HOMOMORPHIC ENCRYPTION: A SURVEY

Daniel Okunbor and Chekad Sarami
Department of Mathematics and Computer Science
Fayetteville State University
Fayetteville, NC 28301
{diokunbor, csarami}@uncfsu.edu)

Abstract:

Homomorphic encryption is an encryption algorithm that allows for computations directly on the encrypted data. We perform important mathematical operations and analyses while the data remain encrypted. In this manner, we eliminate the associated costs for decrypting data before performing these operations and re-encrypting afterwards. It is argued in recent literature that this new and innovative encryption approach may ensure information security, particularly, for encrypted data stored on clouds. With cloud computing, it is no longer necessary to retrieve encrypted data, decrypt, perform operations and encrypt. The cloud-computing platform will have the necessary computing infrastructure and power for performing homomorphic encryption. Homomorphic encryption eliminates man-in-the middle attacks since it would not be necessary to move data between the cloud and the enterprise system unless during the initial uploads and complete deletion of data from the cloud. In this paper, we would give the historical perspective and discuss various applications of homomorphic encryption. We would present different homomorphic encryption models as proposed in the research community. The application of homomorphic encryption in cloud computing has been the focus and therefore, a major part of the paper will be devoted to discussion of security of cloud computing and the attempts made to utilize homomorphic encryption as a model for cloud computing security.

I. Introduction

The word homomorphic was first recorded in the 1800s and was coined from Greek words homos to mean "same" and morphe to mean "form" or "shape." It is used in mathematics to describe two related sets, that is, a mapping from one set to the other that preserves operations on the set elements. The result obtained from applying operations on elements of the first set can be mapped onto the result obtained from applying same operations on elements of the second set. A homomorphism is therefore a map that inherently preserves properties between two algebraic structures having same type. It is in this that context that linear maps of vector spaces are homomorphic.

More formally, given an operation, φ, a map $g: X \to Y$ that preserves the operation φ of elements in both X and Y is homomorphic if

$$g\big(\varphi_X(x_1, x_2, \dots, x_N)\big) = \varphi_Y\big(g(x_1), g(x_2), \dots, g(x_N)\big), for\ x_k \in X, g(x_k) \in Y$$

In algebra, groups (a set consisting of identity and one operation and its inverse) and rings (a set consisting of identity with binary operations and inverse operations) are known to possess homomorphic property. Maps from one group (or ring) to another group (or ring) preserve the operations. This is probably one for the motivations for the adoption of homomorphism in encryption. Most cryptographic algorithms are based on the underlying algebraic concepts of groups and rings.

The idea of homomorphic for use in encryption was first introduced in 1978 in a seminal paper by Ronald Rivest, Leonald Adleman and Michael Dertouzos entitled "On Data Banks and Privacy Homomorphisms" (Rivest, Adleman and Dertouzos, 1978). Focusing on privacy of sensitive data, particularly customer information stored by a loan company that provides a time-shared service, they explained that traditionally, if a customer data is stored in an encrypted form, to perform operations on them, they must be decrypted. This, they claimed is a limitation on the encryption system. They proposed the use of a special privacy homomorphism to encrypt its data so that the time-shared service system can operate on the data without the necessity of decrypting it first. This is what gave birth to homomorphic encryption. Homomorphic encryption is one in which given the encryption algorithm ϕ and its inverse (decryption algorithm) ϕ^{-1} , computations based on operation φ are done on $Y = \phi(X)$ so that $\phi^{-1}(\varphi(Y)) = \varphi(X)$. The requirements are that ϕ and ϕ^{-1} are easily computable; knowledge of ϕ and/or some $x \epsilon X$ should not reveal ϕ^{-1} (chosen plaintext or ciphertext only attacks); and the operation φ is computationally efficient. While Rivest, Adleman and Dertouzos (1978) proposed a few homomorphic encryptions including the RSA, they cautioned that their proposed approaches are susceptible to a variety of attacks and therefore not suitable for privacy homomorphism. Added to the concerns for the proposed system was their practicability.

This early lack of euphemism for homomorphic encryption might have contributed to the limited research on the topic in the 80s and 90s. Although, there were anecdote of research in homomorphic encryption in the 80s and 90s as will be described in subsequent sections, rapid research interest was not until 2009 when Gentry in his doctoral thesis presented the practicality of homomorphic encryption or what is termed fully homomorphic encryption (Gentry, 2009). The growing concerns for security in cloud computing also provide a strong precipitate for homomorphic encryption. A lot of what is available on homomorphic encryption in the literature is still theoretical, although, there are computing libraries that provide support for homomorphic encryption. There is also a growing research in quantum homomorphic encryption for enabling quantum computation on encrypted quantum data.

