International Journal of Applied Engineering Research ISSN 0973-4562 Volume 10, Number 14 (2015) pp 34265-34269

A general analysis of homomorphic cryptosystems

M. Jayanthi1 and Dr. Kannan Balasubramanian2
[1]*Department of Computer Science and Engineering,*
MepcoSchlenk Engineering College, India
E-mail: jayanthimathanan@gmail.com
[2]*Department of Computer Science and Engineering, Mepco Schlenk Engineering College, India*
E-mail: kannanb6@gmail.com

Abstract- *Homomorphic cryptosystems are ones where mathematical operations on the ciphertext have regular effects on the plaintext. The Encryption method called Homomorphic Encryption scheme satisfies a few properties that enable us to perform operations on encrypted data. The definition of Homomorphic Encryption and some examples of Cryptosystems satisfy the property. It is followed by the discussion of some applications of the homomorphic cryptosystems.*

Kewords- Homomorphic, encryption, cryptosystems, paillier, partially homomorphic,

1. Introduction

Homomorphic encryption is a technique that enables mathematical operations to be performed on encrypted data. Homomorphic encryption is a form of encryption which allows specific types of computations to be carried out on ciphertext and generate an encrypted result which, when decrypted, matches the result of operations performed on the plaintext. With most encryption schemes, the encrypted data has to be decrypted entirely before any significant work. e.g., math or programming operations can be done on it. It performs math directly on the encrypted data and have the results of that math show in the underlying data [1].

An encryption scheme that allows meaningful computation on encrypted data, namely a homomorphic encryption scheme[11]. Homomorphic encryption schemes are malleable by design. Homomorphic encryption is the conversion of data into ciphertext that can be analyzed and worked with as if it were still in its original form. Homomorphic encryptions allow complex mathematical operations to be performed on encrypted data without compromising the encryption. In mathematics, homomorphic describes the transformation of one data set into another while preserving relationships between elements in both sets. Because the data in a homomorphic encryption scheme retains the same structure, identical mathematical operations whether they are performed on encrypted or decrypted data will yield equivalent results.

Suppose you want to add two numbers that are stored in an encrypted file. Traditionally the only way to do it was to decrypt the file, add the two numbers and then re-encrypt the file. Of course, to do the addition you had to have access to the entire contents. This also meant that the people could access it while it was stored as plain text. There are lots of of situations where it [2] would be good if the data could be stored in encrypted form, say in the cloud, and still operated on without having to decrypt it. This is the goal of fully homomorphic encryption [3].

Operations: A homomorphic encryption is additive if $\epsilon(x + y) = \epsilon(x) \oplus \epsilon(y)$, where ϵ denotes an encryption scheme, $\oplus$ denotes an operation depending on the used cipher and x and y are plaintext messgages. A homomorphic encryption is multiplicative if $\epsilon(x. y) = \epsilon(x) \oplus \epsilon(y)$, where ϵ denotes an encryption scheme, $\oplus$ denotes an operation depending on the used cipher and x and y are plaintext messgages.

It's a crypto system that allows another party to perform operations on your ciphertext without having knowledge of your secret key. For example, it would allow a service to add two encrypted numbers and return the result without ever decrypting the two numbers.

So if E(x) represents the encryption of the number x with my secret key, another party would be able to compute $E(x_1 + x_2)$ if one were to pass them $E(x_1)$ and $E(x_2)$ using a homomorphic cryptosystem[5].

2. Different Variants of Homomorphic Schemes: Partially Homomorphic Cryptosystems:

In the following examples, the notation $\epsilon(x)$ is used to denote the encryption of the message x.

2.1 Unpadded RSA

If the RSA public key is modulus m and exponent e, then the encryption of a message x is given by $\epsilon(x) = x^e \bmod m$. The homomorphic property is then [3]

$\epsilon(x_1).\ \epsilon(x_2) = x_1^e\, x_2^e \bmod m = (x_1\, x_2)^e \bmod m = \epsilon(x_1.\ x_2)$

2.2 Elgamal

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 $\epsilon(m) = (g^r, m.h^r)$, for some random $r \in \{0, \ldots, q\text{-}1\}$. The homomorphic property is then [15]

$\epsilon(b_1) = (g^r, b_1.h^r),\ \epsilon(b_2) = (g^r, b_2.h^r)$

$\epsilon(b_1).\ \epsilon(b_2) = x^{b1} r_1^2 x^{b2} r_2^2 = x^{b1+b2}(r_1 r_2)^2 = \epsilon(b_1 \oplus b_2)$

where $\oplus$ denotes addition modulo 2, (i.e. exclusive-or). One of its main advantages is that it is simple, natural and efficient; Its security is clearly understood. Its disadvantage is that this scheme requires the encoding of messages into group elements, in order to be semantically secure [12].

