Theory and Application of Homomorphic Encryption

Cody Tinker
Department of Computer Engineering
Rochester Institute of Technology
cwt9976@rit.edu

***Abstract*—Homomorphic Encryption is an encryption scheme that describes a method of performing operations on encrypted data without the need to first decrypt the data. The features of a fully homomorphic encryption scheme are very promising, however, there are many obstacles to still overcome before a fully homomorphic scheme becomes practical to use in practice. This paper explores the theory behind homomorphic encryption, primarily fully homomorphic encryption, and some of the problems that hinder its practicality. In addition, the results of a homomorphically AES circuit is explored to evaluate the proof-of-concept performance of a homomorphic encryption application.**

***Index Terms*—Homomorphic Encryption, FHE, LWE, R-LWE**

I. Introduction

A. Overview

Homomorphic Encryption (HE) is a type of encryption that allows a party to perform computations on encrypted data without the need to decrypt the underlying data. These computations result in ciphertext that matches a ciphertext as if the operation result of the original plaintexts was encrypted. The concept of a homomorphic encryption scheme is not new and has been cited for the first time by Rivest, Adleman, and Dertouzous in 1978 []. Although there has been a lot of difficulty regarding the discovery of an unrestricted HE scheme, the implications of such a scheme that is able to perform arbitrary operations on arbitrary data is huge.

B. Motivation

The purpose of cryptography is to protect data in the presence of an adversary with concerns regarding the following: authentication, confidentiality, integrity, and non-repudiation. A growing aspect in industry is to offload large data storage and expensive computation to third party services. Although traditional cryptographic systems are good at keeping data confidential and intact during transfer, the security then falls onto the third party service provider once the data is decrypted for operational use. As soon as the data is decrypted, the confidentiality aspect of the data is lost as the data is now available in plain sight to the servicer. HE schemes pose a solution to this problem as the data that is sent to the servicer would never have to be decrypted and instead a user can simply instruct the servicer how to operate on the data. In addition, these schemes could also provide means for achieving other specialized forms of cryptographic systems.

II. Homomorphic Encryption

A. Definition

Homomorphic Encryption as type of encryption that allows a party to compute operations on encrypted data without the need to know the underlying plaintext data. Homomorphism is an algebraic term that comes from the greek words *homos*, meaning "same", and *morphe*, meaning "form" []. Algebraically speaking, it is a term that is used to define a map between two algebraic structures that preserves the operations of the structures. In other words, given two algebraic structures with their own respective operations, both denoted by (A,$*$) and (B,$\circ$), and a map f, denoted $f : A \mapsto B$, the map f is said to be homomorphic if the following holds:

$$f(x * y) = f(x) \circ f(y); x, y \in A$$

It follows to say that an encryption scheme that exhibits homomorphism is one whose mapping function, namely the encryption function, preserves the operations between the plaintext space and ciphertext space.

III. Types of Homomorphic Encryption

Homomorphic encryption refers to the type of encryption that allows a user to operate on encrypted data, however, there actually different HE schemes that fall within the definition of three different types: Partially Homomorphic, Somewhat Homomorphic, and Fully Homomorphic. The main distinction lies with what operations, primarily among addition and multiplication, that a user can perform on the data how the operation is restricted by the scheme.

A. Partially Homomorphic Encryption

A Partially Homomorphic Encryption (PHE) scheme is one that allows an unbounded operation on encrypted data but the types of operations allowed is limited to that single operation. Although PHE schemes can be useful, the applications of which they are applicable is limited due to their limited operation set. There exists a couple of applications where such schemes are used such as e-voting and private information retrieval (PIR) [2].

B. Somewhat Homomorphic Encryption

A Somewhat Homomorphic Encryption (SWHE) scheme is one that allows a set of operations, normally both addition and multiplication, but with a bound on either one or both

operations. Normally, these schemes allow one operation, usually addition, to be performed an unlimited amount of times while the other operation, usually multiplication, to be performed a number of times within some bound. SWHE schemes are applicable to applications where the complexity and depth of the functions involved are known beforehand but falls short for applications that desire arbitrary computation on any piece of data. In addition, data can only be operated on to a certain extent before the decrypt function will stop evaluating the ciphertext correctly so this limits what kind of applications can be implemented using such a scheme.

