UDC: 004.056.55

Ramirez
Mexico, Guadalajara,
Operaciones Digitales y
Procesamiento Integral
de Datos Encriptados

Программирование случайного изменения переменных для Гомоморфное шифрование

Описана схема гомоморфного шифрования. Данные шифруются через случайное изменение переменных, а затем операции выполняются на зашифрованном тексте, который одновременно обрабатывается и расшифровать результат. Это улучшает традиционный гомоморфный Шифрование путем объединения двух этапов в один, Гарантия того, что данные могут быть использованы только по назначению. Шум и точность управляемы, без большой эффективности Компромиссы. Двухпартийная Схема между Клиентом и Банком была разработана в качестве Доказательства-Концепция. Банк не имеет доступа к входным данным, представляющимКлиентские данные, но способен обрабатывать зашифрованные векторы. Тем Метод является гибким и может быть адаптирован для разного числа сторон, управление разрешениями, уровни безопасности, оперативность требования и т.д. Гомоморфный энкрипДоказательство с нулевым разглашением, безопасность данных, машинное обучение.

I. INTRODUCTION

Traditionally, if one must process encrypted data, first the data is decrypted, and then the data is processed. This exposes the data to the second, and potentially third parties, in many instances. Homomorphic Encryption [1], [2] addresses this issue by changing the order of these two steps by first processing the data, while still encrypted, and then decrypting the result to plaintext. Machine Learning and AI applications that require sharing mass amounts of sensitive data, smart grids, large networks, traffic control, electronic voting, energy management, among many others, can be implemented if certain privacy issues are solved [3-7]. Let E a function that encrypts natural numbers. Suppose we encrypt two numbers x, y to obtain two new numbers Ex, Ey . We won't worry right

now about which space the ciphertext is defined in, and suppose we have an operation $\bigoplus$ defined in that space. The operation of these gives $Ex \bigoplus Ey$. For most encryption functions, this is not equal to $E(x \bigoplus y)$. Therefore, decrypting $Ex \bigoplus Ey$ does not yield the expected result $x \bigoplus y$. If there exists a computable function D such that $x + y = D(Ex \bigoplus Ey)$, for any choice of $x, y \in \mathbb{N}$, we have a Partially Homomorphic Encryption Scheme for Addition (+). If the same property is also satisfied for multiplication (·), then it is a Fully Homomorphic Encryption Scheme. The first difficulty in HE is that an operation and an encryption function are almost never homomorphic, $E(x + y) \neq Ex \bigoplus Ey$. Mathematical homomorphisms that can be used for HE are not practical solutions because of the complicated structures involved. Consequently, precision and noise are not easily mitigated [8], [9] making the algorithms unpractical or energy inefficient [10], [11], [12] in many scenarios. We explore a Keyless Decryption method that merges processing and decryption into a single step, avoiding Bootstrapping techniques. In this case, the processing function is the decryption key. The first party encrypts the inputs by applying a random change of variable from a library of encryption functions. The second party possesses a corresponding library of process/decryption functions and determines which of these functions will process/decrypt the result. Applying any other function of the library yields a meaningless answer. We describe an encrypted Credit Score calculation for a personal one-year loan as a Proof of Concept.

The commutative diagram for representing a HE Scheme, in its simplest conceptual form, is given by the equation $P = D \circ P^* \circ E$ shown below (1), where $P: A \to B$ is the process and P^* is the process applied to the encrypted data.

$$\begin{array}{ccc} A & \xrightarrow{P} & B \\ {\scriptstyle E}\downarrow & & \uparrow{\scriptstyle D} \\ A & \xrightarrow{P^*} & B \end{array} \tag{1}$$

This diagram illustrates the basic relation for Encrypting the domain of the function and decrypting the result. Applying process P to the plaintext is equivalent to encrypting, then applying P^* and finally decrypting. We propose an HE scheme based on the concept of change of variable. Suppose a calculation $P: \mathbb{R}^n \to \mathbb{R}$ must be applied to an ordered n-tuple of variables $x = (x_i)_{i=1}^n = (x_1, x_2, \dots, x_n) \in \mathbb{R}^n$. We encrypt these n variables by applying a computable function $E: \mathbb{R}^n \to \mathbb{R}^m$ to the ordered n-tuple. An encryption function of P is a computable function $E: \mathbb{R}^n \to \mathbb{R}^m$ if there exists a computable function $P^*: \mathbb{R}^m \to \mathbb{R}$ such that $Px = (P^* \circ E)x$. In this case we say P, P^* are an ordered pair of homomorphic functions [13] with respect to encryption function E.