This paper will present the historical perspective and various homomorphic encryption schemes in Section II. In Section III, we would describe existing techniques, including partially, "somewhat" and fully homomorphic encryptions and their implementations. As indicated earlier, cloud computing is clearly a good candidate for homomorphic encryption. Therefore, Section IV will focus on existing implementations of homomorphic encryption in cloud computing. Section V will summarize and provide concluding remarks.

II. Historical Perspective

As discussed in Section II, the concept of homomorphism dates to the 1800s and was used to describe a variety of algebraic constructs, such as groups, rings, and vector spaces. Its use in cryptography became visible in the paper written by Rivest, Adleman and Dertouzos (1978), in which they proposed few homomorphic cipher systems. A homomorphic encryption is defined as one that permits computations on ciphertexts, producing results which when encrypted correspond to computations performed on plaintexts, see the Figure 1 below for a schematic representation (Crane, 2017).

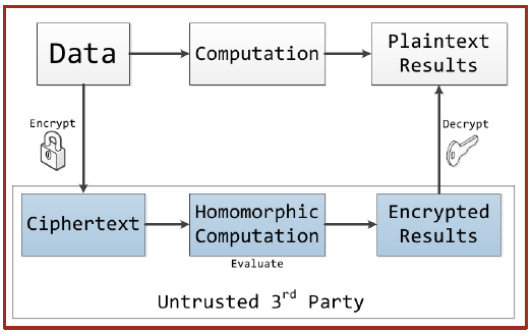

Figure 1: Schematic Representation of Homomorphic Encryption

Mathematically, we have that, given an encryption algorithm scheme consisting of (P, C, K, E, D), where P, C are plaintext and ciphertext spaces that form groups (P, o) and $(C, \dot{o})$, respectively, where o and $\dot{o}$ are operations, K is the key space and E, D are encryption and decryption algorithms respectively. For all $p_1, p_2 \epsilon P \; and \; k \epsilon K$, if

$$E_k(p_1) \dot{o} E_k(p_2) = E_k(p_1 o p_2)$$

holds, the encryption scheme is homomorphic (Yi, Paulet and Bertino, 2014). In addition to the popular Rivest, Shamir and Adelman (RSA) encryption, Rivest, Adleman and Dertouzos described four other homomorphic encryptions. Two of these encryptions are partially homomorphic (**only for addition or multiplication**) and three are fully homomorphic (**that is, a homomorphic encryption that is valid for addition and multiplication**). The discovery of Rivest, Adleman and Dertouzos (1978) is the foundation of subsequent homomorphic encryption schemes. The encryption scheme (GM encryption scheme) developed by Shafi Godlwasser and Silvio Micali in 1982 and based on factorization of large numbersnis partially homomorphic with $o \; and \; \dot{o}$ being addition and multiplication, respectively. The Elgamal encryption scheme, invented by Taher Elgamal in 1985 in partially homomorphic for multiplication only. Other fully homomorphic encryptions, include those developed by Boneh, Goh, and Nissim in 1998 called BGN encryption scheme and by Pascal Paillier in 1999 called Paillier encryption scheme (Yi, Paulet and Bertino, 2014).

The revolutionary work of Gentry in his doctoral thesis spurred the surge for interests in homomorphic encryption. He used the lattice-based cryptographic approach to model fully homomorphic encryption (Gentry, 2009). His encryption scheme is considered as the first implementable fully homomorphic encryption and are

categorized as "somewhat" or bootstrappable encryption. The designation "somewhat" or bootstrappable is due to the minimal "noise" from these homomorphic encryption schemes. Several variants of original Gentry's encryption scheme are now available (Gentry and Halevi, 2010; Smart and Vercauteren, 2010; van Dijk, Gentry, Halevi and Vaikunanatha, 2010). Other fully homomorphic encryptions, include those developed by Brakerski, Gentry, and Vaikuntanathan (2012), Brakerski and Vaikuntanathan (2011), and Brakerski(2012). These encryption schemes have better efficiency and their security is based on the concept of learning with errors (LWE) problem prevalent in machine learning.

