https://doi.org/10.31449/inf.v47i10.4586 Informatica **47** (2023) 41–52 **41**

An Approach for Privacy Preservation Assisted Secure Cloud Computation

V. Swathi[1], M.P.Vani[2]
[1]Research Scholar, SCOPE, Vellore Institute of Technology, Vellore, India
[2]Associate Professor, SITE, Vellore Institute of Technology, Vellore, India
E-mail: swathivelugoti@gmail.com, mpvani@vit.ac.in

Keywords: cloud and data security, privacy, outsourced computation, key management center, homomorphic computations

Received: December 27, 2022

Cloud computing has emerged as a cornerstone for digital transformation, offering a cost-efficient, scalable, and geolocation-neutral infrastructure for managing client data. This paradigm enables clients to offload computational and storage demands to third-party service providers, incurring charges solely for the consumed services. It provides on-the-fly network connectivity to a tailored array of scalable computational assets. Such assets can be provisioned and decommissioned swiftly, necessitating minimal managerial oversight and maximizing operational efficacy. A salient feature of cloud technology is the ability to outsource computations. This shifts the computational burden from clients' resource-restricted devices to the vast computational expanse of the cloud. By embracing outsourcing, clients not only achieve significant temporal and financial efficiencies but also tap into boundless computational prowess on a usage-based billing model, all while sidestepping software and hardware upkeep and operational burdens. Nonetheless, privacy remains a daunting challenge. Conventional encryption techniques ensure data confidentiality, but simultaneously curtail data utility, undercutting the potential economic gains from leveraging public cloud offerings. Operating on encrypted data has long been a cryptanalytic conundrum. To navigate this challenge, our research introduces a secure architecture and an oracle for query vector outsourcing, underpinned by privacy-preserving homomorphic encryption. We delve into a thorough empirical evaluation of our proposed model, scrutinizing its computational and security dimensions. This paper also delineates the outcomes of our empirical investigations and delves into their broader ramifications.

Povzetek: Raziskava uvaja metodo oz. arhitekturo za ohranjanje zasebnosti s homomorfnim šifriranjem pri porabo oblaka.

1 Introduction

The industry has recently undergone immeasurable change as a result of the development of cloud security. Customers are utilising this increased flexibility that cloud-based solutions provide because of the numerous advantages they provide for all parties. Even while most individuals recognise the need of a safe IT environment, they frequently overlook important security issues. Organizations may access infrastructure, platforms, and software offerings via efficient pay-as-you-go methods. Through the use of cloud computing, businesses are releasing capital, streamlining centric IT maintenance, modernising and scaling the business-driven approaches, incorporating security along with the flexibility into certain services as well as solutions, assisting clients in novel ways, and expanding their operations in an ever-evolving segment. There are several advantages to cloud security: it eliminates the need to purchase separate hardware [22]-[29]. There exist mainly three cloud computing deployment models i.e., Private centric cloud, Public centric cloud, Hybrid centric cloud. In the domain of Abstract computational algebra, primarily Homomorphism depicted as a mapping between the certain domain as well as periodic range of an constructive algebraic-oriented set that maintains all algebraic structures. A map is a certain function (an specific operation) that accepts input from a collection of the domains and returns an specific element with-in a certain range (i.e. addition as well as multiplication). Homomorphism is a sort of encryption that is used in cryptography. Traditionally, encryption has been seen as a critical tool for safeguarding the privacy of any sensitive data. Conventional encryption techniques, on the other hand, cannot function on encrypted data without first decrypting it. As a result, clients must give up their privacy in order to access cloud-based services such as file sharing, storage, and collaboration. Clients' main privacy issue, though, is this. The general privacy homomorphism scenario is shown as Figure-1. In fact, all the HE schemes then can be neatly categorized into specifically three types:
- *Partially (Moderate) Homomorphic driven Encryption (PHE)*. permits only single variety of operation (either additive or multiplicative) but infinite amount of system calls.

- *Somewhat (Tolerable) Homomorphic Encryption (SHE)*. permits few type of computational operations possessing

certain (limited) amount of times.

- *Fully (Totally) Homomorphic oriented Encryption (FHE)*. permits infinite number of computational operations possessing infinitely many cardinal of system calls.
For conceiving a cryptosystem that mainly allows certain homomorphic driven evaluation of the arbitrary natured functions, it is adequate for allowing only the addition as well as multiplication oriented operations. Remarkably, any certain boolean circuitry can be then represented exploiting only the *XOR (addition)* as well as *AND (multiplication)* oriented gates. Different implementations so far, has proved that fully homomorphic encryption still need to be ameliorate to become more practical across all platforms.

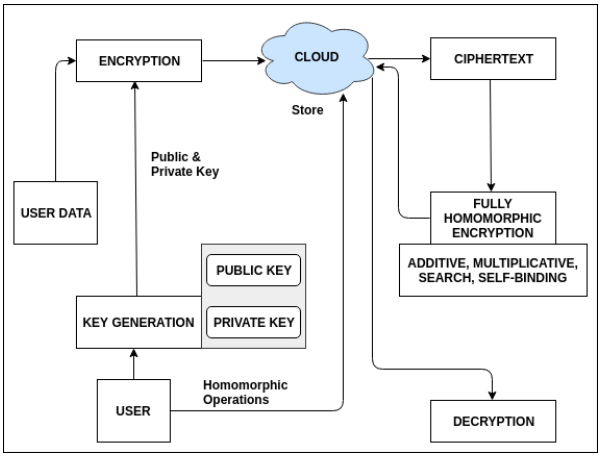

Figure 1: Privacy homomorphism scenario for cloud data security

The challenge of constructing a fully homomorphic encryption (FHE) system remained unresolved for over three decades. Various classifications of homomorphic encryption exist, including partially homomorphic encryption (PHE), somewhat homomorphic encryption (SHE), levelled fully homomorphic encryption (LFHE), and fully homomorphic encryption (FHE). Computations are typically represented using either Boolean or arithmetic circuits. PHE allows evaluation of circuits containing only one type of gate—either addition or multiplication, but not both. SHE, on the other hand, supports computations on circuits with both types of gates but is limited to a specific subset of these circuits. LFHE facilitates the computation of arbitrary circuits but only up to a predefined depth. In contrast, FHE supports the evaluation of arbitrary circuits regardless of their depth, making it the most powerful and versatile form of homomorphic encryption.