2.3 Paillier

In the Paillier cryptosystem, if the public key is the modulus m and the base g, then the encryption of a message x is $\epsilon(x) = (g^x r^m \bmod m^2)$, for some random $r \in \{0,,\ldots, m\text{-}1\}$. The homomorphic property is then [13]

$\epsilon(x_1) = (g^{x1} r^m \bmod m^2)$

$\epsilon(x_2) = (g^{x2} r^m \bmod m^2)$

$\epsilon(x_1).\epsilon(x_2) = (g^{x1} r_1^m)(g^{x2} r_2^m) = g^{x1+x2}(r_1 r_2)^m = \epsilon(x_1 + x_2 \bmod c)$

2.4 Okamoto–Uchiyama System

The system works in the group $(Z / nZ)^*$, where n is of the form p^2q and p and q are large primes. Like many public key cryptosystems, this scheme works in the group **$(Z / nZ)^*$**. A fundamental difference of this cryptosystem is that here n is of the form p^2q, where p and q are large primes [4]. This scheme is homomorphic and hence malleable.

Key generation

A public/private key pair is generated as follows:

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

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 Z / nZ

- Select $r \in Z / nZ$ at random. Set
- $C = (g^m h^r \bmod n)$

Message decryption

$$\frac{x-1}{p}$$

If we define $L(x) = \frac{x-1}{p}$, then decryption becomes $m = L(C^{p-1} \bmod p^2) / L(g^{p-1} \bmod p^2) \bmod p$

2.5 Naccache–Stern Cryptosystem

Like many public key cryptosystems, this scheme works in the group *(Z / nZ)* 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 $p_1, \ldots, p_k$.
- Divide the set in half and set $u = \prod_{i=1}^{k/2} p_i$ and $v = \prod_{k/2+1}^{k} p_i$.
- 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 $\varphi(n)/4$.

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 Z/ σ Z.

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

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 ciphertext c, to decrypt, we calculate

$Ci \equiv C^{\phi(n)/Pi} \bmod n$. Thus

$C^{\phi(n)/Pi} \equiv X^{\sigma \phi(n)/Pi} g^{m \phi(n)/Pi} \bmod n$

$\equiv g^{(mi+yiPi)\phi(n)/Pi} \bmod n$

$\equiv g^{(mi\phi(n)/Pi} \bmod n$

where $m_i \equiv m \bmod p_i$.

- Since p_i is chosen to be small, m_i can be recovered by exhaustive search, i.e. by comparing Ci to $g^{j\phi(n)/Pi}$ 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.7

2.6 Damgard–Jurik Cryptosystem

It is a generalization of the Paillier cryptosystem.

Key generation

1. Choose two large prime numbers p and q randomly and independently of each other.
2. Compute $n = pq$ and $\lambda = \text{lcm}(p\text{-}1, q\text{-}1)$. [4]
3. Choose an element $g \epsilon Z^*_{n\, s+1}$ such that $g=(1+n)^j x \bmod n^{s+1}$ for a known j relative prime to n and $x \epsilon H$.
4. Using the Chinese Remainder Theorem, choose d such that $d \bmod n \epsilon Z^*_n$ and $d = 0 \bmod \lambda$. 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

1. Let $\boldsymbol{m}$ be a message to be encrypted where $\boldsymbol{m} \epsilon Z^*_n$.
2. Select random r where $r \epsilon Z^*_{n\, s+1}$.
3. Compute ciphertext as: $c = g^m . r^{ns} \bmod n^{s+1}$.

Decryption

1. Ciphertext $v \epsilon Z^*_{n\, s+1}$
2. Compute $C^d \bmod n^{s+1}$c. If C is a valid ciphertext then $C^d=(g^m r^{ns})^d=((1+n)^{jm} x^m r^{ns})^d=(1+n)^{jmd \bmod ns}(x^m r^{ns})^{d \bmod \lambda}=(1+n)^{jmd \bmod ns}$
3. Apply a recursive version of the Paillier decryption mechanism to obtain jmd. As jd is known, it is possible to compute $m = (jmd).(jd)^{-1} \bmod n^s$

2.7 Castagnos Scheme:
Castagnos explored the possibility of improving the performance of homomorphic encryption schemes using quadratic fields quotations [7]. This scheme achieves an expansion value of 3 and the ratio of encryption/decryption cost with s=1 over Paillier's scheme can be estimated to be about 2[6].

