Subject (Homomorphic Encryption)

2022-1st Semester

Undergraduate Research

Student ID.	20161190
Name	Minjae Lee
Department	Computer Science Engineering
Project Advisor	Miran kim

FIRST IN CHANGE

I. Introduction

1) Topic and Purpose of Research

In the past, encryption were simply used to protect data. But modern times, they have been developed to enable many meaningful applications. One of them is homomorphic encryption enables processing encrypted data without decrypting it. When homomorphic encryption first appeared, there were issues related to speed. But with better algorithms and optimizations, progress has been made enough to be applied to various technologies. Because of this feature, as other technologies develop, there are more areas where homomorphic encryption can be applied.

These days machine learning became popular, the importance of homomorphic encryption that can perform operations while protecting data necessary for learning has also increased. Homomorphic encryption may be applied to various fields such as medical fields where patient personal information is important, advertisements, and finance and so on. Also the prospect of outsourcing an increasing amount of data storage and management to cloud services raises many new privacy concerns for individuals and businesses. The privacy concerns can be satisfactorily addressed if users encrypt the data they send to the cloud. If the encryption scheme is homomorphic, the cloud can still perform meaningful computations on the data, even though it is encrypted.

In this paper, I will summary how the encryption system is formed and what techniques are applied based on the papers that I read.

II. Main Subject

1) Research Planning

- basic of HE

- Ring Learning With Error
- Basic operation

- Technique

- Relinearization
- Message Encoding
- Modulus switching and Bootstrapping
- Setting parameter

III. Conclusion and Discussion

1) Research Results

• Ring Learning With Error

RLWE is parametrized by parameters (m,q,χ) where m is the number of samples, as in the LWE problem above, q is a positive integer (the "modulus parameter") and χ is a probability distribution over the ring $R = Z[X]/f(X)$ (the "error distribution"). The RLWE assumption requires that the following two probability distributions are computationally indistinguishable:

> Distribution 1. Choose $m + 1$ uniformly random elements $s, a_1, \ldots, a_m$ from the ring R/qR, and m more elements $e_1, \ldots, e_m$ from the ring R chosen from the error distribution χ. Compute $b_i := sa_i + e_i$, all computations carried out over the ring R/qR. Output $\{(a_i,b_i) : i = 1, ...m\}$.
>
> Distribution 2. Choose $2m$ uniformly random elements $a_1, \ldots, a_m, b_1, \ldots, b_m$ from the ring R/qR. Output $\{(a_i,b_i) : i = 1, ...m\}$.

When making b, we don't just multiply the secret key, we add a little error, making it difficult to find the secret key. Using this rlwe system, homomorphic encryption has security.

• Basic function and operation

ParamGen(λ, *PT, K, B*) → Params The parameter generation algorithm is used to instantiate various parameters used in the HE algorithms outlined below. λ denotes the desired security level of the scheme. *PT* denotes the underlying plaintext space. *K* denotes the dimension of the vectors to be encrypted. *B* denotes an auxiliary parameter that is used to control the complexity of the programs/circuits that one can expect to run over the encrypted messages.

PubKeygen(Params) → SK, PK, EK The public key-generation algorithm is used to generate a pair of secret and public keys. The public key can be shared and used by anyone to encrypt messages. The secret key should be kept private by a user and can be used to decrypt messages. The algorithm also generates an evaluation key that is needed to perform homomorphic operations over the ciphertexts.

SecKeygen(Params) → SK, EK The secret key-generation algorithm is used to generate a secret key. It should be kept private by the user. The algorithm also generates an evaluation key

PubEncrypt(PK, M) → C The public encryption algorithm takes as input the public key of the scheme and any message M from the message space. The algorithm outputs a ciphertext C.

SecEncrypt(SK, M) → C The secret encryption algorithm takes as input the secret key of the scheme and any message M from the message space. The algorithm outputs a ciphertext C.

Decrypt(SK, C) → M The decryption algorithm takes as input the secret key of the scheme, SK, and a ciphertext C. It outputs a message M from the message space.