1.1 Contribution highlights

The following are the highlights of this paper's contributions:-

– This paper presents a novel approach to address the crucial issue of privacy preservation while outsourcing sensitive information to the cloud for scientific computations. The proposed method employs a secure oracle based on privacy homomorphism to protect the confidentiality of query vectors.

– The paper introduces a robust and efficient secure oracle that enables the outsourcing of query vectors while maintaining data privacy. By leveraging privacy homomorphism, the proposed oracle ensures that the cloud can perform computations on encrypted data without compromising sensitive information.

– The paper includes a comprehensive empirical analysis of the proposed prototype, covering critical aspects such as computational complexity, security, and correctness. The evaluation demonstrates the effectiveness and practicality of the proposed solution.

– The adopted prototype allows specified types of computations to be performed directly on encrypted ciphertext. This capability ensures that the cloud can process data without requiring access to plaintext, enhancing data security.

– The experimental results confirm that the encrypted outcomes achieved from the proposed computations can be accurately decoded to match the results of equivalent plaintext operations. This verification ensures the correctness and reliability of the privacy-preserving computations.

– The paper highlights the real-world applicability of the proposed solution, showcasing its potential to protect sensitive information during cloud-based scientific computations without sacrificing performance or security. It discusses potential avenues for further research and extension of the proposed approach. These include exploring more advanced privacy-preserving techniques, scalability to larger datasets, and integration with emerging cryptographic protocols.

Justifying the expansion on potential applications, especially in real-world cloud computing scenarios, is essential for a comprehensive understanding and applicability of the method. By delving deeper into practical applications, this research not only demonstrates the versatility and relevance of the method but also showcases its tangible benefits. Existing cloud services operate in diverse environments with varying data privacy needs.

1.2 Organization order of the paper

Remaining portions of this research paper are structured as - Section 2 overview previous works done. Section 3 presents some significant definitions and technical preliminaries. Our adopted technological framework along with the stepwise algorithmic methodological procedure is provided in the Section 4. Section 5 discusses the empirical-centric analysis of this proposed system. Experiments re-

sults and discussion is provided in the section 6. Section 7 provides the conclusive summary.

2 Related work

This section summarizes the significant developments in this domain happened over past years. Peeter Laud et al. [12], in their work, discuss about some flaws in the existing security definitions which are shown in recent attacks and further talked about the methods for securely implementing scientific problem solving algorithms. Lifei Wei et al. [13] proposed SecCloud which provides secure storage as well as secure computation auditing together inside the cloud. SecCloud achieves the batch verification, probabilistic sampling methods as well as privacy cheating discouragement by designated the verifier signature. Xiaofeng In a two untrusted programme scenario, Chen et al. [14] suggested a novel safe outsourcing approach for exponentiation modulo a prime. Frederik A. et al. [15], in their work, address three fundamental questions - One is about the fully homomorphic encryption (the theoretical terminology, originally proposed in 1978 by Rivest et al. [1], and later a major breakthrough in terms of its practicality by Craig G. [8], in 2009), next is - for what purpose this phenomenon can be used and lastly, authors thrown light on the state of art on FHE today.
Jannatul F. et al. [16] given a new protocol which is contracting out a large-scope Linear natured Fractional-oriented Programming (LFP) task to a less secure cloud is secure, traceable, and economical. Jianfeng Wang and X. Chen [17] focussed on the methods of verifiable, economical data storage along with the secure data deduplication. Kristian G. et al. [18] discussed that the fully homomorphic-oriented schemes are not possible for some algebraic structures. Dario F. et al. [19] attempted for secure and more flexible Group Password-Based Authenticated Key Exchange (GPAKE). Aayush Jain et al. [20] have discussed the prime construction of threshold fully (totally) homomorphic encryption for any variety of access structure which is persuaded by a monotone oriented boolean formula. Chen T. et al. [21] have given a new new key establishment protocol for Internet of Things (IoT). This scheme utilizes Kronecker product phenomenon. Bo Zhang et al. [24] focuses on sorting and classifying reviewing the status of existing techniques for ensuring the probity of SQL query oriented evaluation results in the particular DaaS paradigm. Yanguo P. et al. [25] proposed the secure the near approximate k-Nearest Neighbour query-module over the encoded format high-dimensional natured data. Ahmed El-Yahyaoui et al. [26] given an scheme, applicable towards scientific problem outsourcing to cloud in a secure manner, based on the new mathematical structure (Quaternionique matrices) that is certainly noise free. Multi-party computing and homomorphic encryption were presented by L.Wang et al. [27]. The multi-cloud architecture solves the problem of permanent failure as well as vendor level lock-in. This strategy has shown to be extremely accessible and cost-effective. V. F. Ramesh et al. provided a cloud-casted digital picture oriented locker utility with enough secure client's identification along with a unique image oriented cryptologic for maintaining the concealment of client photos in their work [28]. Homomorphic satisfiable properties of some significantly well-notorious HE schemes is compiled in Table-1.