II. ENCRYPTION BY CHANGE OF VARIABLE

Suppose $P: \mathbb{R}^n \to \mathbb{R}$ is a function of n independent variables, and one party wishes for a second party to compute $P(x_1, x_2, \dots, x_n)$ without having the capability of knowing the input vector $(x_1, x_2, \dots, x_n)$. Let $E: \mathbb{R}^n \to \mathbb{R}^n$, defined by functions $e_1, e_2, \dots, e_n: \mathbb{R}^n \to \mathbb{R}$ such that

$$P \circ E = D^{-1} \circ P \tag{2}$$

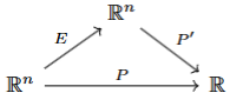
for some invertible function $D^{-1}: \mathbb{R} \to \mathbb{R}$.

$$\begin{array}{ccc} \mathbb{R}^n & \xrightarrow{P} & \mathbb{R} \\ {\scriptstyle E}\downarrow & & \downarrow{\scriptstyle D^{-1}} \\ \mathbb{R}^n & \xrightarrow{P} & \mathbb{R} \end{array}$$

Suppose the change of variables E expresses $P \circ E$ in terms of an invertible function D^{-1}, of P. If the inverse function D is computable then we have encryption and decryption functions E, D, respectively. For example, let $P: \mathbb{R}^2 \to \mathbb{R}$ be the function $P(x, y) = x + y^2$, and let $e_1(x) = a^2x^2 + 2a^2xy^2$ and $e_2(y) = ay^2$, for some parameter $a \in \mathbb{R}$. Then

$$(P \circ E)(x, y) = e_1(x) + \big(e_2(y)\big)^2 = a^2(x + y^2)^2 = a^2\big(P(x, y)\big)^2.$$

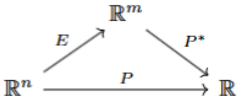
We have found $P \circ E$ to be expressed in terms of P, namely $(P \circ E)(x, y) = a^2\big(P(x, y)\big)^2$. In this case, the function D^{-1} is given by $D^{-1}(x) = a^2x^2$. Therefore, the decryption function is given by $D(x) = \frac{\sqrt{x}}{a}$. This implies $P = D \circ P \circ E = \frac{1}{a}(P \circ E)^{\frac{1}{2}}$. Let us understand what this relation means, and what it permits. Suppose we wish for a second party to compute $P(x_0, y_0) = x_0 + y_0^2$, for constants $x_0, y_0 \in \mathbb{R}$, but we do not wish to share the plaintext vector (x_0, y_0) with the second party. First, we encrypt the inputs, using the function E described above, to obtain $E(x_0, y_0) = \big(e_1(x_0, y_0), e_2(x_0, y_0)\big)$, and send this encrypted vector to the second party. The second party will then take this encrypted vector and apply function $P' = D \circ P$ to obtain the desired result. We collapse (1) into a three diagram. If we consider the composition $P' = D \circ P$ as a single function, the resulting relation is $P = P' \circ E$.



(3)

Can the second party decrypt the encrypted vector to find the original inputs x_0, y_0 with the information available? The information available is 1) Processing function P, 2) Encryption function E and Encrypted Variables e_1, e_2, and 3) Final result p_0. From 1) and 3) we have equation $x_0 + y_0^2 = p_0$, where $p_0 = (P' \circ E)(x_0, y_0)$. Two more equations are given by $a^2x_0^2 + 2a^2x_0y_0^2 = e_1$ and $ay_0^2 = e_2$. A solution for a, x_0, y_0, of this system of equations must be found. If we find a solution for a, we can then calculate x, y. Let us specify the bit size of the values in question. We have 64-bit inputs x_0, y_0. We also have a 128-bit encryption key, a. This key is unknown to the second party, otherwise it could easily decrypt e_1, e_2 to find x_0, y_0. For purposes of finding solutions to this system of equations the key a is an unknown integer of 128 bits. Finding the original inputs is equivalent to finding the key a. If the second party knows the inner workings of the scheme, they would know to start by finding the factors of the 256-bit number e_2. Now that we understand where vulnerabilities arise, we can ask a few questions to maximize security. Let us start by asking if a type of integer is optimal for the security parameter? For example, if a is always chosen to be a prime number, then finding a is trivial. To find a, you must simply find the factors of $e_2 = ay^2$ that are smaller than 64-bits. Once we have the 128-bit key a, we can calculate y and then x. Thus, the key should not be chosen to be a prime number. The more prime factors in a, the better. For every encryption function E, of some processing function P, we have different security parameters. In each case, the keys have different required forms to maximize security.

