, h), where $h = g^x$, and $x$is the secret key, then the encryption of a message $m$is $\mathcal{E}(m) = (g^r, m \cdot h^r)$, for some random $r \in \{0, \ldots, q-1\}$. The homomorphic property is then

$$\mathcal{E}(x_1) \cdot \mathcal{E}(x_2) = (g^{r_1}, x_1 \cdot h^{r_1})(g^{r_2}, x_2 \cdot h^{r_2}) = (g^{r_1+r_2}, (x_1 \cdot x_2) h^{r_1+r_2}) =$$

Goldwasser–Micali

The Goldwasser–Micali (GM) crypto system[3] is an asymmetric key encryption algorithm developed by Shaff Goldwasser and Silvio Micali in 1982. GM has the distinction of being the first probabilistic public-key encryption scheme which is provably secure under standard cryptographic assumptions. However, it is not an efficient cryptosystem, as cipher texts may be several hundred times larger than the initial plaintext. To prove the security properties of the cryptosystem, Goldwasser and Micali proposed the widely used definition of semantic security.

Because encryption is performed using a probabilistic algorithm, a given plaintext may produce very different cipher texts each time it is encrypted. This has significant advantages, as it prevents an adversary from recognizing intercepted messages by comparing them to a dictionary of known cipher texts.

The scheme relies on deciding whether a given value *x* is a square mod *N*, given the factorization (*p*, *q*) of *N*. This can be accomplished using the following procedure:

1. Compute $x_p = x \bmod p$, $x_q = x \bmod q$.
2. If $x_p^{(p-1)/2} \equiv 1 \pmod{p}$and $x_q^{(q-1)/2} \equiv 1 \pmod{q}$, then *x* is a quadratic residue mod *N*.

Key Generation

Alice generates two distinct large prime numbers *p* and *q*, randomly and independently of each other.

1. Alice computes $N = p\,q$.
2. She then finds some non-residue x such that the legendra symbols satisfy $\left(\frac{x}{p}\right) = \left(\frac{x}{q}\right) = -1$ and hence the Jacobi symbol $\left(\frac{x}{N}\right)$ is +1. The value x can for example be found by selecting random values and testing the two Legendre symbols. If $(p, q) = 3 \bmod 4$ (i.e., N is a Blum integer), then the value $N - 1$ is guaranteed to have the required property.

The *public key* consists of (x, N). The secret key is the factorization (p, q).

Message encryption

Suppose Bob wishes to send a message m to Alice:

1. Bob first encodes m as a string of bits $(m_1, ., m_n)$.
2. For every bit m_i, Bob generates a random value y_i from the group of units modulo N, or $\gcd(y_i, N) = 1$. He outputs the value $c_i = y_i^2 x^{m_i} \pmod N$.

Bob sends the cipher text $(c_1, .. , c_n)$.

Message Decryption

Alice receives $(c_1, ..., c_n)$. She can recover m using the following procedure:

1. For each i, using the prime factorization (p, q), Alice determines whether the value c_i is a quadratic residue; if so, $m_i = 0$, otherwise $m_i = 1$.

Alice outputs the message $m = (m_1, ..., m_n)$.

In the Goldwasser micali cryptosystem , if the public key is the modulus m and quadratic non-residue x, then the encryption of a bit b is $\mathcal{E}(b) = x^b r^2 \bmod m$, for some random $r \in \{0, \ldots, m-1\}$. The homomorphic property is then $\mathcal{E}(b_1) \cdot \mathcal{E}(b_2) = x^{b_1} r_1^2 x^{b_2} r_2^2 = x^{b_1+b_2}(r_1 r_2)^2 = \mathcal{E}(b_1 \oplus b_2)$

where $\oplus$ denotes addition modulo 2.