Schemes	Operations
RSA (1978) [1]	MULT
GM (1982) [2]	ADD
El-Gamal (1985) [3]	MULT
Benaloh (1994) [4]	ADD
NS (1998) [5]	ADD
OU (1998) [6]	ADD
Paillier (1999) [7]	ADD
DJ (2001) [9]	ADD
Galbraith (2002) [10]	ADD
KTX (2007) [11]	ADD

Table 1: Homomorphic satisfiable well-notorious properties HE schemes

3 Background, preliminaries and definitions

Some significant technical preliminaries and definitions are as follows:

3.1 Homomorphic encryption

Homomorphic Encryption is today's most extensively used cryptographic method. Homomorphic centric Encryption technology is a variety of encryption method that facilitates calculations to be done on certain ciphertext, resulting in an encrypted output that certainly matches along with the result of operations accomplished on the certain plaintext when decoded. Homomorphic Encryption is categorized into two general types - *Partially homomorphic encryption* as well as *Fully (totally) homomorphic encryption*. Fully Homomorphic centric encryption has the dominant position as both the Multiplicative along with the Additive homomorphic constraints are being satisfies here. First FHE oriented cryptosystem [1] proposed by C. Gentry in the year 2010 was a great breakthrough. Homomorphic computation scenario on Data, in general is depicted as Fig-2. This figure illustrates a cryptographic scheme involving operations on encrypted data. The process begins with two distinct data values, a and b. Each of these values undergoes an encryption process, resulting in their respective encrypted forms, denoted as $\text{Enc}_k(a)$ and $\text{Enc}_k(b)$. Interestingly, while still in their encrypted states, these values can be subjected to a specific operation, represented by the symbol o. This operation yields an encrypted result, $\text{Enc}_k(a\, o\, b)$, which encapsulates the outcome of performing

the operation on the original unencrypted values. Upon decryption, the process showcases its true strength. When the encrypted versions of a and b are decrypted using the key k, the original values a and b are faithfully retrieved. More impressively, decrypting the outcome of the operation on the encrypted values—specifically, $\text{Dec}_k(\text{Enc}_k(a\,o\,b))$—yields the same result as if the operation o had been applied directly to the original, unencrypted values, producing $a\,o\,b$. The described scheme is a representation of homomorphic encryption. This type of encryption allows for computations to be performed on ciphertexts, and when the results are decrypted, they match the expected outcome of the operations as if they were conducted on the plaintext.

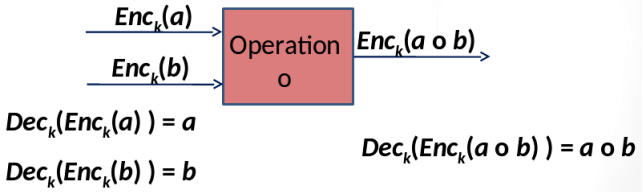

Figure 2: Homomorphic centric computation scenario on data

3.1.1 Ring homomorphism

Let, α as well as β are the rings. A particular fun_def $f : \alpha \to \beta$ necessarily be satisfying ring oriented homomorphism,

$$if \forall z_1, z_2 \in \alpha. \tag{1}$$

$$f(z_1 + z_2) = f(z_1) + f(z_2) \tag{2}$$

$$f(z_1 * z_2) = f(z_1) * f(z_2) \tag{3}$$

$$f(1_\alpha) = 1_\beta \tag{4}$$

The given equations i.e, 1, 2, 3, 4 present specific properties of a function f:. Collectively, these properties are typically associated with homomorphisms in algebraic structures, such as groups or rings. A function f upholding these properties preserves the structure between two algebraic systems, mapping one to the other while retaining the inherent operations like addition and multiplication.

3.1.2 RSA cryptosystem: multiplicative oriented homomorphic encryption

RSA cryptosystem satisfies multiplicative homomorphic property, thus it is a use-case of partial oriented homomorphic encryption conduction mechanism. Suppose, CT_1 and CT_2 are two certain ciphertexts. $\mathcal{P}_1$ and $\mathcal{P}_2$ are the plaintexts.
$CT_1 = \mathcal{P}_1^e \bmod n$
$CT_2 = \mathcal{P}_2^e \bmod n$
where, variable e: is the public key specific exponent; variable $n = p.q$: is calculated product for the two massive sized prime natured numbers p along with q.

$CT_1 \,.\, CT_2 = \mathcal{P}_1^e \,.\, \mathcal{P}_2^e \bmod n$
So, the multiplicative oriented HE property is: $(\mathcal{P}_1.\mathcal{P}_2)^e \bmod n$.
So, if in case, the usual encryption of an specific message $\mathcal{P}$ is provided by - $E(\mathcal{P}) = \mathcal{P}^e \bmod n$
Homomorphic centric computational effect is then - $E(\mathcal{P}_1) \,.\, E(\mathcal{P}_2) = \mathcal{P}_1^e \,.\, \mathcal{P}_2^e \bmod n = (\mathcal{P}_1.\mathcal{P}_2)^e \bmod n = E(\mathcal{P}_1.\mathcal{P}_2)$

3.1.3 Goldwasser-Micali cryptosystem: additive homomorphic encryption

The GM oriented cryptosystem particularly is dependent onto the difficulty of the quadratic complex residuosity problem [Kaliski 2005]. If there is an specific signed integer x correspondingly $x^2 \equiv a(mod\ n)$, the count a is depicted quadratic partial residue modulo n. The quadratic oriented residuosity issue is used to determine whether or not a particular integer q is considered as quadratic modulo with n. The following is particularly an interpretation of the GM cryptosystem:

$c_i = E(m_i) = y_i^2 x^{m_i} (mod\ n)$
$\forall m_i = 0, 1$

where, variable $m = m_0 m_1 ... m_r$, variable $c = c_0 c_1 ... c_r$ as well as r is considered as the certain block size exploited for the certain message space as well as x is chosen from computational space i.e., Zn^* at random manner for each and every chunk of encryption, where the Zn^* is considered as the specific multiplicative oriented subgroup of the integers with modulo within n that comprehends all the specific numbers which are smaller as compare to r as well as co-prime to r.

The GM cryptosystem's homo-structured characteristic demonstrates that the encryption about the summation $E(m_1 \oplus m_2)$ can correctly be determined precisely from individually encrypted form of bits $E(m_1)$ along with $E(m_2)$. The procedure is the same with exclusive-OR since the plain form message and encoded format are members of the certain set (0, 1), hence, GM is definitely homomorphic upon only the addition for associated binary numbers. GM cryptosystem is not multiplicative homomorphic.