EvalAdd(Params, EK, C1, C2) → C3. EvalAdd is a randomized algorithm that takes as input the system parameters Params, the evaluation key EK, two ciphertexts C1 and C2, and outputs a ciphertext C3.

EvalAddConst(Params, EK, C1, M2) → C3. EvalAddConst is a randomized algorithm that takes as input the system parameters Params, the evaluation key EK, a ciphertext C1, and a plaintext M2, and outputs a ciphertext C3.

EvalMult(Params, EK, C1, C2) → C3. EvalMult is a randomized algorithm that takes as input the system parameters Params, the evaluation key EK, two ciphertexts C1 and C2, and outputs a ciphertext C3.

EvalMultConst(Params, EK, C1, M2) → C3. EvalMultConst is a randomized algorithm that takes as input the system parameters Params, the evaluation key EK, a ciphertexts C1, and a plaintext M2, and outputs a ciphertext C3.

Refresh(Params, flag, EK, C1) → C2. Refresh is a randomized algorithm that takes as input the system parameters Params, a multi-valued flag (which can be either one of “Relinearize”, “ModSwitch” or “Bootstrap”), the evaluation key EK, and a ciphertext C1, and outputs a ciphertext C2. The correctness property of Refresh is that if C1 is an encryption of plaintext element M1, then C2 should be an encryption of M1 as well.

Previously, there was somewhat homomorphic encryption (SHE) that has a limitation on several operations. In particular, after implement multiplication operation, some restrictions accumulated. However, with several techniques such as relinearization and bootstrapping, we can implement operation with no restriction.

• Relinearization

In the homomorphic operation, when mult operation implement, the ring element increases rapidly unlike add operation. So if we implement the Mult operation several times, it becomes very slow. To solve this problem, after multiplication, a task called "relinearization" that reduces ring elements should be performed. Let's see the add and multiple operations, respectively.

Add operation Let $ct = (c_0, c_1, \dots, c_\delta)$ and $ct' = (c'_0, c'_1, \dots, c'_\gamma)$ be the two ciphertexts. Homomorphic addition is done by simple component-wise addition of the ciphertexts. As a result of implementation, the number of Ring elements is $\max(\delta, \gamma) + 1$.

$$ct_{add} = \left(c_0 + c'_0, c_1 + c'_1, \dots, c_{\max(\delta,\gamma)} + c'_{\max(\delta,\gamma)}\right) \in R_q^{\max(\delta,\gamma)+1}$$

Mult operation Let $ct = (c_0, c_1, \dots, c_\delta)$ and $ct' = (c'_0, c'_1, \dots, c'_\gamma)$ be the two ciphertexts. Let v be a symbolic variable and consider the expression. As a result of implementation, the number of Ring elements is $\delta + \gamma + 1$.

$$ct_{mult} = \left(\Sigma_{i=0}^{\delta} c_i v^i\right) \times \left(\Sigma_{i=0}^{\gamma} c'_i v^i\right) = \Sigma_{i=0}^{\delta+\gamma} \hat{c}_i v^i = \left(\hat{c_0}, \dots, \hat{c}_{\delta+\gamma}\right) \in R_q^{\delta+\gamma+1}$$

As we can see above, homomorphic multiplication operation increases the number of ring elements in a ciphertext. Assume that we run Mult operation on two ciphertexts (each containing two ring elements) produced by the encryption algorithm. Then the resulting ciphertext contains three ring elements. The idea is like that

$$ct_{mlt} = c_2 s^2 + c_1 s + c_0 \rightarrow c_0^{relin} + c_1^{relin} s$$

So, we change $c_2 s^2 \approx c_2(-as - te + t^i s^2 + as) \rightarrow d_0 + d_1 s = bc_2 + ac_2 s$. However, because the coeficent of c_2 is too large as z_q, c_2 is divided to reduce the size of the coefficients. $c_2 s^2 = \left(\Sigma_{\mathrm{i}=0}^{\lceil \log_t q \rceil - 1} c_{2i} t^i\right) s^2 = \Sigma c_{2i}\left(t^i s^2\right) \approx \Sigma c_{2i}\left(-a_i s - te + t^i s^2 + a_i s\right) = \Sigma c_{2i}(b_i + a_i s)$. As a result, we can set two equation. Where $h_i = (a_i, b_i)$ come from "homomorphism key"