Benaloh Cryptosystem

The **Benaloh Cryptosystem[4]** is an extension of the Goldwasser micali cryptosystem(GM) created in 1994 by Josh (Cohen) Benaloh. The main improvement of the Benaloh Cryptosystem over GM is that longer blocks of data can be encrypted at once, whereas in GM each bit is encrypted individually. Like many public key cryptosystem, this scheme works in the group $(\mathbb{Z}/n\mathbb{Z})^*$ where n is a product of two large primes. This scheme is homomorphic and hence malleable.

Key Generation

A public/private key pair is generated as follows:

- Choose a block size r.
- Choose large primes p and q such that r divides $(p\text{-}1)$, $\gcd(r, (p\text{-}1)/r) = 1$ and $\gcd(q\text{-}1, r) = 1$.
- Set $n = pq$
- Choose $y \in (\mathbb{Z}/n\mathbb{Z})^*$ such that $y^{(p-1)(q-1)/r} \not\equiv 1 \mod n$.

The public key is then y, n, and the private key is the two primes p, q.

Message encryption

To encrypt a message m, where m is taken to be an element in $\mathbb{Z}/r\mathbb{Z}$

- Choose a random $u \in (\mathbb{Z}/n\mathbb{Z})^*$
- Set $E_r(m) = y^m u^r \mod n$

Message decryption

$$(y^m u^r)^{(p-1)(q-1)/r} \equiv y^{m(p-1)(q-1)/r} u^{(p-1)(q-1)} \equiv y^{m(p-1)(q-1)/}$$

To understand decryption, we first notice that for any *m*, *u* we have

Since $m < r$ and $y^{(p-1)(q-1)/r} \not\equiv 1 \mod n$, we can conclude that $(y^m u^r)^{(p-1)(q-1)/r} \equiv 1 \mod n$ if and only if *m* = 0. So if $z = y^m u^r \mod n$ is an encryption of *m*, given the secret key *p*, *q* we can determine whether *m*=0. If *r* is small, we can decrypt *z* by doing an exhaustive search, i.e. decrypting the messages y^{-I}z for *i* from 1 to *r*. By pre computing values, using the Baby step Gaint step algorithm, decryption can be done in time $O(\sqrt{r})$.

Security

The security of this scheme rests on the Higher residuosity problem specifically, given *z*, *r* and *n* where the factorization of *n* is unknown, it is computationally infeasible to determine whether *z* is an *r* th residue mod *n*, i.e. if there exists an *x* such that $z \equiv x^r \mod n$.

In the Benaloh cryptosystem if the public key is the modulus *m* and the base *g* with a blocksize of *c*, then the encryption of a message *x* is $\mathcal{E}(x) = g^x r^c \bmod m$, for some random $r \in \{0, \ldots, m-1\}$. The homomorphic property is then

$$\mathcal{E}(x_1) \cdot \mathcal{E}(x_2) = (g^{x_1} r_1^c)(g^{x_2} r_2^c) = g^{x_1+x_2}(r_1 r_2)^c = \mathcal{E}(x_1 + x_2$$

Pailler cryptosystem

The Paillier crypto system[5], named after and invented by pascal pailler in 1999, is a probabilistic asymmetric algorithm for public key cryptography. The problem of computing n-th residue classes is believed to be computationally difficult. The decisional composite residuosity assumption is the intractability hypothesis upon which this cryptosystem is based.

The scheme is an additive homomorphic cryptosystem; this means that, given only the public-key and the encryption of m_1 and m_2, one can compute the encryption of $m_1 + m_2$.

Key Generation

1. Choose two large prime numbers *p* and *q* randomly and independently of each other such that $\gcd(pq, (p-1)(q-1)) = 1$.
   This property is assured if both primes are of equivalent length, i.e., $p, q \in 1||\{0,1\}^{s-1}$ for security parameter s.
2. Compute $n = pq$ and $\lambda = \operatorname{lcm}(p-1, q-1)$.
3. Select random integer g where $g \in \mathbb{Z}_{n^2}^*$
4. Ensure n divides the order of g by checking the existence of the following modular multiplicative inverse
   $\mu = (L(g^\lambda \mod n^2))^{-1} \mod n$
   , where function L is defined

   as $L(u) = \frac{u-1}{n}$.
   Note that the notation a/b does not denote the modular multiplication of a times the modular multiplicative inverse of b but rather the quotient of

$a$divided by b, i.e., the largest integer value $v \geq 0$ to satisfy the relation $a \geq vb$.