3.2 Computational verifiability

Lemma 1: *If integer factorization on a large scale is impossible, factorising the N in certain polynomial time is impossible.*
Proof: Consider that x is a competitor who can factorise a provided number N into definite primes i.e., p as well as q with the same structured bit length space in a polynomial specific time. Assume that the associated probability of this operation is p'. A number N might have minimum two prime order factors for each element $fact_i$. As a result, the attacker's chance of factorising it is p_{r^n}, which

is virtually as low as p'. Therefore, the resultant standard probability by which an intruder/ attacker can certainly do factorization N is $\prod_{i=1}^{m} p''_r \leq (p')^m$. Now, if the p' is approximately negligible, then the overall estimated probability will also approximately imperceptible.

Definition 1: An specific matrix $\mathcal{M} \in R^{n,n}$ that can be thought of as an orthogonal if this is properly satiating one out of the computationally similar constraints - (i) $\mathcal{M}.\mathcal{M}^T = \mathcal{M}^T.\mathcal{M} = I_n$ (ii) Variable $\mathcal{M}$ is certainly invertible as well as $\mathcal{M}^{-1} = \mathcal{M}^T$.

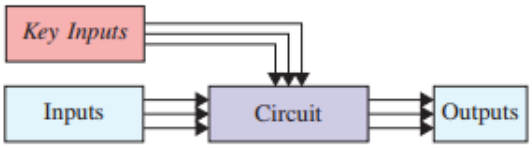

Figure 3: Circuit computation

3.2.1 Compute circuitry

The correctness of N an specific input over the computational domain $(0,1)^w$ circuitry can usually evaluated utilizing multi-key as well as levelled FHE in conjunction with the NAND gate computational logic. The exploitation of NAND gate in computational circuitry is as follows: NAND and NOR logic gates are also known universal gates since they can be integrated to produce any of the other logic gates e.g., OR, AND, and NOT gates. However, in Complementary metal-oxide–semiconductor (CMOS) circuit, NAND gate is both smaller, area-wise, and faster than a NOR gate. Therefore, adaptation of NAND is overall computationally and economically cheaper. One computation oriented synopsis is depicted as Fig-3. Here, its clearly observable that specific NAND circuitry based logic being functionally feasible and complete, computation can be carried out to any circuit depth. The client then can check the result by using decryption multiplicative as well as additive circuits after receiving the result. As a result, the plaintext retrieved is confirmed. Few example scenarios are provided in section 6.2.1 and 6.2.2.

3.3 Applications

The mostly observed natural examples of privacy homomorphism are existing in the scenarios of two-party setting. Making encrypted requests to search engines is a basic example. Party $\mathcal{A}$ creates a publicly defined key pk for the completely homomorphic oriented encryption technological scheme along with cipherdata i.e. $\psi_1, \cdots, \psi_t$ that usually encode the query $\pi_1, \cdots, \psi_t$ under the pk to execute an encrypted search.
Searching over encrypted data is another natural use. In this case, $\mathcal{A}$ saves his data on a server (for example, the Internet) so that he may access them without having to use his own entity computer. However, he encrypts his files due to the reason that the malevolent server may otherwise see or disclose his personal information. Let $\pi_1, \cdots, \pi_t$ be the bits that depict the files that are usually encrypted to generate ciphertexts i.e. $\psi_1, \cdots, \psi_t$. Assume that at the some point in the future, $\mathcal{A}$ wishes to retrieve all of his encoded files which fulfil a query - for example, any files possessing the term $\mathbb{X}$ within the 5 words of '$\mathbb{Y}$' but not specifically the word '$\mathbb{Z}$'. He submits his inquiry to the trusted server, that converts it into a C circuit. Then the server delivers these ciphertexts to $\mathcal{A}$ by setting ψ_i^* $\leftarrow$ Evaluate_fun($pk, C_i, \psi_1, \cdots, \psi_t$). $C(pi_1, \cdots, pi_t)$, the bits of the files that meet his query, are decrypted using $\mathcal{A}$. (A must indicate an upper limit on the amount of certain bits the answer should contain in this specific application, as in the variety of encoded search pertinence, and the particular server's encoded feedback will be specifically truncated to that loftier bound.)

3.4 Definitions on computational aspects

Definition 1. (Correctness of the Homomorphic-oriented Encryption): We generally consider that a particular homomorphic centric encryption protocol ε is absolutely appropriate for specific circuits i.e., in the C_ε if, for the any variety of key-pair (sk, pk) specific outcome by KeyGen $\varepsilon(\lambda)$, any variety of circuit $C \in C_\varepsilon$, any variety of plaintexts i.e. $\pi_1, \cdots, \pi_t$, as well as any variety of ciphertexts i.e., $\psi = \psi_1, \cdots, \psi_t$ with the $\psi \leftarrow Encr_\varepsilon(pk, \pi_i)$, it is most genuine case that -

$$\text{if } \psi \leftarrow Evaluate\_fun_\varepsilon(pk, C, \psi) \text{ , therefore}$$
$$Decrypt\_fun_\varepsilon(sk, \psi) \rightarrow C(\pi_1, \cdots, \pi_t)$$

except with the negligible amount of probability instead arbitrary natured coins in the $Evaluate\_fun_\varepsilon$ function.
By the itself, mere complete correctness generally aborts to suspend the trivial protocols. In more confined way, consider we now define function Evaluate the $\varepsilon(pk, C, \psi)$ to specifically just confine output as (C, ψ) without "computing" the computational circuit or the cipherdata at all, and then carry out $Decrypt\_fun_\varepsilon$ operation to decode the modular component cipherdata and then apply C to the results. This protocol is absolutely correct, but not much interesting. We can certainly address this variety of shortcoming specifically by the higher-limiting the absolute length of cipherdata outcome by the function of: Evaluate ε. One certain manner to carry-out this is by adopting an out bound upon the particular magnitude of the decoded circuit D_ε particularly for the method ε that reckon on mainly only onto the trust factor and security specific parameter, as depicted in the below definition.