III. Existing Homomorphic Encryptions

As described in Section II, a homomorphic encryption is based on key generation, encryption, homomorphic computations and decryption. Encryption, as we know it is categorized into symmetric (secret key encryption) such as Data Encryption Standard (DES) and Advanced Encryption Standard (AES) and asymmetric (public key encryption) such as RSA, Elgamal and Elliptic Curves. It is a well-known fact that symmetric encryption is computationally more efficient than asymmetric, however, they suffer the key exchange problem. The public key encryption while relatively less computationally efficient solves the key exchange problem as espoused by Diffie and Hellman. and therefore, in practice the symmetric encryption is used in combination with the public key encryption. In this respect, the public key encryption is used for exchanging the symmetric encryption secret through the unsecure communication channel. However, the homomorphic encryption described here is not based on this principle. A homomorphic encryption must be such that computations are easily performed in the encrypted mode. Although, most of the homomorphic encryption algorithms are based on public key encryption, they are some proposals for the homomorphic encryption for AES at the circuitry level (Gentry, Halevi and Smart, 2012). This paper focuses mostly on public key encryption. In what follows, we would present some of the existing examples of homomorphic encryption.

a. RSA Encryption: The public key $k = (n, e)$ and for $p_1, p_2 \epsilon P$

$$E_k(p_1).E_k(p_2) = p_1^e.p_2^e(mod\ n) = (p_1.p_2)^e(mod\ n) = E_k(p_1.p_2)$$

Therefore, RSA has multiplicative homomorphic property only and it is not fully homomorphic. This is an unpadded RSA that is known to be insecure.

b. Paillier Encryption: The public key $k = (n, g)$ and for $p_1, p_2 \epsilon P$

$$E_k(p_1).E_k(p_2) = g^{p_1}.g^{p_2}(mod\ n) = g^{p_1+p_2}(mod\ n) = E_k(p_1 + p_2)$$

Therefore, Paillier encryption has additive homomorphic property. Although, it is claimed that it has multiplicative property if an encrypted plaintext is raised to the power another plaintext $mod(n^2)$.

c. Goldwasser-Micali (GM) Encryption: In the GM encryption, the message is encoded into bit string, $p = (b_1 b_2 b_3 \cdots b_l)$ and the bits are encrypted one at a time to obtain the ciphertext: $c = (c_1 c_2 c_3 \cdots c_l)$. The public key: $k = (a, N)$, where a is a function of two large $N = qr$ and m_l are random numbers such that $\gcd(N, m_l) = 1$ and

$$\left(a_q^{\frac{q-1}{2}}, a_r^{\frac{r-1}{2}}\right) = -1(mod(q), mod(r))$$

$$E_k(c_i) \cdot E_k(c_j) = (m_i^2 \cdot a^{b_i}) \cdot (m_j^2 \cdot a^{b_j})\ (mod\ N) = (m_i \cdot m_j)^2 \cdot a^{(b_i+b_j)}\ \ (mod\ N) = \ E_k(b_i + b_j)$$

Therefore, the GM has additive homomorphic property.

d. Boneh-Goh-Nissim Encryption (BGN). The public key $k = \{N, G, G_1, e, g, h\}$, where $N = q_1 q_2$ is the order of the cyclic group G and the map $e: G \times G \to G_1$ and $g\ and\ u$ are random generators and $set\ h = u^{q_2}$. The plaintext must be encoded so in the set $\{0,1,2, \dots, T\}, T < q_2$. To encrypt a message p, pick r and $c = g^p h^r$. Therefore

$$E_k(p_1) \cdot E_k(p_2) = g^{p_1} h^{r_1} \cdot g^{p_2} h^{r_2} = g^{p_1+p_2} h^{r_1+r_2} = E_k(p_1 + p_2)$$

Therefore, BGN has additive homomorphic property.

e. Dijk, Gentry, Halevi and Vaikuntanathan (DGHV) Encryption: This scheme is based on Example 5 of the homomorphic encryptions in Rivest, Adelman and Dertouzos (1978). Assume that the message $m \in \{0,1\}$ and let p be the secret-key and $q\ and\ r$ are random numbers, then the encryption and decryption are

$$c = q \cdot p + 2r + m, \qquad m = (c\ mod\ p)\ mod\ 2.$$

This is fully homomorphic, since, we can show that,

$$c_1 + c_2 = q' \cdot p + 2r' + m_1 + m_2 \text{ and } c_1 \cdot c_2 = q'' \cdot p + 2r'' + m_1 \cdot m_2$$

To generalize, compute a set of τencryptions of 0's and define the public key using

$$x_i = q_i \cdot p + 2r_i$$

and the encryption is

$$c = m + 2r + \sum_{1}^{\tau} \varepsilon_i \cdot x_i,\ \varepsilon_i \in \{0,1\}$$

The computational "noise" that is associated with DGHV scheme and other homomorphic encryption schemes resulted in different implementations and variants. The "somewhat" homomorphic encryption applies smaller degree polynomial homomorphically to the ciphertexts. The "squashed" homomorphic encryption applies smaller degree polynomials for decryption.