- The public (encryption) key is (n, g).
- The private (decryption) key is (λ, μ).

If using p, q of equivalent length, a simpler variant of the above key generation steps would be to set $g = n + 1, \lambda = \varphi(n)$, and $\mu = \varphi(n)^{-1} \mod n$, where $\varphi(n) = (p-1)(q-1)$.

Encryption

1. Let m be a message to be encrypted where $m \in \mathbb{Z}_n$
2. Select random $r$where $r \in \mathbb{Z}_n^*$
3. Compute cipher text as: $c = g^m \cdot r^n \mod n^2$

Decryption

1. Cipher text $c \in \mathbb{Z}_{n^2}^*$
2. Compute message: $m = L(c^\lambda \mod n^2) \cdot \mu \mod n$

Homomorphic properties

A notable feature of the Paillier cryptosystem is its homomorphic properties. As the encryption function is additively homomorphic, the following identities can be described:

- **Homomorphic addition of plaintexts**

  The product of two cipher texts will decrypt to the sum of their corresponding plaintexts,

$$D(E(m_1, r_1) \cdot E(m_2, r_2) \mod n^2) = m_1 + m_2 \mod n.$$

  The product of a cipher text with a plaintext raising g will decrypt to the sum of the corresponding plaintexts,

$$D(E(m_1, r_1) \cdot g^{m_2} \mod n^2) = m_1 + m_2 \mod n.$$

- **Homomorphic multiplication of plaintexts**

  An encrypted plaintext raised to the power of another plaintext will decrypt to the product of the two plaintexts,

$$D(E(m_1, r_1)^{m_2} \mod n^2) = m_1 m_2 \mod n,$$

$$D(E(m_2, r_2)^{m_1} \mod n^2) = m_1 m_2 \mod n.$$

  More generally, an encrypted plaintext raised to a constant *k* will decrypt to the product of the plaintext and the constant,

$$D(E(m_1, r_1)^k \mod n^2) = km_1 \mod n.$$

However, given the Paillier encryptions of two messages there is no known way to compute an encryption of the product of these messages without knowing the private key.

Okamoto–Uchiyama cryptosystem

The Okamoto–Uchiyama crypto system[6] was discovered in 1998 by T. Okamoto and S. Uchiyama. The system works in the group $(\mathbb{Z}/n\mathbb{Z})^*$, where *n* is of the form p^2q and *p* and *q* are large primes.

Key Generation

A public/private key pair is generated as follows:

- Generate large primes p and q and set $n = p^2q$.
- Choose $g \in (\mathbb{Z}/n\mathbb{Z})^*$ such that $g^p \neq 1 \mod p^2$.
- Let $h = g^n \mod n$.

The public key is then (n, g, h) and the private key is the factors (p, q).

Message Encryption

To encrypt a message m, where m is taken to be an element in $\mathbb{Z}/n\mathbb{Z}$

- Select $r \in \mathbb{Z}/n\mathbb{Z}$ at random. Set

$$C = g^m h^r \mod n$$

Message Decryption

If we define $L(x) = \frac{x-1}{p}$, then decryption becomes

$$m = \frac{L\left(C^{p-1} \mod p^2\right)}{L\left(g^{p-1} \mod p^2\right)} \mod n$$

How the system works

The group $(\mathbb{Z}/n\mathbb{Z})^* \simeq (\mathbb{Z}/p^2\mathbb{Z})^* \times (\mathbb{Z}/q\mathbb{Z})^*$

The group $(\mathbb{Z}/p^2\mathbb{Z})^*$ has a unique subgroup of order p, call it H. By the uniqueness of H, we must have

$$H = \{x : x \equiv 1 \mod p\}.$$

For any element x in $(\mathbb{Z}/p^2\mathbb{Z})^*$, we have $x^{p-1} \mod p^2$ is in H, since p divides $x^{p-1} - 1$.