C. Fully Homomorphic Encryption

A Fully Homomorphic Encryption (FHE) scheme is one that allows an arbitrary set of operations to be evaluated on encrypted data an unlimited number of times. Despite the usefulness of the prior two types of schemes, this type of scheme is the most promising as it offers nearly no limitations on the type of program one can execute with encrypted data. However, despite much research effort, it has proved difficult to construct a scheme that achieves full and unbounded homomorphism. No such schemes were even devised until Craig Gentry's breakthrough in 2009. Since then, several FHE schemes have been published and much improvement has been seen in the development of these schemes, but there are still several issues that need to be improved upon before these schemes will realistically see use in industry.

IV. A Basic Scheme

A basic scheme is presented to demonstrate the homomorphism properties as it applies to an encryption scheme. The scheme is borrows heavily from Van Dijk's FHE scheme over integers [5]. The encryption methods demonstrate how a message is embedded in the ciphertext and the decryption method demonstrates how the message can be retrieved by stripping away encryption elements. The addition and multiplication operations demonstrate the homomorphism of the scheme.

A. Parameters

The scheme only requires a prime value p. This value is the secret key. The scheme has a message space $m \in \{0, 1\}$.

B. Encryption

Generate two values, e and q, randomly from some distribution. The message m is encrypted to obtain ciphertext c by:

$$c = E(m) = m + 2e + pq \tag{1}$$

Many SWHE and FHE schemes follow a similar format for encryption. The idea is that a message can be encrypted by summing it with a product of two element plus a small "noise" term.

C. Decryption

The ciphertext c is decrypted to obtain the message m by:

$$m = D(m) = (c \bmod p) \bmod 2 \tag{2}$$

The proof of correctness is as follows:

$$\begin{gathered} m = (c \bmod p) \bmod 2 \\ m = (m + 2e + pq \bmod p) \bmod 2 \\ m = (m + 2e) \bmod 2 \\ m = m \ \square \end{gathered}$$

It should be noted that the ciphertext only decrypts correctly if $(m + 2e) < p/2$.

D. Addition

Two ciphertexts, c_1 and c_2, that encrypt two messages, $m_1 and m_2$, can be added to obtain a new ciphertext, c_3. Decrypting c_3 results in a plaintext value whose value equals the evaluation of m_1 + m_2. The proof of homomorphism of the scheme over addition is as follows:

$$\begin{gathered} c_3 = c_1 + c_2 \\ c_3 = m_1 + 2e_1 + pq_1 + m_2 + 2e_2 + pq_2 \\ c_3 = (m_1 + m_2) + 2(e_1 + e_2) + p(q_1 + q_2) \\ c_3 = (m_1 + m_2) + 2e_3 + pq_3 \end{gathered}$$

It can be seen that the resulting ciphertext retains an encryption structure of the sum of messages m_1 and m_2.

E. Multiplication

Two ciphertexts, c_1 and c_2, that encrypt two messages, $m_1 and m_2$, can be multiplied to obtain a new ciphertext, c_3. Decrypting c_3 results in a plaintext value whose value equals the evaluation of $m_1 m_2$. The proof of homomorphism of the scheme over multiplication is as follows:

$$\begin{gathered} c_3 = c_1 c_2 \\ c_3 = (m_1 + 2e_1 + pq_1)(m_2 + 2e_2 + pq_2) \\ c_3 = (m_1 m_2) + 2(m_1 r_2 + m_2 r_1 + 2e_1 e_2) + kp \end{gathered}$$

Note that the value k is simply the sum of product terms with the value p from the multiplication of the two ciphertexts. It can be seen that the resulting ciphertext retains an encryption structure of the product of messages m_1 and m_2.

F. Summary

It can be seen that the scheme presented is indeed homomorphic. It should be noted that the scheme is only able to operate on binary bit data. The reasoning is this allows a user to construct a binary circuit using the homomorphic operations. A binary addition operation will result in an operation that matches a binary XOR-gate and a binary multiplication operation will result in an operation that matches a binary AND-gate. A construction using these gates can perform any operation as these two binary operations form a functionally complete set. Other FHE schemes follow similar constructions.