Definition 2. (Compact oriented Homomorphic-constrained Encoding): Any such homomorphic encoding system ε is considered as compact if there exist any polynomial depicted f so that the decryption centric algorithm of ε can then be described as a computational circuit D_ε of arbitrary size at most $f(\lambda)$ for each and every argument of the security-constraint parameter λ.

Definition 3. (Compactly Computes): If ε is compact as well as also accurate for special kind of circuits in C_ε, we claim that a particular homomorphic oriented encryption methodology ε "compactly then evaluates" the circuits in C_ε. Because homomorphic encoding techniques in which the specific ciphertext size substantially grows in sub-linearly fashion along-with the total size of the computational circuit are certainly still attractive for abundant of applications, we can investigate alternative relaxations of compactness. For example, we may allow the secret natured key and ciphertext sizes to expand polynomially associated with the circuit depth. We'll refer to such systems as "quasi-compact" informally.

Definition 4. (Fully Homomorphic Encryption): If a homomorphic encryption system ε compactly assesses all circuits, it is said to be completely homomorphic. This definition may appear overly broad, because, as previously stated, quasi-compactness might suffice; nevertheless, we omit exploiting quasi-compactness in our core definition as it is difficult to codify and we scarcely exploit the concept anyway. Another prime reason it's too strict is that it eliminates levelling methodologies, that only assess circuits with a depth of d and a public oriented key dependant length of $poly(d)$; consequently, the underneath bound relaxation.

Definition 5. (Leveled-constrained Fully Homomorphic Encryption): We consider that a set of homomorphic encryption-oriented protocols $\varepsilon^{(d)} : d \in Z+$ is generally leveled-oriented fully homomorphic in case, for every $d \in Z+$, they atmost all exploit the similar decryption circuit, $\varepsilon(d)$ that compactly computes all the homomorphic circuits of the depth mostly d (that utilize some certain set of computational gates), and then the amortized computational-oriented complexity depicted as $\varepsilon(d)'s$ procedures is of polynomial nature in λ, d, as well as (in the particular case of Evaluate function ε) the overall structure of the computational circuit C.

Definition 6. (Circuit-constrained Private Homomorphic Encryption): We consider that a particular homomorphic-oriented encryption protocol ε is more specifically circuit-partial private for the circuits in the C_ε if, for such certain key-pair i.e., (sk, pk) outcome by the KeyGen $\varepsilon(\lambda)$, any computational circuit $C \in C_\varepsilon$, as well as any kind of fixed natured ciphertexts, represented as - $\psi = \psi_1, \cdots, \psi_t$ that certainly are existing in the firmware image of function Encrypt ε for the plaindata $\pi_1, \cdots, \pi_t$, the under-mentioned distributions (over the certain arbitrary coins in the Encrypt_fun ε, Evaluate_fun ε) are (particularly statistically) indiscernible:

$$Encrypt\varepsilon(pk, C(\pi_1, \cdots, \pi_t)) \approx Evaluate\varepsilon(pk, C, \psi)$$

The certain obvious exactitude constraints must still stated good.

Definition 7. (Leveled-constrained Circuit Private Homomorphic Encryption): Similarly to circuit private-constrained homomorphic encryption, except that each level can have a different variety of distribution, and the certain distributions only have to be equal if they are specifically associated with the same unique level (inside the circuit). Unlike the circuit oriented privacy, levelled circuit privacy has not guarantee compactness in and of itself. That is, the ciphertext magnitude can expand exponentially along with the number of threshold levels in a levelled circuit private homomorphic encryption technique.

4 Proposed procedure

The adopted methodological framework in addition to the detailed algorithmic natured procedure steps are provided in this section. The various components in this framework are described as follows: The first procedure (Key generation module) is stepwise described in Algorithm 1. Encryption of client's private and confidential data is carried out in Algorithm 2. Computation on encrypted data on homomorphic circuitry at cloud end is performed in Algorithm 3. The decryption using private key and verification of inherent homomorphic property is discussed in Algorithm 4.

Algorithm 1: Key Generation Module

1. Randomly choose two massive sized and equal length prime numbers α and β such that their gcd($\alpha.\beta$, $(\alpha-1).(\beta-1)$) = 1
2. Compute $n \leftarrow \alpha.\beta$
3. Compute $\lambda \leftarrow$ least common multiple $(\alpha-1, \beta-1)$
4. Choose a random integer θ where, $\theta \in Z^*_{n^2}$
5. Make sure that specific n divides mainly the associated order of θ by probing the core existence of below modular oriented multiplicative-centric inverse,
$\delta = (\mathbb{L}(\theta^\lambda \ mod \ n^2))^{-1} \ mod \ n$
where, $\mathbb{L}$ natured function is generally elucidated as,
$\mathbb{L}(i) = \frac{i-1}{n}$
6. The public as well as private key pairs for particular encryption as well as decryption are now: (n, θ) & (λ, δ) respectively.

return()

5 Empirical analysis

This section discusses about the empirical analysis of the developed method in terms of computational complexity, security and correctness analysis.

