Securing Data in Clouds using the SDC Algorithm: Current Trends and Research Directions

Dennis Redeemer Korda
Department of ICT
Bolgatanga Technical University
Bolgatanga, UER, Ghana

Eric Ayintareba Akolgo
Department of Computer Science
Regentropfen College of Applied Sciences
Bolgatanga, UER, Ghana

Emmanuel Oteng Dapaah
Department of ICT
E.P College of Education
Bimbila, NR, Ghana

Dickson Kodzo Mawuli Hodowu
Ghana Water Company Limited
Department of Technology & Innovation
Bolgatanga, UER, Ghana

ABSTRACT
The rapid evolution of cloud computing has brought forth an urgent need to ensure the security and confidentiality of data stored in the cloud. Access control methods, while useful, may not provide sufficient protection. As a viable alternative, data encryption has emerged as a robust solution, empowering organizations to encode data before transferring it to the cloud. Within the realm of encryption algorithms, homomorphic encryption stands out for its unique capabilities. By allowing computations on ciphertext data without revealing the original information, it addresses both confidentiality and data processing needs in cloud computing. This review delves into the critical topic of cloud computing security with a focus on safeguarding data privacy using homomorphic encryption. Various encryption algorithms are explored, including the renowned Gentry cryptosystem, DGHV algorithm, Gen10, SDC algorithm, and the cutting-edge GSW algorithm. Homomorphic encryption, which permits computations on ciphertexts, is dissected, distinguishing between additive and multiplicative homomorphisms. Notably, fully homomorphic encryption, a groundbreaking concept, supports both addition and multiplication operations on ciphertexts.

The inception of fully homomorphic encryption by Gentry marked a pivotal moment in the field, enabling computations on encrypted data. Subsequent research expanded upon this concept, introducing variations and practical implementations. One such implementation aims to bridge the gap between cloud computing and data confidentiality, offering a glimpse into the future of secure cloud computing. This paper sheds light on the key homomorphic encryption algorithms utilized in safeguarding cloud computing data, providing a comprehensive overview of their features and potential applications.

Keywords
Encryption, Algorithms, Data Security, Homomorphic Encryption Algorithms and Cloud Computing

1. INTRODUCTION
The past few years have seen escalating attention in the study and application of cloud computing and the various ways of safeguarding the data stored in the cloud. Some of the main concerns regarding how to guarantee the confidentiality and control which user group has access to the data. Access control methods can be used to manage which user group has the privilege to modify data on the cloud but these methods may not be sufficient for a secured cloud. An alternate solution that secures that from both the Cloud Service Provider (CSP) and unauthorized users is encryption where data is encoded by the organization before submitting it for storage in the cloud.

There are some encryption algorithms for various purposes. Suppose the encryption algorithm used for a secured cloud happens to be homomorphic, cloud users can execute significant computations on the encoded data, deprived of altering the initial data. Applying homomorphic encryption to encode data not only guarantees the safety of cloud figuring data but also reposes some form of assurance for cloud customers. Aside from the characteristic of homomorphic encryption allowing computations on ciphertext operations, the problem of the efficiency of ciphertext retrieval is also curtailed unlike in the traditional cryptosystems. Consequently, [1] proposed a down-to-earth, straightforward and completely homomorphic encryption algorithm, utilizing fundamentally rudimentary modular arithmetic, gotten after Gentry cryptography to guarantee security saving in the cloud, where scrambled data can be worked upon legitimately devoid of the influence of the confidentiality of the encryption frameworks so it can splendidly understand the necessity of cyphertext recovery and further data dispensation in cloud computing.

This review presents and discusses issues on Cloud computing security purposely on the most proficient method to secure the secrecy of information in the cloud by using homomorphic encryption. Some of the algorithms reviewed include the Gentry cryptosystem, DGHV algorithm, Gen10, SDC algorithm and the most recent GSW algorithm.

A homomorphic encryption algorithm is an encryption algorithm that legitimately permits the execution of ciphertext activities on ciphertexts, where the results being naturally scrambled permits, anyone, to control what is encoded, even deprived of knowing the mystery key [1]. Within the fully Homomorphic encryption, there is the addictive homomorphic encryption that permits only additions to the raw data, there is also the multiplicative homomorphic encryption that also permits only multiplication on the raw data, while the last type allows both additions and products on the raw data.

Various types of fully homomorphic encryption algorithms have been suggested since Rivest et al. (1978), hinted at the possibility of fully homomorphic encryption algorithms. Some

additive homomorphic encryption algorithms that have evidence of semantic security are Goldwasser-Micali, Benaloh, Naccache-Stern, Okamoto-Uchiyama, Paillier, and Damgard-Jurik [1]. A semantically secure cryptosystem is one in which just little data about the plaintext can be practicably removed from the ciphertext and this is grounded on computational complexity [2]. RSA and ElGamal are multiplicatively homomorphic with an unbounded number of modular multiplications. According to [3], if: for Enc(d1) and Enc(d2) it is imaginable to compute Enc (f (d1, d2)), where f can be Addition, Multiplication or both then it is said to be fully homomorphic. They illustrated how these operations are carried out in Figure 1.

Fully Homomorphic Encryption (FHE) was originally demonstrated to be conceivable in the ongoing, leap-forward research of Gentry, which underpins both addition and multiplication on ciphertexts (Li et al., 2012). Afterwards, a few more fully homomorphic encryption algorithms were constructed but without working implementations. A subsequent FHE algorithm, which demonstrates that Gentry's model lattice-based algorithm can be substituted by a very simple somewhat homomorphic algorithm which utilizes integers instead of ideal lattices and therefore theoretically more straightforward, with comparable properties regarding homomorphic tasks and effectiveness was introduced in [4]. With the rapid penetration of cloud computing and its associated confidentiality concerns, [5], proposed another fully homomorphic encryption algorithm, intending to help make cloud computing compatible with confidentiality. A functioning implementation of the FHE alongside its performance measures was introduced in [1].