2.8 Galbraith Scheme:
This is an adaptation of the existing homomorphic encryption schemes in the context of elliptic curves [6]. The most important advantage of this scheme is that the cost of encryption and decryption can be decreased using larger values of s. In addition, the security of the scheme increases with the increase in the value of s as it is the case in Damgard-Jurik's scheme. Its expansion is equal to 3. For s = l the ratio of the encryption cost for this scheme over that of Paillier's scheme can be estimated to be about 7, while the same ratio for the cost of decryption cost is about 14 for the same value of s.

3.Efficient Encoding of Integers For Arithmetic Operations:

Given a list of integers $(m_1,\ldots,m_l) \epsilon Z^l$. To compute their sum or product over the integers homomorphically, the choice is to encrypt them directly. For every *m* in the list, compute

$Enc(pk,m) = (c_0,c_1)= (a_0u + tg + m, a_1u + tf)$

Obtain $\sum_i m_i$ over the integers. Break each *m* into bits $(m^{(0)},\ldots,m^{(n-1)})$, create a degree (n-1) polynomial $pm(x)= \sum_j m_i^{(j)} x^j$ and encrypt m as $Enc(pk,m)= (C_0,C_1)= (a_0u + tg + pm, a_1u + tf)$. The result is simply $pm_{add}(x)= \sum_i pm_i(x)$. Homomorphic cryptography offers a similar pair of pathways. We can do arithmetic directly on the plaintext inputs ***x*** and ***y***. Or we can encrypt ***x*** and ***y***, apply a series of operations to the ciphertext values, then decrypt the result to arrive at the same final answer. The two routes pass through parallel universes: plainspace and cipherspace. A number is conveniently represented as a sequence of bits (binary digits 0 and 1) and algorithms act on the bits according to rules of logic and arithmetic.

Among the many operations on numbers addition and multiplication is used. Doing mathematics in cipherspace is much stranger. Encryption is a process that thoroughly scrambles the bits of a number, whereas algorithms for arithmetic are extremely finicky and give correct results only if all the bits are in the right places [9]. Homomorphic addition takes milliseconds, multiplication generally less than a second. These timings are a vast improvement over earlier efforts. The operations on the objects are addition and multiplication. Plaintext integers are encrypted by doubling; then any sequence of additions and multiplications can be carried out; finally the result is decrypted by halving.

4. Applications of Homomorphic Encryption

Homomorphic encryption is one of the most exciting new research topics in cryptography, which promises to make cloud computing perfectly secure. With it, a Web user would send encrypted data to a server in the cloud, which would process it without decrypting it and send back a still-encrypted result. a user can store encrypted data on a server, and allow the server to process the encrypted data without revealing the data to the server.

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.

4.1 Protection of Mobile Agents:
One of the most interesting applications of homomorphic encryption is its use in protection of mobile agents. Since all conventional computer architectures are based on binary strings and only require multiplication and addition, such homomorphic cryptosystems would offer the possibility to encrypt a whole program so that it is still executable. Hence, it could be used to protect mobile agents against malicious hosts by encrypting them [8].

The protection of mobile agents by homomorphic encryption can be used in two ways: (i) computing with encrypted functions and (ii) computing with encrypted data. Computation with encrypted functions is a special case of protection of mobile agents. In such scenarios, a secret function is publicly evaluated in such a way that the function remains secret. Using homomorphic cryptosystems the encrypted function can be evaluated which guarantees its privacy. Homomorphic schemes also work on encrypted data to compute publicly while maintaining the privacy of the secret data. This can be done encrypting the data in advance and then exploiting the homomorphic property to compute with encrypted data [16].

4.2 Multiparty Computation:
In multiparty computation schemes [10], several parties are interested in computing a common, public function on their inputs while keeping their individual inputs private. This problem belongs to the area of computing with encrypted data. Usually in multiparty computation protocols, we have a set of $n \geq 2$ players whereas in computing with encrypted data scenarios n=2. Furthermore, in multi-party computation protocols, the function that should be computed is publicly known, whereas in the area of computing with encrypted data it is a private input of one party.

4.3 Secret Sharing Scheme:
In secret sharing schemes, parties share a secret so that no individual party can reconstruct the secret form the information available to it. However, if some parties cooperate with each other, they may be able to reconstruct the secret. In this scenario, the homomorphic property implies that the composition of the shares of the secret is equivalent to the shares of the composition of the secrets.