Algorithm 2: Encryption of Client's Private Data

1. Consider $\mathcal{M}$ as plain message from client where, $0 \leq \mathcal{M} < n$
2. Select the random integer x where, $0 \leq x < n$
3. Compute particular ciphertext $\mathcal{C} \leftarrow \theta^{\mathcal{M}}.x^n \; mod \; n^2$

return()

Algorithm 3: Computation on Encrypted Data on Homomorphic Circuitry at Cloud End

1. Perform homomorphic computation at cloud side $\prec\circlearrowleft (\boxplus, \boxdot) \succ$ on encrypted client's private data upto optimal circuit depth.
2. The calculated result is returned to the client.

return()

Algorithm 4: Secure Decryption of Computed Result by Client

1. Consider $\mathcal{C}$ is ciphertext result (encrypted format) which needs to be decrypted where, $\mathcal{C} \in Z^*_{n^2}$
2. Compute $\mathcal{M} \leftarrow \mathbb{L}(\mathcal{C}^{\lambda} \; mod \; n^2).\delta \; mod \; n$

return()

Definition 5.1. The security of the proposed procedure primarily relies on the computational intractability of the Residue Class Problem, which involves distinguishing between n^{th} power residues and non-n^{th} power residues modulo n^2. More formally, consider the set of n^{th} residues (ζ_n), which forms a multiplicative subgroup of $Z^{\star}_{n^2}$, the group of units modulo n^2. It can be shown that the size of the group $Z^{\star}_{n^2}$ is approximately $n.\chi(n)$, where $\chi(n)$ is the Euler's totient function. The cardinality of the set of n^{th} residues, denoted by $|\zeta_n|$, equals the value of the Euler's totient function, $\chi(n)$. Consequently, the set ζ_n is substantially smaller than $Z^{\star}_{n^2}$, making the task of distinguishing an arbitrary element of $Z^{\star}_{n^2}$ as an n^{th} residue or a non-n^{th} residue non-trivial and computationally hard.

Theorem 5.1. Each r^{th} residue $\vartheta \in Z^{\star}_{r2}$ possesses at most r distinct r^{th} roots in $Z^{\star}_{r2}$.

Lemma 5.1. Consider, $\Im$ be an element in $Z^{\star}_{n^2}$. Now, if the cyclic order of ($\Im$ mod n^2) is a non-zero multiple of n, then $Enc_{\Im}$ is thought of to be bijective.

Lemma 5.2. With ref. to section 4.3, function $\mathcal{C} \mapsto \; \sqsubset c \sqsupset_{\Im}$ is circuit homomorphism from $(Z^{\star}_{n^2}, \boxdot)$ to $(Z_n, \boxplus)$ for any $\Im \in \prec\circlearrowleft (\boxplus, \boxdot) \succ$, i.e., $\forall \; \mathcal{C}_i \in Z^{\star}_{n^2}$, the following condition holds:-
$\sqsubset c_1.c_2 \sqsupset_{\Im} \; = \; \sqsubset c_1 \sqsupset_{\Im} + \sqsubset c_2 \sqsupset_{\Im} \; mod \; n$. It proves that the method is significantly easy to compute at client end but very hard to invert at cloud end.

Utilizing privacy homomorphism here in this framework for query vector outsourcing offers several notable computational strengths and novelties:

- Data Confidentiality: Privacy homomorphism ensures that the data remains encrypted throughout the computation process. This means that even during outsourcing, the actual data values remain hidden, ensuring data confidentiality.
- Computation on Encrypted Data: One of the significant advantages of privacy homomorphism is the ability to perform computations directly on encrypted data without the need for decryption. This feature is particularly valuable for query vector outsourcing, as computations can be conducted off-site or on a third-party server without compromising data privacy.
- Reduced Local Computation: By leveraging privacy homomorphism, businesses and organizations can outsource intensive computations to cloud servers or third-party platforms. This not only offloads the computational burden from local systems but also leverages the powerful computational capabilities of cloud platforms, all while maintaining data privacy.
- Preservation of Data Integrity: With privacy homomorphism, results obtained after processing encrypted data, once decrypted, are consistent with the results of operations performed on the raw, unencrypted data. This ensures that the integrity of the data and computations remains uncompromised.
- Enhanced Security for Third-Party Computations: Since the third-party servers or platforms only handle encrypted data and never see the actual values, the risks associated with data breaches or malicious attacks are significantly reduced.
- Scalability: Privacy homomorphism is conducive to scalable operations. As organizations grow and the volume of data for query vector outsourcing increases, the homomorphic encryption schemes can efficiently handle larger datasets without compromising on security.
- Versatility: Privacy homomorphism can support various operations, including addition and multiplication, making it versatile for a range of query types and computational needs in the outsourcing scenario.

6 Experiments and results discussion

This section presents experimental set-up details, procedure and compilation of obtained results in different test case scenarios.

6.1 Setup and simulation environment

Our experimental simulation driven set-up contains the system environment as Ubuntu (Version: 16.04 LTS) with specifically 64 bit OS, 8 GB of RAM, processor i.e., Intel Core i7-860 along with 2.80 GHz clock speed, count of cores as 4 and total 8 threads were available. Python (v'3.5.2) being installed and exploited for the implementation.

6.2 Procedure and obtained results

6.2.1 Experimental scenario-I

```
Step 1 --> Key Generation

alpha, beta = 971 911
n is = 884581
Lambda = 88270
Theta = 585146362844
c is 30679922824
L is 34683.0
Delta is 166699
Public Key is (n, Theta): 884581,
585146362844
Private Key is (Lambda, Delta): 88270,
166699
-----------------------------------
Step 2 --> Encryption Using Public
Key(Client/ User side)

Plain Message M1 is: 316

Ciphertext_1 is: 244518097031

Plain Message M2 is: 982

Ciphertext_2 is: 638403686475
-----------------------------------
Step 3 --> Homomorphic Computation On
Encrypted Data(Cloud side)

Add Circuit_Result = 882921783506

Mult Circuit_Result = 15610125
4554442152355725
-----------------------------------
Step 4 --> Decryption Using Private
Key(Client/ User side)
Plaintext_1 (after decryption) is:
316.0
Plaintext_2 (after decryption) is:
982.0
-----------------------------------
Step 5 --> Verfication of Homomorphic
Property
Homo_Prop: 1298.0
```

6.2.2 Experimental scenario-II

```
Step 1 --> Key Generation

alpha, beta = 499 829
n is = 413671
Lambda = 68724
Theta = 165047574144
c is 65394766365
L is 158084.0
Delta is 16593
Public Key is (n, Theta): 413671,
165047574144
Private natured Key is (Lambda,
Delta): 6872
4, 16593
-----------------------------------
Step 2 --> Encryption Using Public
Key(Client/ User side)

Plain Message M1 is: 2964

Ciphertext_1 is: 167960038665

Plain Message M2 is: 99231

Ciphertext_2 is: 31997839481
-----------------------------------
Step 3 --> Homomorphic Computation
On Encrypted Data(Cloud side)

Add Circuit_Result = 199957878146

Mult Circuit_Result = 53743583564252
23532865
-----------------------------------
Step 4 --> Decryption Using Private
Key(Client/ User side)
Plaintext_1 (after decryption) is:
2964.0
Plaintext_2 (after decryption) is:
99231.0
-----------------------------------
Step 5 --> Verfication of Homomorphic
Property
Homo_Prop: 102195.0
```

Therefore, the total of the plaintexts decrypted by the product of two or more ciphertexts.