V. Gentry's Breakthrough

In 2009, Craig Gentry revealed in his PhD dissertation [3] the first plausible and achievable FHE scheme. His discovery is huge milestone for the long term problem of finding a FHE scheme, but the arguably more valuable information that came from his dissertation was the general blueprint that he outlines for achieving FHE schemes that has paved the way for researchers. Gentry shows that one can obtain a FHE scheme using a SWHE scheme as a basis using two techniques he dubs "squashing" and "bootstrapping". A SWHE scheme is said to be *bootstrappable* if it is able to evaluate its own decryption circuit. The idea is if a SWHE scheme is bootstrappable, a ciphertext can be "refreshed" or "re-encrypted" with an elaborate scheme of encrypting and decrypting a ciphertext homomorphically to produce a clean ciphertext.

A. Squashing

In order for a SWHE scheme boostrappable, it must be able to evaluate its own decryption circuit. However, as it turns out, a decryption circuit usually involves a circuit of large depth. The squashing technique is a method that reduces the decryption circuit depth. The squashing technique involves finding a set whose sum of elements equals the inverse of the secret key and then multiplying a ciphertext by each of the set elements. This reduces the decryption circuit to one whose depth is small enough that the SWHE scheme can evaluate.

B. Bootstrapping

The bootstrapping technique is a method to create a "fresh" ciphertext from a noisy ciphertext that will decrypt to the same plaintext. The bootstrapping method assumes there are two sets of public and secret key pairs and a ciphertext that is encrypted with the first key pair. First, squashing is applied to a ciphertext to make it bootstrappable. The technique basically involves decrypting the ciphertext homomorphically using an encryption of the first secret key and then applying encryption using the second public key. The resulting ciphertext is a ciphertext with "fresh" noise but encrypted under a new key pair. This technique allows a SWHE scheme to act as a FHE scheme by refreshing a ciphertext after any operations have been evaluated. However, it should be noted that this particular method is computationally expensive which is a major drawback to the implementation of a FHE scheme.

VI. Security and Parameter Sizes

Security is still a somewhat open question as it relates to HE schemes since the hardness assumptions and parameters change from scheme to scheme. There has been a recent effort for standardization of HE [1]. The standard uses the Learning With Errors (LWE) and Ring-Learning With Errors (RLWE) hardness problems for the basis of its security analysis as these hardness problems form the security basis for the most practical and promising FHE schemes.

A. LWE

The LWE problem is defined as follows [6]:

> "Fix a size parameter $n \geq 1$, a modulus $q \geq 2$, and a distribution χ on $\mathbb{Z}_q$. Let $\mathbb{A}_{s,\chi}$ on $\mathbb{Z}_q^n \times \mathbb{Z}_q$ be the probability distribution obtained by choosing a vector $\mathbf{a} \in \mathbb{Z}_q^n$ uniformly at random, choosing $e \in \mathbb{Z}_q$ according to χ, and outputting $(\mathbf{a}, \langle \mathbf{a}, \mathbf{s} + e \rangle)$, where additions are performed in $\mathbb{Z}_q$. We say that an algorithm solves LWE with modulus q and error distribution χ if, for any $\mathbf{s} \in \mathbb{Z}_q^n$, given an arbitrary number of independent samples from $\mathbb{A}_{s,\chi}$ it outputs $\mathbf{s}$ with high probability."

The LWE problem is known to be reduced to the hard lattice problems.

B. RLWE

The RWLE problem is a special case of the LWE problem where the oracle $\mathbb{A}_{s,\chi}$ is chosen to have a special algebraic structure. The RLWE problem is defined as follows [7]:

> "Fix a dimension parameter $n \geq 1$ and n is a power of 2, a modulus $q \geq 2$, and a distribution χ on R, where R is a ring defined by $\mathbb{Z}_q[x]/\langle x^n + 1 \rangle$. Let $\mathbb{A}_{s,\chi}$ on R be the probability distribution obtained by choosing an element $a \in R$ uniformly at random, choosing $e \in R$ according to χ, and outputting $(a, a \cdot s + e)$, where multiplications and additions are performed in R. We say that an algorithm solves RLWE with modulus q and error distribution χ if, for any $s \in R$, given an arbitrary number of independent samples from $\mathbb{A}_{s,\chi}$ it outputs s with high probability."

The RLWE scheme offers increases in performance and a reduction in key sizes for applicable FHE schemes and therefore is preferred over LWE as a security basis. The RLWE problem is speculated to be reduced to hard lattice problems.

VII. Security Parameters

The standard defines several tables of security parameters based on the RLWE hardness problem and known attacks. The tables provide pairs of (n,q), where n is the dimension of the ring elements and q is the coefficient modulus of the ciphertext, for varying levels of security. One collection of parameter results from [1] is represented in Table I.