The map L should be thought of as a logarithm from the cyclic group H to the additive group $\mathbb{Z}/p\mathbb{Z}$, and it is easy to check that $L(ab) = L(a) + L(b)$, and that the L is an isomorphism between these two groups. As is the case with the usual logarithm, $L(x)/L(g)$ is, in a sense, the logarithm of x with base g.

We have

$$(g^m h^r)^{p-1} = (g^m g^{nr})^{p-1} = (g^{p-1})^m g^{p(p-1)rpq} = (g^{p-1})^m \mod p^2.$$

So to recover m we just need to take the logarithm with base g^{p-1}, which is accomplished by

$$\frac{L\left((g^{p-1})^m\right)}{L(g^{p-1})} = m \mod p.$$

Naccache–Stern cryptosystem.

The **Naccache–Stern crypto system[7]** is a homomorphic public key encryption whose security rests on the higher residuosity problem. The Naccache–Stern cryptosystem was discovered by David Naccache and Jacques Stern in 1998. Like many public key cryptosystem, this scheme works in the group $(\mathbb{Z}/n\mathbb{Z})^*$ where n is a product of two large primes.. This scheme is homomorphic and hence malleable.

Key Generation

- Pick a family of k small distinct primes $p_1,...,p_k$.
- Divide the set in half and set $u = \prod_{i=1}^{k/2} p_i \quad v = \prod_{k/2+1}^{k} p_i$.
- Set $\sigma = uv = \prod_{i=1}^{k} p_i$
- Choose large primes a and b such that both $p = 2au+1$ and $q=2bv+1$ are prime.
- Set $n=pq$.

- Choose a random *g* mod *n* such that *g* has order φ(*n*)/4.

The public key is the numbers σ, *n*, *g* and the private key is the pair *p*, *q*.

When *k*=1 this is essentially the Benaloh cryptosystem.

Message encryption

This system allows encryption of a message *m* in the group $\mathbb{Z}/\sigma\mathbb{Z}$.

- Pick a random $x \in \mathbb{Z}/n\mathbb{Z}$.
- Calculate $E(m) = x^{\sigma} g^{m} \mod n$

Then *E(m)* is an encryption of the message *m*.

Message decryption

To decrypt, we first find *m* mod p_i for each *i*, and then we apply the Chinese remainder theorem to calculate *m* mod σ.

Given a cipher text *c*, to decrypt, we calculate

$c_i \equiv c^{\phi(n)/p_i} \mod n$. Thus

$$\begin{aligned} c^{\phi(n)/p_i} &\equiv x^{\sigma\phi(n)/p_i} g^{m\phi(n)/p_i} \mod \\ &\equiv g^{(m_i + y_i p_i)\phi(n)/p_i} \mod \\ &\equiv g^{m_i \phi(n)/p_i} \mod n \end{aligned}$$

where $m_i \equiv m \mod p_i$.

Since p_i is chosen to be small, m_i can be recovered be exhaustive search, i.e. by comparing c_i to $g^{j\phi(n)/p_i}$ for *j* from 1 to p_i-1.

Once m_i is known for each *i*, *m* can be recovered by a direct application of the Chinese remainder theorem.

Damg ard–Jurik cryptosystem

The Damg ard–Jurik crypto system[8] is a generalization of the Pailler cryptosystem.. It uses computations modulo n^{s+1} where n is an RSA modulus and s a (positive) natural number. Paillier's scheme is the special case with $s = 1$. The order $\varphi(n^{s+1})$ of $Z^*_{n^{s+1}}$ can be divided by n^s. Moreover $Z^*_{n^{s+1}}$ can be written as the direct product of $G \times H$. G is cyclic and of order n^s, while H is isomorphic to Z^*_n. For encryption, the message is transformed into the corresponding coset of the factor Group G/H and the security of the scheme relies on the difficulty of distinguishing random elements in different cosets of H. It is semantically secure if it is hard to decide if two given elements are in the same coset. Like Paillier, the security of Damgård–Jurik can be proven under the decisional composite residuosity assumption.