Client's data size	Key Size	Exec. Time
10 bits	$\{40, 18\}$ bits	2.3 sec
17 bits	$\{45, 19\}$ bits	12.1 sec
64 bits	$\{64, 29\}$ bits	3120.6 sec

Table 2: Comparison results on various scenarios

Table 2 presents the compilation of results on various test case scenarios. Experimental evaluations are performed using proposed procedure on client's private data as well as key pairs (public, private) of varied sized length (in bits). For each simulation, the protocol execution time (in seconds) is also mentioned. Table 3, titled "Benchmarking",

Scheme	PK Size	Exec. Time
Liangmin W. et al. [27]	$< 32 >$ bits	1604.3 sec
Vinod R. F. et al. [28]	$< 64 >$ bits	3399.1 sec
Proposed Framework	$< 64 >$ bits	3120.6 sec

Table 3: Benchmarking

offers a comparative overview of various cryptographic schemes, emphasizing their private key sizes and execution times. Among the three schemes highlighted, the work by Liangmin W. et al., employs a private key size of fewer than 32 bits and executes in 1604.3 seconds. Conversely, the scheme proposed by Vinod R. F. et al., uses a private key size of fewer than 64 bits, but its execution time is notably higher at 3399.1 seconds. Remarkably, the "Proposed Framework", while maintaining a private key size of fewer than 64 bits akin to Vinod R. F. et al.'s scheme, achieves a faster execution time of 3120.6 seconds. This suggests that the proposed framework exhibits computational novelty by enhancing efficiency without compromising the robustness of the private key size, presenting a promising advancement in the field of cryptographic methods.

The decision to use privacy homomorphism for query vector outsourcing stems from its inherent ability to facilitate computations on encrypted data without needing decryption. This property ensures that sensitive data remains confidential even during computational processes, making it an optimal choice for outsourcing tasks to environments where data privacy is paramount. Compared to other methods, privacy homomorphism provides a distinct advantage in terms of security. Traditional outsourcing methods might require data to be decrypted on third-party servers, exposing it to potential breaches. Privacy homomorphism, on the other hand, mitigates this risk by maintaining data encryption throughout the computation process. In terms of efficiency, while initial homomorphic encryption schemes were computationally intensive, advancements in the field have led to more optimized solutions that can cater to real-world applications. Thus, the balance of robust security and enhanced efficiency makes privacy homomorphism a compelling choice for query vector outsourcing over conventional methods.

7 Conclusive summary

In today's digital age, data privacy is of paramount importance. It's crucial to safeguard user accounts and assets from potentially malicious cloud service providers. Traditional solutions often involve encrypting data modules and entrusting the keys entirely to the service provider. However, this approach sacrifices control over the confidentiality of sensitive data. This paper introduces a prototype that addresses these issues using privacy homomorphism. For most homomorphic encryption methods, the multiplicative depth of circuits poses a significant constraint when executing computations on encrypted data. Despite annual advancements in this domain, the technology grapples with intricacies of more complex datasets and demands substantial computational power for practical, real-time operations. Exploring opportunities for the practical implementation of this framework will serve as the future direction for our research.

Authors' contribution:

All authors have equally contributed in this research work. The credit author's statement is as follows:

V. Swathi: Conceptualization, methodology, implementation, writing original draft, writing-review and editing.

M.P. Vani: Formal analysis, supervision, visualization, investigation.

References

[1] R. Rivest, L. Adleman, and M. Dertouzos. On data banks and privacy homomorphisms. In Foundations of Secure Computation, pp. 169-180, 1978. https://doi.org/10.14264/219648

[2] S. Goldwasser, S. Micali, Probabilistic encryption and how to play mental poker keeping secret all partial information. Proc. 14th Symposium on Theory of Computing, pp. 365-377, (1982). https://doi.org/10.1145/800070.802212

[3] ElGamal T, A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Trans Inf Theory (31), pp. 469-472, 1985. https://doi.org/10.1007/3-540-39568-7_2

[4] Benaloh, J. Dense Probabilistic Encryption. In: Proceedings of the Workshop on Selected Areas of Cryptography, pp. 120-128 (1994). https://doi.org/10.1007/978-3-642-21969-6_22

[5] Naccache, David; Stern, Jacques, A New Public Key Cryptosystem Based on Higher Residues. Proceedings of the 5th ACM Conference on Computer and Communications Security. CCS'98 ACM, pp. 59-66, (1998). https://doi.org/10.1145/288090.288106

[6] Okamoto, Tatsuaki; Uchiyama, Shigenori, A new public-key cryptosystem as secure as factoring, Advances in Cryptology - EUROCRYPT'98. Lecture Notes in Computer Science, 1403. Springer, pp. 308-318, (1998). https://doi.org/10.1007/bfb0054135

[7] Paillier, Pascal, Public-Key Cryptosystems Based on Composite Degree Residuosity Classes, EUROCRYPT'99. Springer, pp. 223-238, (1999). https://doi.org/10.1007/3-540-48910-x_16