It should be noted that these parameters will directly affect the amount of space the ciphertexts and keys require. In fact the ciphertext expansion for a bit of data is expected be equal to some multiple of the product of n and log$q$. Therefore, it is easy to see that with large parameters the ciphertext expansion will grow very large.

VIII. Performance

One of the greatest hindrances to a practical FHE scheme is its performance, that is, how fast a scheme will operate and how much memory it requires to operate. An AES-128 circuit was homomorphically evaluated with an implementation the

TABLE I
FHE PARAMETERS FOR VARYING SECURITY LEVELS

n	logq	security
1024	29	128
	21	192
	16	256
2048	56	128
	39	192
	31	256
4096	111	128
	77	192
	60	256
8192	220	128
	154	192
	120	256
16384	440	128
	307	192
	239	256
32768	880	128
	612	192
	478	256

BGV FHE scheme [] in order to assess these qualities. The AES circuit was chosen because of its natural benchmark features: AES is widely used, AES is nontrivial, and AES has a regular structure. Both a non-bootstrapped variant and a bootstrap variant were implemented and compared for purposes of evaluating the affect that bootstrapping has on the performance. For the non-bootstrapper variant, 120 AES blocks were packed into a single ciphertext and were encrypted and decrypted homomorphically. For the bootstrapping variant, 180 AES were packed into a single ciphertext and were encrypted and decrypted homomorphically. The program was executed on an Intel Core i5-3320M running at 2.6GHz with 4GB of RAM. The execution results of the program is shown in Table II [4].

TABLE II
HOMOMORPHIC AES PERFORMANCE

Test	m	$\phi(m)$	lvls	$\|Q\|$	security	params/key-gen	Encrypt	Decrypt	memory
no bootstrap	53261	46080	40	886	150-bit	26.45 / 73.03	245.1	394.3	3GB
bootstrap	28679	23040	23	493	123-bit	148.2 / 37.2	1049.9	1630.5	3.7GB

As the results suggest, it is somewhat expensive to compute even on small amounts of data with a FHE scheme. However, at the same time, it does suggest practicality is within grasp. One note by the authors is that no parallelism is utilized in their implementation despite how parallelizable such a program is []. In addition, there may be significant speed ups achievable through hardware solutions.

IX. CONCLUSION

Homomorphic encryption is an exciting topic for cryptographers because of the implications that a FHE scheme could offer. With FHE, cloud based computations can be completely encrypted and thus provide full confidentiality to its users. This would prove beneficial to computation heavy applications that handle sensitive data such as machine learning algorithms or handling of medical data. However, as research has shown, current FHE schemes come with serious drawbacks regarding its performance. Ciphertext expansion and memory usage is a serious issue and bitwise computations on encrypted data result in slow processing time. Although progress has been steady since Gentry's discovery in 2009, there is still work to be done.

REFERENCES

[1] Albrecht, M., Chase, M., Chen, H., Ding, J., Goldwasser, S., Gorbunov, S., Sahai, A. (2018). Homomorphic Encryption Standard.

[2] Acar, A., Aksu, H., Uluagac, A. S., & Conti, M. (2017). A Survey on Homomorphic Encryption Schemes: Theory and Implementation, 135. https://doi.org/10.1145/0000000.0000000

[3] Gentry, C. (2009). Fully homomorphic encryption using ideal lattices. Proceedings of the 41st Annual ACM Symposium on Symposium on Theory of Computing - STOC 09, 169. https://doi.org/10.1145/1536414.1536440

[4] Gentry, C., & Smart, N. P. (2015). Homomorphic Evaluation of the AES Circuit. Retrieved from http://eprint.iacr.org/2012/099.pdf

[5] Van Dijk, M., Gentry, C., Halevi, S., & Vaikuntanathan, V. (2010). Fully homomorphic encryption over the integers. Advances in Cryptology EUROCRYPT 10, 2443. https://doi.org/10.1007/978-3-642-38348-9_20

[6] Regev, O. (n.d.). The Learning with Errors Problem, 3(015848), 123.

[7] Lyubashevsky, V., Peikert, C., & Regev, O. (2010). On ideal lattices and learning with errors over rings. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 6110 LNCS(015848), 123. https://doi.org/10.1007/978-3-642-13190-5_1