$$c_0^{relin} = c_0 + \Sigma_{\mathrm{i}=0}^{\lceil \log_t q \rceil - 1} c_{2i} b_i$$

$$c_1^{relin} = c_1 + \Sigma_{\mathrm{i}=0}^{\lceil \log_t q \rceil - 1} c_{2i} a_i$$

Output the 2-element ciphertext $ct_{mlt} = (c_o^{relin}, c_1^{relin})$. And develop the c_0^{relin}.

$$\begin{aligned} c_0^{relin} &= c_0 + \Sigma c_{2,i} b_i \\ &= c_0 + \Sigma_i c_{2,i}\left(-a_i s - te_i + t^i s^2\right) \\ &= c_0 - \left(\Sigma_i c_{2,i} a_i\right)s - te_{relin} + \left(\Sigma_i c_{2,i} t^i\right)s^2 \\ &= c_0 - \left(c_1^{relin} - c_1 s\right) - te_{relin} + c_2 s^2. \end{aligned}$$

So, we can express

$$\begin{aligned} c_0^{relin} + c_1^{relin} s &= c_0 + c_1 s + c_2 s^2 - te_{relin} \\ &= t(e_{mult} - e_{relin}) + mm' \end{aligned}$$

We looked at how to reduce the ring element through a method called relinearization when doing homomorphic multiplication operation. Although a relinearization error occurs, the product of the two messages can be obtained by taking $mod(t)$. Through this, it is possible to maintain a small ring element even if homomorphic multiplication is implemented.

• Message Encoding

Message encoding is the process of putting multiple messages in each slot into a plaintext that has information about many message. In the past, researchers focused on how to efficiently encrypt and decrypt. However, efficient message encoding and decoding do an important role in performance. So research on this part is also active these day. In here, the encoding method is CKKS scheme's method that capable of decimal operation. The conventional method used the CRT matrix and canonical embedding method, but using these method in the CKKS scheme, the complex number is included in the plain text. To solve this problem, add using extension to make a message into a conjugate form. And then we can make plaintext using CRT matrix that same method before used.

CRT Matrix

CRT Matrix make the function $f(x) = \Sigma_{i=0}^{n-1} a_i X^i$. And this function can also express the matrix.

$$\begin{bmatrix} 1 & \zeta_0 & \cdots & \zeta_0^{n-2} & \zeta_0^{n-1} \\ 1 & \zeta_1 & & \zeta_1^{n-2} & \zeta_1^{n-1} \\ 1 & \vdots & \ddots & \vdots & \vdots \\ 1 & \vdots & & \vdots & \vdots \\ 1 & \vdots & & \ddots & \vdots \\ 1 & \zeta_{n-1} & \cdots & \zeta_{n-1}^{n-2} & \zeta_{n-1}^{n-1} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ \vdots \\ \vdots \\ a_{n-1} \end{bmatrix} = \begin{bmatrix} f(\zeta_0) \\ f(\zeta_1) \\ \vdots \\ \vdots \\ \vdots \\ f(\zeta_{\phi(n)}) \end{bmatrix}$$

This can think like $CRT(\sigma) \times$ Plaintext Coefficient = Evaluation value. Like the relationship between coefficients and points in FFT algorithms. Through the CRT matrix, It can move between the message space and the plate text space. This process is called message encoding and decoding.

Extension Matrix

However, as mentioned briefly above, in the CKKS scheme have a problem that the plaintext coefficient comes out as a decimal point. Using the extensions matrix it can be solved. If we make a conjugate pair of messages and multiply this vector by the CRT matrix, it has a rational coefficient. Then this rational number can be rounded to create a plaintext with an integer coefficient. Extension Matrix π is like that.

$$\pi = \begin{pmatrix} 1 & 0 & 0 & i \\ 0 & 1 & i & 0 \\ 0 & 1 & -i & 0 \\ 1 & 0 & 0 & -i \end{pmatrix} \begin{pmatrix} a \\ c \\ b \\ d \end{pmatrix} = \begin{pmatrix} a+bi \\ c+di \\ c-di \\ a-bi \end{pmatrix}$$

The encoding process is summarized as follows.