[8] Craig G. Fully homomorphic encryption using ideal lattices, STOC. Vol. 9. 2009. https://doi.org/10.1145/1536414.1536440

[9] Ivan Damgard, Mads Jurik, A Generalisation, a Simplification and Some Applications of Paillier's Probabilistic Public-Key System. Public Key Cryptography, pp. 119-136, (2001). https://doi.org/10.7146/brics.v7i45.20212

[10] Galbraith, Jay R., Organizing to Deliver Solutions, Organizational Dynamics, 31 (Autumn), pp. 194-207, (2002). https://doi.org/10.1016/s0090-2616(02)00101-8

[11] Akinori Kawachi, Keisuke Tanaka, and Keita Xagawa, Multi-bit cryptosystems based on lattice problems, In Public Key Cryptography-PKC 2007. Springer, pp. 315-329. https://doi.org/10.1007/978-3-540-71677-8_21

[12] Peeter Laud, Alisa Pankova, On the (Im)possibility of Privately Outsourcing Linear Programming, CCSW'13, November 8, 2013, Berlin, Germany. https://doi.org/10.1145/2517488.2517490

[13] Lifei Wei, Haojin Zhu, Zhenfu Cao, Xiaolei Dong, Weiwei Jia, Yunlu Chen, Athanasios V. Vasilakos, Security and privacy for storage and computation in cloud computing, Information Sciences 258 (2014), pp. 371-386. https://doi.org/10.21275/v4i12.nov151792

[14] Xiaofeng Chen, Jin Li, Jianfeng Ma, Qiang Tang, Wenjing Lou, New Algorithms for Secure Outsourcing of Modular Exponentiations, IEEE Transactions on Parallel and Distributed Systems, Vol. 25, No. 9, September 2014. https://doi.org/10.1109/tpds.2013.180

[15] Frederik Armknecht, Colin Boyd, Christopher Carr, Kristian Gjosteen, Angela Jaschke, Christian A. Reuter, Martin Strand, A Guide to Fully Homomorphic Encryption, (2015). https://doi.org/10.1007/978-3-642-31410-0_15

[16] Jannatul Ferdush, Tasnim Mehzabin, M. M. A. Hashem, Securely Outsourcing of Large Scale Linear Fractional Programming Problem to Public Cloud, in Procs. of the IEEE 2016 5th International Conference on Informatics, Electronics and Vision (ICIEV 2016), Bangladesh. https://doi.org/10.1109/iciev.2016.7760028

[17] Jianfeng Wang, Xiaofeng Chen, Efficient and Secure Storage for Outsourced Data: A Survey, Data Sci. Eng. (2016) 1(3): pp. 178-188. https://doi.org/10.1007/s41019-016-0018-9

[18] Kristian Gjosteen, Martin Strand, Can there be efficient and natural FHE schemes. https://doi.org/10.1145/3538969.3544417

[19] Dario Fiore, Maria Isabel, Gonzalez Vasco, Claudio Soriente, Partitioned Group Password-Based Authenticated Key Exchange, (2017). https://doi.org/10.1093/comjnl/bxx078

[20] Aayush Jain, Peter M. R. Rasmussen, Amit Sahai, Threshold Fully Homomorphic Encryption, (2017). https://ia.cr/2017/257

[21] I-Chen Tsai, Chia-Mu Yu, Haruo Yokota, Sy-Yen Kuo, Key Management in Internet of Things via Kronecker Product, 2017 IEEE 22nd Pacific Rim International Symposium on Dependable Computing, IEEE, (2017). https://doi.org/10.1109/prdc.2017.25

[22] Swathi, V., and M. P. Vani. "Security and privacy challenges in cloud: survey and research direction." International Journal of Computation Enginering Research (IJCER) 7.08 (2017): 63-72. https://doi.org/10.23883/ijrter.2018.4083.xwpna

[23] Swathi, V., D. Saidulu, and B. Chandrakala. "Enabling Secure and Effective Spatial Query Processing on the Cloud using Forward Spatial Transformation." International Journal of Computer Engineering In Research Trends 4.7 (2017): 301-307. https://tinyurl.com/yc4aywpz

[24] Bo Zhang, Boxiang Dong, Wendy Hui Wang, Integrity Authentication for SQL Query Evaluation on Outsourced Databases: A Survey, August 2018. https://doi.org/10.1109/tkde.2019.2947061

[25] Yanguo Peng, Hui Li, Jiangtao Cui, Jianfeng MA, Yingfan Liu, Towards Secure Approximate k-Nearest Neighbor Query Over Encrypted High Dimensional Data, IEEE Access, Vol. 6, pp. 23137-23151, 2018. https://doi.org/10.1109/access.2018.2830355

[26] Ahmed El-Yahyaoui, Mohamed Dafir Ech-Cherif El Kettani, An Efficient Fully Homomorphic Encryption Scheme, International Journal of Network Security, Vol.21, No.1, pp. 91-99, Jan. 2019. https://tinyurl.com/4vbbmkky

[27] Liangmin Wang, Zhendong Yang, Xiangmei Song, SHAMC: A Secure and highly available database system in multi-cloud environment, Future Generation Computer Systems, Volume 105, April 2020, pp. 873-883. https://doi.org/10.1016/j.future.2017.07.011

[28] Vinod Ramesh Falmari, M. Brindha, Privacy preserving cloud based secure digital locker using Paillier based difference function and chaos based cryptosystem, Journal of Information Security and Applications, Volume 53, August 2020, p. 102513. https://doi.org/10.1016/j.jisa.2020.102513

[29] Swathi, V., and M. P. Vani. "Privacy-Cheating Discouragement: A New Homomorphic Encryption Scheme for Cloud Data Security." 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT). IEEE, 2020. https://doi.org/10.1109/icccnt49239.2020.9225481

[30] Swathi, V., and M. P. Vani. "A secure increased key policy attribute in cloud computing." Journal of Innovation in Computer Science and Engineering 11.1 (2021): 26-29. https://tinyurl.com/49tdww78