4.4 Threshold Schemes:
Both secret sharing schemes and the multiparty computation schemes are examples of threshold schemes. Threshold schemes can be implemented using homomorphic encryption techniques.

4.5 Zero-Knowledge Proofs:
This is a fundamental primitive of cryptographic protocols and serves as an example of a theoretical application of homomorphic cryptosystems. Zero knowledge proofs are used to prove knowledge of some private information. For instance, consider the case where a user has to prove his identity to a host by logging in with her account and private password. The user wants her private information (i.e., her password) to stay private and not to be leaked during the protocol operation. Zero-knowledge proofs guarantee that the protocol communicates exactly the knowledge that was intended, and no (zero) extra knowledge.

4.6 Election Schemes
In election schemes, the homomorphic property provides a tool to obtain the tally given the encrypted votes without decrypting the individual votes. Watermarking and fingerprinting schemes: Digital watermarking and fingerprinting schemes embed additional information into digital data. The homomorphic property is used to add a mark to previously encrypted data. In general, watermarks are used to identify the owner/seller of digital goods to ensure the copyright.

4.7 Oblivious Transfer:
It is an interesting cryptographic primitive. Usually in a two-party 1-out-of-2 oblivious transfer protocol, the first party sends a bit to the second party in such as way that the second party receives it with probability ½, without the first party knowing whether or not the second party received the bit.

4.8 Commitment Schemes:
Commitment schemes are some fundamental cryptographic primitives. In a commitment scheme, a player makes a commitment. She is able to choose a value from some set and commit to her choice such that she can no longer change her mind. She does not have to reveal her choice although she may do so at some point later. Some commitment schemes can be efficiently implemented using homomorphic property.

4.9 Lottery Protocols:
Usually in a cryptographic lottery, a number pointing to the winning ticket has to be jointly and randomly chosen by all participants. Using a homomorphic encryption scheme this can be realized as follows: Each player chooses a random number which she encrypts. Then using the homomorphic property the encryption of the sum of the random values can be efficiently computed. The combination of this and a threshold decryption scheme leads to the desired functionality.

4.10 Mix-Nets
Mix-nets are protocols that provide anonymity for senders by collecting encrypted messages from several users. For instance, one can consider mix-nets that collect ciphertexts and output the corresponding plaintexts in a randomly permuted order. In such a scenario, privacy is achieved by requiring that the permutation that matches inputs to outputs is kept secret to anyone except the mix-net. In particular, determining a correct input/output pair, i.e., a ciphertext with corresponding plaintext, should not be more effective then guessing one at random. A desirable property to build such mix-nets is reencryption which is achieved by using homomorphic encryption.

4.11 Data Aggregation In Wireless Sensor Networks:
In-network data aggregation in WSNs is a technique that combines partial results at the intermediate nodes route to the base station, thereby reducing the communication overhead and optimizing the bandwidth utilization in the wireless links. In applications such as healthcare and military surveillance where the sensitivity of private data of the sensor is very high, the aggregation has to be carried out in a privacy-preserving way, so that the sensitive data are not revealed to the aggregator. Homomorphic encryption schemes can be applied to protect privacy of input data while computing an arbitrary aggregation function in a wireless sensor network.

5. Comparative Analysis
Comparative analysis of all the different cryptographic algorithms reveals many facts about cryptographic algorithms execution [14].

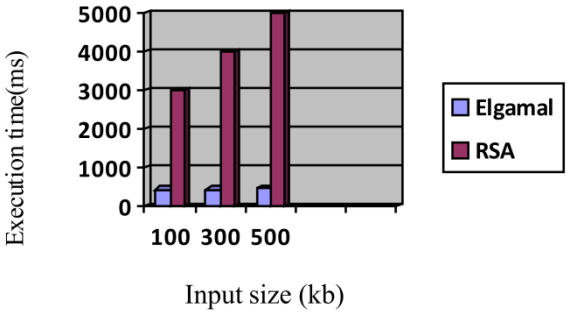

Fig.1. Input Size Vs Execution Time

RSA needs extra processing power than Elgamal for cryptographic algorithms. Paillier is faster than Benaloh. Key size of 4096 is used in RSA and Elgamal. Both algorithms are found very slow on cloud network as compared to local system. Both algorithm needs an extra processing power for its fast operation from cloud network, it needs better configuration machines (more number of processors, fast processors, more RAM and cache memory) to operate than other cryptographic algorithms. Homomorphic cryptosystems have also added new challenges towards the secure and fast execution of programs on cloud. homomorphic algorithms have to pass a long way before their actual implementation on cloud.