Step1 => $\left(m_1, m_2, \ldots m_{\frac{n}{2}}\right) \in C^{\frac{N}{2}}$

Step 2 => $\left(m_1, m_2, \ldots, m_{\frac{N}{2}}, \overline{m}_1, \overline{m}_2, \ldots, \overline{m}_{\frac{N}{2}}\right) \in C^N$

Step 3 => inverse of CRT Matrix * step2's message vector = plaintext coefficient

• Modulus switching and Bootstrapping

The modulus switching operation is intended to reduce the norm of the noise, to compensate for the noise increase that results from all the other operations.

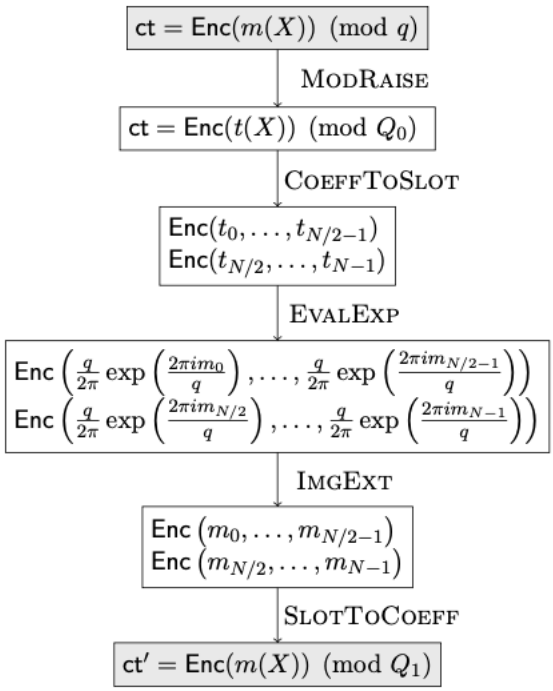
The operation SwitchModulus(c) takes the ciphertext $c = ((c_0, c_1), t, v)$ defined modulo q_t and produces a ciphertext $c' = ((c_0', c_1'), t-1, v')$ defined modulo q_{t-1}. At the above expression, c is ciphertext. t is level of prime q that $0 \leq t \leq L-1$. v is the noise magnitude in this ciphertext. a preset constant B_{scale} which is meant to bound the magnitude of the added noise term from this operation. Modulus switching works as follows:

SwitchModulus $((\boldsymbol{c_0}, \boldsymbol{c_1}), \boldsymbol{t}, \boldsymbol{v})$**:**

1. If $t < 1$ then abort;
2. $v' \leftarrow \frac{q_{t-1}}{q_t}\ v + B_{scale}$;
3. If $v' > q_{t-1}/2c_m$ then abort;
4. $c_i' \leftarrow Scale(c_i, q_t, q_{t-1})$ for $i = 0,1$;
5. Output $((c_0', c_1'), t-1, v')$.

As seen above, the Switch Modulus operation is efficient in reducing norm. However, If user continue to reduce the level of q and then the level becomes 1, user can no longer perform the operation. In order to solve this problem, a technique called bootstrapping that raises the level of q came out. Although bootstrapping methods exist depending on various schemes, I will describe the scheme method of CKKS, which is recently used for machine learning. The overall bootstrapping process is shown in the figure below.

Pipeline of bootstrapping process



Modulus raising Step This is a process that change the smaller q to large Q. The cyphertext $ct = Enc(m(X))(mod\ q) \rightarrow Enc(t(X))(mod\ Q_0)$. $t(X)$ is inner product that $< ct, sk > (mod\ X^N + 1)$.