We give a definition of encryption and decryption functions that supersedes the previous ones. If, more generally, given an encryption function $E: \mathbb{R}^n \to \mathbb{R}^m$ we can find a computable function $P^*: \mathbb{R}^m \to \mathbb{R}$ such that $P = P^* \circ E$, then we have an encryption scheme for process P.

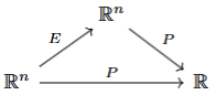


There is no straightforward method for finding the most secure and efficient encryption and processing functions E, P^*, for a given process P. We use the fact that certain encryption functions are safe (the keys, and therefore the plaintext inputs, cannot easily be found). Even if the encryption and processing functions E, P^* are known, finding the keys is a very difficult problem. One way to ensure the encryption function remains unknown is to have a large library of encryption functions for each processing

function, and then randomly select an encryption function each time. This solution is discussed in Sections 6 and 7. The method proposed here allows for an FHE scheme that can be adapted for different number of parties, permission configurations and security needs. In Section 5, we outline a simple two-party scheme to simulate a personal loan Credit Score where the first party is a Client, and the second party is a Bank. The Client will encrypt the inputs and share them with the Bank. The Bank receives these encrypted values, processes the encrypted data and sends the result back to the Client.

III. KEYLESS DECRYPTION

We will illus trate a special case of relation (3). Let $P(x,y,z) = \frac{x^2+y^2}{z}$. If we make the change of variables $e_1(x) = ax,\ e_2(y) = ay,\ e_3(z) = a^2z$, then we have $P = P \circ E$. Again, $P \circ E$ is a function of P . In this case $D^{-1} = I$ is the identity function. This example of $P(x,y,z) = \frac{(ax)^2+(ay)^2}{a^2z}$ cancels nicely with the encryption function, so that $D = I$ is the identity function. When this happens, we say this is a Keyless Decryption Scheme because $P' = I \circ P = P$. In this case, the function that must be applied to the encrypted data is the same function that would be applied to the original plaintext. No new function has be defined because, $P' = P$.



This example of Keyless Decryption is not semantically secure for several reasons, but non-trivial and semantically secure examples of Keyless Decryption can be found for certain functions. In this example, if any of x, y, z is zero, then its corresponding Encrypted Variable will be zero. Secondly, the bit-length of the Variables is easily deduced from the Encrypted Variables, which makes it useless for almost any application. Ultimately, the encryption function of this example does not work for the following reason. It is easy to find Key a , to calculate the Inputs. The Key is the number that divides e_1, e_2 once and divides e_3 two times. There are different techniques for defining secure encryption functions and keys without recurring to excessive bit-lengths, because there are encryption functions whose equations do not determine the keys easily. The solution presented in Sections 6 and 7 uses several strong encryption functions, and randomly chooses a different one each time, adding an extra layer of security. Given a large library of encryption functions $\{E_j\}_j$, for P , each encryption function E_j is associated to a unique processing function P_j^* .

V. MATHEMATICAL MODEL

We define encryption and processing functions that yield a decrypted plaintext Credit Score, to be shared with the Client. The Credit Score function is normalized. If the income to expense ratio and the worth to debt ratio go to infinity, the Credit Score is 10. If, on the other hand, the expenses and debt grow with respect to income and worth, the Credit Score is 0. The minimum approval score is suggested at 5 points, given the current settings and weights of the parameters, but these can be modified for different cases. A reasonably calibrated model for loan approvals based on a client's financial data (income, capital, debt, expenses, etc.) is proposed. Parameters can be modified to adapt the Credit Score model to different scenarios (different types of loans such as micro loan, business loan, loan period, interest, etc.). The proposed processing function P is

$$P(NI, TE, W, TD) = \frac{A\,NI}{\sqrt{NI^2 + \alpha\,TE^2}} + \frac{B\,W}{\sqrt{W^2 + \beta\,TD^2}}$$

where A, α, B, β are parameters, which we will set to numeric values to calibrate our model. There are other parameters such as the interest rate and loan period which will be set to 15% interest rate for a one-year loan, and which are used to calculate the Newly Incurred Debt, Service to Debt, Total Debt and Total Expenses (including the new loan).