Key generation

1. Choose two large prime numbers *p* and *q* randomly and independently of each other.
2. Compute $n = pq$ and $\lambda = \operatorname{lcm}(p-1, q-1)$.
3. Choose an element $g \in \mathbb{Z}^*_{n^{s+1}}$ such that $g = (1+n)^j x \mod n^{s+1}$ for a known j relative prime to n and $x \in H$.
4. Using the chinese remainder theorem , choose d such that $d \mod n \in \mathbb{Z}^*_n$ and $d = 0 \mod \lambda$. For instance d could be λ as in Paillier's original scheme.

- The public (encryption) key is (n, g).
- The private (decryption) key is d.

Encryption

Let m be a message to be encrypted where $m \in \mathbb{Z}_{n^s}$.

1. Select random r where $r \in \mathbb{Z}^*_{n^{s+1}}$.
2. Compute cipher text as: $c = g^m \cdot r^{n^s} \mod n^{s+1}$.

Decryption

1. Cipher text $c \in \mathbb{Z}^*_{n^{s+1}}$
2. Compute $c^d \ mod \ n^{s+1}$. If *c* is a valid cipher text then $c^d = (g^m r^{n^s})^d = ((1+n)^{jm} x^m r^{n^s})^d$
3. Apply a recursive version of the pailler decryption mechanism to obtain jmd. As jd is known, it is possible to compute $m = (jmd) \cdot (jd)^{-1} \ mod \ n^s$.

Residue Number System

RNS is homomorphic encryption[9] technique which can be used to achieve security. A residue number system is defined by a set of *N* integer constants,

$\{m_1, m_2, m_3, \ldots, m_N\}$,

referred to as the *moduli*. Let *M* be the least common multiple of all the m_i.

Encryption

Any arbitrary integer *X* smaller than *M* can be represented in the defined residue number system as a set of *N* smaller integers

$\{x_1, x_2, x_3, \ldots, x_N\}$

with

$x_i = X$ modulo m_i

Decryption

Chinese Remainder theorem can be used to decrypt the number represented in Residue Number System. The Chinese Remainder Theorem is the theorem that enables the conversion of residues back into more traditional form such as Decimal form. With this theorem, for residues ***{r₁, r₂, ..., rₙ}*** of a number ***x***, can be converted back into ***x***, provided the greatest common divisor of any pair of moduli is 1. The theorem can be expressed as:

$$x = \sum_{i=1}^{N} A_i T_i r_i \bmod M$$

Where, r_i is the residue, A_i is $\frac{M}{m_i}$, where ***M*** is the product of moduli and T_i is the multiplicative inverse over the ring modulo m_i, popularly written as m_i^{-1}.

4. Conclusion

This paper presents a partially homomorphic encryption techniques used in cloud computing to achieve security.

5. References

1. http://en.wikipedia.org/wiki/RSA

2.http://en.wikipedia.org/wiki/ElGamal_encryption

3.http://en.wikipedia.org/wiki/Goldwasser%E2%80%93Micali_cryptosystem

4.http://en.wikipedia.org/wiki/Benaloh_cryptosystem

5.http://en.wikipedia.org/wiki/Paillier_cryptosystem

6.http://en.wikipedia.org/wiki/Okamoto%E2%80%93Uchiyama_cryptosystem

7.http://en.wikipedia.org/wiki/Naccache%E2%80%93Stern_cryptosystem

8.http://en.wikipedia.org/wiki/Damg%C3%A5rd%E2%80%93Jurik_cryptosystem

9.http://icsd.i2r.a-star.edu.sg/staff/sethome/pdf/004.pdf