Coefficient to slot Step this step aims to put the coefficients $t_0, \ldots, t_{N-1}$ in plaintext slots in order to evaluate the modular reduction function $F(t)$ coefficient wisely. Since each ciphertext can store at most $N/2$ plaintext values, we will generate two ciphertexts encrypting the vectors $z'_0 = (t_0, \ldots, t_{\frac{N}{2}-1})$ and $z'_1 = (tN, \ldots, t_{N-1})$, respectively.

Evaluation of exponential function Step To perform this process, approximate the modular reduction function. To approximate this, paper suggest the sin function. If use the whole sin function, it is very different from modulus, so only the front part of the sin function is used. And used the range from $-q/2$ to $q/2$ not from 0 to $q-1$. These can be approximated by sin function, and the range of error bound is reasonable.

Then, the goal is to homomorphically evaluate the trigonometric function $\frac{q}{2\pi} \cdot \sin\left(\frac{2\pi}{q} \cdot t\right)$ with an input $t = \langle ct, sk \rangle$ that inner product. Taylor approximation and Paterson-Stockmeyer method were used for homomorphic evaluate, but there was a problem with efficiency. So paper proposed double-angle formulas, and use this formulas for evaluation.

Imaginary part extraction Step At this step, takes two ciphertexts encrypting the value $\frac{q}{2\pi}\exp(\frac{2\pi i m_j}{q})$ in their plaintext slots for $0 \leq j < N$. And using the relation

$$\sin\left(\frac{2\pi m_j}{q}\right) = \frac{1}{2}(\exp\left(\frac{2\pi i m_j}{q}\right) - \exp(\frac{-2\pi i m_j}{q}))$$

extract their imaginary parts as $\frac{q}{2\pi}\exp\left(\frac{2\pi i m_j}{q}\right) \approx m_j$. Then applying the evaluation method of slot wise conjugation.

Slot to coefficient Step This step is to pack all the coefficients m_j in the plaintext slots of two ciphertexts back in a single ciphertext that is inverse of Coefficient to slot step. When the final procedure is over returns an encryption of $m(X)$ with a ciphertext modulus Q_1, which is much larger enough than the initial modulus q to allow us to perform further homomorphic operations on the ciphertext.

Previously, there was Somewhat Homomorphic encryption (SHE) that has a limitation on several operations. In particular, after implement multiplication operation, some restrictions accumulated. However, with several techniques such as relinearization and bootstrapping, we can implement operation with no restriction. HE with these characteristics is called Fully Homomorphic Encyption (FHE).

• Setting parameter

Homomorphic encryption is lattice-based ciphers. This is based on the difficulty of finding a vector that satisfies a specific condition, if the dimension of the lattice is high. To assess the security, they use primal strategy finds the closest vector to c when $As + e = c \bmod q$. Attacker tries to break a security

by applying a lattice reduction algorithm. To prevent this, it is necessary to understand some variables and technique.

Error Distribution

Chosen from a sufficiently wide and "well spread" distributions, we do not have meaningful attacks on RLWE that are better than LWE attacks, regardless of the ring. For power-of-two cyclotomics, it is sufficient to sample the noise in the polynomial basis $e \in Z[x]/\Phi_k(x)$ independently at random from a very "narrow" distribution. If the cyclotomics is not power of 2, the error can be extracted using the DD12 method or the CP16 method. However, since there are many conveniences point to use power of 2 cyclomics, it is usually set to power of 2.

Secret Key

Choosing the key from the same distribution as the error does not weaken the scheme. Choosing an even smaller secret key has a significant performance advantage. There are three choices of secret distribution: uniform, the error distribution, and ternary. But for good performance, usually choose ternary distribution(i.e., each coefficient is chosen uniformly from {-1,0,1}).

Number of Samples

For most of the attacks listed in the tables below, the adversary needs a large number of LWE samples to apply the attack with maximum efficiency. Collecting many samples may be feasible in realistic systems, since from one ring-LWE sample one can extract many "LWE-like" samples.