VI. RANDOMIZING METHODS

We can add another layer of security. Suppose we have a library of k different encryption functions $\{E_j\}_{j=1}^{k}$, of a given processing function P. If the client encrypts the variables using E_i, for some $E_i \in \{E_j\}_j$, then the encrypted variables will have to be processed with the corresponding processing function P_j^*. Furthermore, the codimension of E_j can be different from another encryption functions codimension. It is possible to implement a randomized selection of the encryption functions that adds an extra layer of security. After transforming the inputs into the variables, in Module I, the variables are sent to Module II. Now, when Module II receives the variables, it chooses a method from the library of encryption functions $\{E_j\}_j$, to encrypt the variables. Once j is fixed, the vector of variables in $\mathbb{R}^n$ will be transformed into a new vector of encrypted variables. The bank has a corresponding library of processing functions $\{P_j^*\}_j$. Upon receiving the vector of encrypted variables, the bank will be able to identify and run the correct processing function. Having a large library of encryption methods is beneficial for the security of the overall scheme because the data type and the number of encrypted variables can be different each time process P has to be applied to a vector x, making it harder to implement pattern recognition. A basic version of the personal loan Credit Score, detailed above, has been implemented and will be made available for auditing purposes, as well as for establishing collaboration opportunities.

VII. GENERALIZATION

In many instances it is desirable to have data encrypted in such a way that it can then be re-encrypted for use in one of many different processing functions. Consider a one-party scheme where the User has a numerical database $\mathcal{E}(DB)$ encrypted at rest and wishes to perform operations on elements of the database but wants to maintain the data confidential. The finite library of functions for processing the data is $\{P\}_P = \{R, S, \dots, T\}$. The User requests for any of the functions to be applied to a vector of the database. Suppose we have a library of at least two processing functions, where $R(x, y) = x + y$ and $S(x, y) = xy$. Every processing function $P \in \{P\}_P$ has its own library of encryption functions. The encryption libraries are $\{E_{R,i}\}_i = \{E_{R,1}, E_{R,2}, \dots\}, \{E_{S,j}\}_j = \{E_{S,1}, E_{S,2}, \dots\}, \dots$. Each encryption library has its own cardinality. The User sends a request to the database which includes an encrypted index for a processing function $P: \mathbb{R}^n \to \mathbb{R}$, and the encrypted addresses of the elements of the database. The request is a vector of $n + 1$ coordinates $(\#P, x_1, x_2, \dots, x_n)$. The first coordinate of the request is an index, $\#P$, indicating to the database the processing function. The other n coordinates of the request are the addresses of the elements of the database. Then, the database randomly selects an encryption function $E: \mathbb{R}^n \to \mathbb{R}^m$ from the set $\{E_{P,j}\}_j$. The database has a library of processing functions $\{P^*_E\}_{P,E} = \{R^*_{E_{R,1}}, R^*_{E_{R,2}}, \dots, S^*_{E_{S,1}}, S^*_{E_{S,2}}, \dots\}$, for all $P \in \{P_j\}_j$ and all $E \in \{E_{P,j}\}_j$. The processing function P^*_E depends on the processing function P and the encryption function E , so that if we change the process P then the encrypted processing function P^*_E is also different. Similarly, if we change the encryption function E then the function P^*_E changes again. The database identifies and applies the correct function P^*_E to the other n encrypted values.

VIII. CONCLUSIONS

Although HE is a promising concept in the science of cryptography, the same reasons that make it safe also make it non applicable in many situations. Current energy and time efficiency standards of HE are not met in a wide range of crucial applications. This change of variable method offers many advantages including the fact that the encrypted data can only be used for the intended purposes, and noise and precision are managed without significant efficiency trade-offs. This scheme is applicable to a wide range of processing functions, including addition and multiplication from which a FHE scheme can be constructed. Trigonometric functions and the numeric derivative can also be homomorphically encrypted. It is a flexible scheme that can be adapted to different number of participants, permission configurations, security demands, efficiency requirements, etc. These characteristics make it a far and wide reaching solution with applications in online security, private Digital Signal Processing, smart cities and traffic

problems, Machine Learning such as in specific cases of AI and Neural Network training utilizing sensitive data, operating vectors of encrypted databases, smart grids and energy management, among other activities directly dependent on the difficult marriage of data security and efficient computing. The proposed method can be integrated to SoC by implementing this encryption scheme in a processor architecture with a Simple and Linear Fast Adder having a linear and scalable topology [14], [15], [16]. Encrypted Processing Units can help achieve secure cloud computing without sacrificing efficiency at hardware and software level.