IV. Cloud Computing and Homomorphic Encryption

There are several applications of homomorphic encryption, such as e-voting and computing on the cloud. In the case of e-voting, voters encrypt their votes and aggregate results are computed from the encrypted votes using any of the homomorphic encryption schemes as appropriate without knowing how the voter voted or knowing the voter's information.

In cloud computing, the cloud receives data in encrypted form and does not know the plaintext data (m). The cloud can perform computations $f(m)$ on the ciphertexts without knowing m. Typically, these computations are written in Boolean circuits using XOR and AND operations, since the operations XOR and AND provide homomorphic maps, the ciphertext of the computed operation is easily decrypted by the user to obtain the corresponding value in plaintext. Clearly, the user has the secret key and only the public key is made available to the

cloud provider. The computations referred here may include computing aggregates, searching and editing data on the cloud without the cloud providers learning anything about the data.

V. Conclusion

In this paper, we present the history of homomorphic encryption and describe some of the existing homomorphic encryption schemes, including those that are partially and fully homomorphic in nature. A fully homomorphic encryption has additive and multiplicative properties. The DGHV scheme has received an extensive coverage due to its practicability. The DGHV scheme has several variants and implementations and there are available as software libraries such as HELib that implements DGHV schemes. The Boolean circuitry nature of DGHV scheme makes it very attractive for cloud computing.

References

Brakerski, Z., Gentry, C., and Vaikuntanathan, V. 2012. (Leveled) fully homomorphic encryption without bootstrapping, Procs. of the 3rd Innovations in Theoretical Computer Science Conference, ACM, pp. 309–325.

Crane, S. 2017. Implementing a Simple Homomorphic Encryption Scheme Stephen Crane - Cal Poly Pomona http://www.cpp.edu/~honorscollege/documents/convocation/SCI/CS_Crane.pdf (Presentation at the Kellog Honors College Convocation, California Polytechnic).

Brakerski, Z., and Vaikuntanathan, V. 2011. Efficient fully homomorphic encryption from (standard) LWE, 2011 IEEE 52nd Annual Symposium on Foundations of Computer Sci., IEEE Computer Soc., Los Alamitos. pp. 97–106.

van Dijk, M., Gentry, C., Halevi, S., and Vaikuntanathan, V. 2010. Fully homomorphic encryption over the integers, Advances in cryptology.Lecture Notes in Comput. Sci., vol. 6110, Springer, Berlin, pp. 24–43.

Smart, N.P.,and Vercauteren, F. 2010. Fully homomorphic encryption with relatively small key and ciphertext sizes, Public key cryptography—PKC 2010, Lecture Notes in Comput. Sci., vol. 6056, Springer, Berlin, pp. 420–443.

Rivest, R., Adleman, L., and Dertouzos, M. 1978. On data banks and privacy homomorphisms. Foundations of secure computation 4, no. 11: 169-180.

Coron, J., Mandal, A., Naccache, D., and Tibouchi, M. 2011. Fully homomorphic encryption over the integers with shorter public keys, Advances in cryptology. Lecture Notes in Comput. Sci., vol. 6841, Springer, pp. 487–504.

Gentry, C. 2009. Fully Homomorphic Encryption Using Ideal Lattices, in *Procs. 41st Annual ACM Symposium on Theory of Computing*. See, A fully homomorphic encryption scheme, Ph.D. thesis, Stanford University, 2009.

Yi, X., Paulet, P. and Bertino, E. 2014. Homomorphic Encryption and Applications, Springer Briefs in Comp. Sci.

Gentry, C., and Halevi, S. 2011. Implementing Gentry's fully-homomorphic encryption scheme, Advances in cryptology—EUROCRYPT 2011, Lecture Notes in Comput. Sci., vol. 6632, Springer, Heidelberg, pp. 129–148.

Gentry, C., Halevi,S., and Smart N. 2012. Fully homomorphic encryption with polylog overhead, Advances in cryptology—EUROCRYPT 2012, Lecture Notes in Comput. Sci., vol. 7237, Springer, Heidelberg, pp. 465–482.