6. Conclusion
A homomorphic cryptosystem is a cryptosystem with the additional property that there exists an efficient algorithm to

compute an encryption of the sum or the product, of two messages given the public key and the encryptions of the messages but not the messages themselves. It can be used whenever the need of doing computations on pieces of un-owned information appears.

Instead of encrypting each plaintext bit separately, multiple bits can be packed together, thereby amortizing the encryption effort and reducing overhead. Homomorphic addition takes milliseconds, multiplication generally less than a second. These timings are a vast improvement over earlier efforts. Common cryptography can't directly calculate on encrypted data, but homomorphic encryption can, meanwhile, the operation results of homomorphic encryption will be automatically encrypted.

References

[1] Kristin Lauter, Michael Naehrig and Vinod Vaikuntanathan "Can Homomorphic Encryption be Practical?" Proceedings of the ACM Cloud Computing Security Workshop (CCSW 2011, Chicago IL, USA, 2011, (pp. 113-124).

[2] Craig Gentry, Amit Sahai, Brent Waters, Homomorphic Encryption from Learning with Errors: Conceptually-Simpler, Asymptotically-Faster, Attribute-Based. IACR Cryptology ePrint Archive, 2013

[3] S. Sobitha Ahila etal State Of Art in Homomorphic Encryption Schemes. Int.Journal of Engineering Research and Applications Vol. 4, Issue 2 (Version 6), February 2014, pp.37-43

[4] Galbraith, S. D. (2002). Elliptic Curve Paillier Schemes. Journal of Cryptology, Vol 15, No 2, pp.129-138, August 2002.

[5] Castagnos, G. (2007). An Efficient Probabilistic Public-Key Cryptosystem over Quadratic Fields Quotients. Finite Fields and Their Applications, Vol 13, No 3, pp. 563-576, July 2007.

[6] Z. Erkin, M. Beye1, T. Veugen, and R. L. Lagendijk, " Privacy-Preserving Content-Based Recommendations through Homomorphic Encryption" Proceedings of the 33rd WIC Symposium on Information Theory in the Benelux and The 2rd Joint WIC/IEEE Symposium on Information Theory and Signal Processing in the Benelux, *PP.71-77*,2012

[7] Dan Boneh, Craig Gentry, Shai Halevi, Frank Wang, David J. Wu: Private Database Queries Using Somewhat Homomorphic Encryption. ACNS 2013: 102-118

[8] Y.Bhargav *et al*, Homomorphic Recommendations for Data Packing- A SurveyInternational Journal of Computer Science and Mobile Applications, Vol.1 Issue. 5, November- 2013, pg. 18-37

[9] Niranjan Kumar Nakkala1, Ch.Ram Mohan2, Dr.N.V.Rao, Generating Private Recommendations Efficiently Using GAE Datastore and Data Packing, *International Journal of Advanced Research in Computer and Communication Engineering Vol. 3, Issue 2, February 2014*

[10] Jean-Sébastien Coron, Tancrède Lepoint, Mehdi Tibouchi: Scale-Invariant Fully Homomorphic Encryption over the Integers. IACR Cryptology ePrint Archive 2014pg. 311-328

[11] Reginald L. Lagendijk, Zekeriya Erkin, Mauro Barni: Encrypted Signal Processing for Privacy Protection: Conveying the Utility of Homomorphic Encryption and Multiparty Computation. IEEE Signal Process. Mag. 30(1): 82-105 (2013)

[12] C. Fontaine and F. Galand, "A survey of homomorphic encryption for nonspecialists," *Eurasip J. Inform. Security*, vol. 10.1155/2007/13801, 2007

[13] Naoki Ogura, Go Yamamoto, Tetsutaro Kobayashi, Shigenori Uchiyama: An Improvement of Key Generation Algorithm for Gentry's Homomorphic Encryption Scheme. IWSEC 2010:70-80

[14] Vineet kumar singh and Maitreyee Dutta, "Analyzing Cryptographic algorithms for secure cloud network", International Journal of studies in Computer Science and Engineering, Vol. 3, No.6, 2014

[15] Naehrig, Michael, Kristin Lauter, and Vinod Vaikuntanathan. "Can homomorphic encryption be practical?" Proceedings of the 3rd ACM workshop on Cloud computing security workshop, ACM, 2011

[16] Dimitrios Zissis and Dimitrios Lekkas "Addressing cloud computing security issues", Future Generation Computer Systems, Elsevier 2010, 28 (2012) 583–592.