BIBLIOGRAPHICAL LIST

[1] *Ronald L. Rivest, Len Adleman, and Michael L. Detouzos.* On data banks and privacy homomorphisms. In Foundations of Secure Computation, pages 165–179. Academic Press, 1978. Available at https://people.csail.mit.edu/rivest/pubs.html#RAD78.

[2] *C. Gentry*, (2009). "A Fully Homomorphic Encryption Scheme," Doctoral Dissertation, Symposium on the Theory of Computing, NY, New York, USA, 2009.

[3] *Nathan Dowlin, Ran Gilad-Bachrach, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing*. CryptoNets: "Applying Neural Networks to Encrypted Data with High Throughput and Accuracy." In 33rd International Conference on Machine Learning (ICML 2016), volume 48 of Proceedings of Machine Learning Research, pages 201–210. PMLR, 2016. URL: http://proceedings.mlr.press/v48/giladbachrach16.html.

[4] *Florian Bourse, Michele Minelli, Matthias Minihold, and Pascal Paillier*. "Fast Homomorphic Evaluation of Deep Discretized Neural Networks." In Advances in Cryptology – CRYPTO 2018, Part III, volume 10993 of Lecture Notes in Computer Science, pages 483–512. Springer, 2018. doi:10.1007/978-3-319-96878-0 17.

[5] *Christina Boura, Nicolas Gama, Mariya Georgieva, and Dimitar Jetchev*. "Simulating Homomorphic Evaluation of Deep Learning Predictions." In Cyber Security Cryptography and Machine Learning (CSCML 2019), volume 11527 of Lecture Notes in Computer Science, pages 212–230. Springer, 2019. doi:10.1007/ 978-3-030-20951-3 20

[6] *Marcelo Blatt, Alexander Gusev, Yuriy Polyakov, and Shafi Goldwasser*. "Secure Large-Scale Genome-Wide Association Studies Using Homomorphic Encryption." Cryptology ePrint Archive, Report 2020/563, 2020. https://ia.cr/2020/563.

[7] iDASH secure genome analysis competition. http://www.humangenomeprivacy.org.

[8] *Martin R. Albrecht, Rachel Player, and Sam Scott.* "On the concrete hardness of learning with errors." Journal of Mathematical Cryptology, 9(3):169–203, 2015. doi:10.1515/jmc-2015-0016.

[9] *Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé*. "Classical hardness of learning with errors." In 45th Annual ACM Symposium on Theory of Computing, pages 575–584. ACM Press, 2013. doi:10.1145/2488608.2488680.

[10] *Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachéne.* TFHE: Fast fully homomorphic encryption over the torus. Journal of Cryptology, 33(1):34–91, 2020. Earlier versions in ASIACRYPT 2016 and 2017. doi:10.1007/s00145-019-09319-x.

[11] *Léo Ducas and Daniele Micciancio.* “FHEW: Bootstrapping Homomorphic Encryption in Less than a Second.” In Advances in Cryptology-EUROCRYPT 2015, Part I, volume 9056 of Lecture Notes in Computer Science, pages 617–640. Springer, 2015. doi:10.1007/978-3-662-46800-5 24.

[12] *Joppe W. Bos, Kristin E. Lauter, et. al.* Improved security for a ring-based fully homomorphic encryption scheme. In Cryptography and Coding (IMACC 2013), volume 8308 of Lecture Notes in Computer Science, pages 45–64. Springer, 2013. doi:10.1007/978-3-642-45239-0 4.

[13] *Ramirez, J. 2015.* Systems and Categories. arXiv:1509.03649v5 [math.CT]

[14] *Ramirez, J. “Simple* and Linear Fast Adder of Multiple Inputs and Its Implementation in a Compute-In-Memory Architecture,” 2024 International Conference on Artificial Intelligence, Computer, Data Sciences and Applications (ACDSA), Victoria, Seychelles, 2024, pp. 1-11, doi:10.1109/ACDSA59508.2024.10467957.

[15] *Ramirez, J.* “SIMPLE AND LINEAR FAST ADDER,” WIPO, Patentscope. Publication Number: WO/2023/220537. Publication Date: 16/11/2023. Applicant’s name: Juan Pablo Ramirez

[16] *Ramirez, J.* 2023. “Canonical Set Theory with Applications from Matrix Operations and Data Structures to Homomorphic Encryption”. Author’s personal homepage: www.binaryprojx.