Sampling Method

All the error distributions require choosing the coefficients independently at random from either the discrete or the continuous Gaussian with some standard deviation $\sigma > 0$. Sampling continuous Gaussian is quite straightforward, but sampling from a discrete Gaussian distribution is harder. There are several known methods to sample from a discrete Gaussian, but recommended the Von Neumann-type sampling method. It can easily be adapted to sample exactly from the discrete normal distribution whose parameters are rational numbers.

Constant-Time Sampling

Attacker can figure out information by the sampling time it takes. In many applications, it is therefore important that the entire error-sampling process is constant-time. To prevent this problem, fix some upper bound $T > 0$ such that sampling all the n coordinates e_i sequentially without interruption takes time less than T time with overwhelming probability. Then after these samples are generated, using time t, we wait for $(T - t)$ time. In this way, it is possible to prevent information from leaking by maintaining a constant sampling time of T.

Post-Quantum Security

Recently quantum computers appear, many security-related concerns are being raised. The homomorphic encryption is post-quantum security that is safe for quantum attacks. To guarantee security, we need to set the parameters more conservatively.

Table

distribution	n	security level	logq	uSVP	dec	dual
uniform	1024	128	29	131.2	145.9	161.0
		192	21	192.5	225.3	247.2
		256	16	265.8	332.6	356.7
	2048	128	56	129.8	137.9	148.2
		192	39	197.6	217.5	233.7
		256	31	258.6	294.3	314.5
	4096	128	111	128.2	132.0	139.5
		192	77	194.7	205.5	216.4
		256	60	260.4	280.4	295.1
	8192	128	220	128.5	130.1	136.3
		192	154	192.2	197.5	205.3
		256	120	256.5	267.3	277.5
	16384	128	440	128.1	129.0	133.9
		192	307	192.1	194.7	201.0
		256	239	256.6	261.6	269.3
	32768	128	880	128.8	129.1	133.6
		192	612	193.0	193.9	198.2
		256	478	256.4	258.8	265.1

The error is extracted from Gaussian distribution. There are three methods of extracting secret keys (uniform, error distribution, ternary). This table is part of the uniform distribution method, and the value of $logq$ are slightly different by choosing method.

Set the dimension and desired security level. Then, the logq value can be determined by referring to the table. But at the key switching process, the value of q raise. So we consider that max value of q when setting parameter q. If you do not want to increase the q value further, there is also a method of keeping the q value low by adjusting the maximum value of q by reducing the depth during the bootstrapping process. uSVP, dec, dual are representative lattice attack methods, each indicating how much run time is required. It can be seen that as the security level goes up, the amount required increases.

2) Discussion

I summarized the basic principles and operations of isomorphic encryption, as well as various techniques. It was difficult to commercialize HE due to problems such as limitation of number of calculations(Multiplication) and running time. But these problems have been resolved to some extent as various techniques have been introduced and optimization. In the past, it focused on data protection rather than data use. In the future, as industries such as metaverse develop, technologies that can maintain security and utilize data will be needed.

Homomorphic encryption has developed from restricted number of operation to without restricted number of operation, from integer range to rational range. As such, it is expected that there will be schemes that support all of the four fundamental arithmetic operations or more faster schemes in the future. As mentioned above, HE technology is a promising technology because it can be used in more diverse fields as technology develops.

※ References

[1] Naehrig, M., Lauter, K., & Vaikuntanathan, V. (2011, October). Can homomorphic encryption be practical?. In *Proceedings of the 3rd ACM workshop on Cloud computing security workshop* (pp. 113-124).

[2] Lauter, Kristin, Wei Dai, and Kim Laine. "Protecting Privacy through Homomorphic Encryption."

[3] Gentry, Craig, Shai Halevi, and Nigel P. Smart. "Homomorphic evaluation of the AES circuit." *Annual Cryptology Conference*. Springer, Berlin, Heidelberg, 2012.

[4] Cheon, Jung Hee, et al. "Homomorphic encryption for arithmetic of approximate numbers." *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, Cham, 2017.

[5] Cheon, Jung Hee, et al. "Bootstrapping for approximate homomorphic encryption." *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, Cham, 2018.