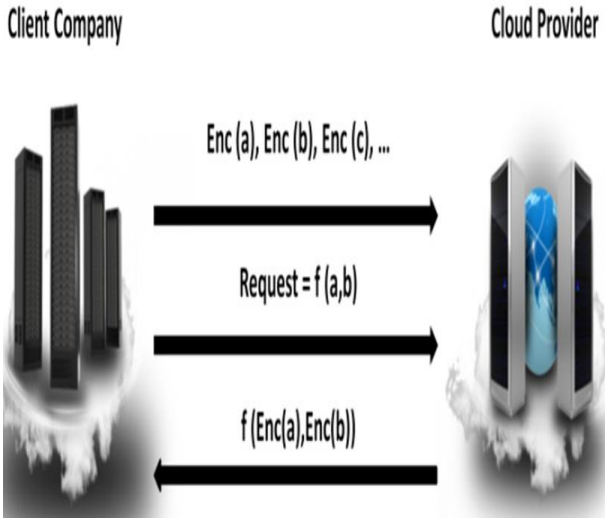

Figure 1: Application of FHE in Cloud Computing.

There are some homomorphic encryption algorithms that are utilized in securing data in cloud computing environments. Under this section, the Gentry cryptosystem, DGHV, Gen10, SDC encryption algorithms and the GSW algorithms are discussed.

2. GENTRY CRYPTOSYSTEM

Gentry's algorithm was based on lattice cryptography and it supported addition and multiplication procedures mutually on ciphertexts [6]. These addition and multiplication operations relate to AND (∧) and XOR (⊕) operations in Boolean algebra respectively. This is outstanding because it provided the basis for many functions to be derived from them [6]. For instance, ¬A can be derived from A ⊕ 1, and (¬A) ∧ (¬B) can also be derived from A ∨ B, and then transformed to (A ⊕ 1) ^ (B ⊕ 1). The common term for the construction of cryptographic primitives (encryption functions) that involve itself is known as lattice-based cryptography and any basis of Rn the subgroup of every single linear combination with integer coefficients of the basis vectors forms a lattice.

Gentry's lattice-based cryptography comprises numerous stages which start based on what was suggested as a somewhat homomorphic encryption (SWH) algorithm utilizing ideal lattices which are restricted to assessing low-degree polynomials over scrambled data. This restriction to some extent is a result of the noise in each ciphertext and as more computations (additions and or multiplications) are executed on the ciphertext, this noise grows until eventually, the noise brands the resultant ciphertext undecryptable. Afterwards, it jams the decryption process with the goal that it tends to be communicated as a small degree polynomial that is upheld by the algorithm. Then lastly, it uses a bootstrapping transformation, by means of an iterative self-implanting, to acquire a fully homomorphic algorithm [7].

Bootstrapping transformation technique efficiently "refreshes" the ciphertext by utilizing the decryption technique homomorphically, and in so doing acquires a new ciphertext with lower noise. Thus, the algorithm is said to be bootstrappable if it is capable of evaluating not just the decryption circuit which simply permits recryptions of the plaintext yet additionally improved variants of it [8]. It is along these lines conceivable to compute a discretionary number of additions and multiplications short of expanding an excess of noise by "refreshing" the ciphertext intermittently when the noise becomes excessively huge.

Unlike well-known algorithms like the RSA and Diffie-Hellman cryptographic algorithms which are effortlessly compromised by quantum PCs, some lattice-based constructions are resistant to these attacks. Figure 2 illustrates the difference between conventional algorithms and fully homomorphic encryption algorithms (FHE).

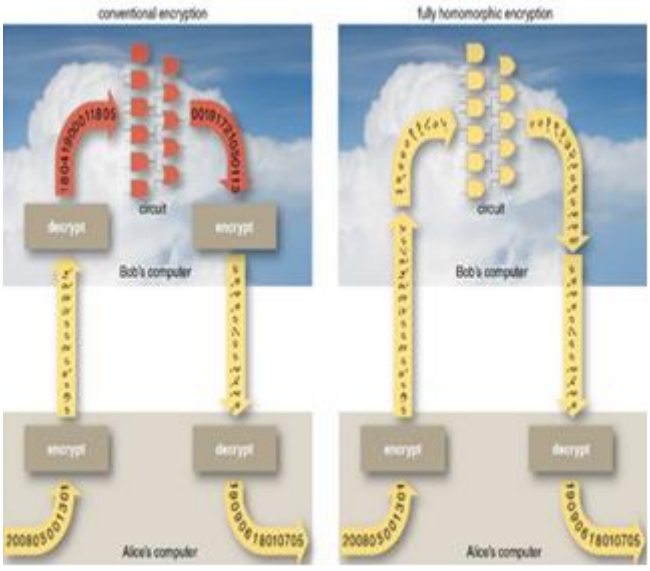

Figure 2: Conventional encryption and FHE

3. DGHV ALGORITHM

The researchers in [4], recommended a subsequent fully homomorphic encryption known as the DGHV algorithm, which improves upon the Gentry cryptosystem by showing that the Somewhat Homomorphic constituent of the ideal lattices

can be supplanted with an easier homomorphic algorithm which utilizes integers instead. This algorithm is, thus, theoretically uncomplicated as compared with the Gentry cryptosystem, in any case, has comparative qualities for homomorphic tasks and effectiveness.

A DGHV fully homomorphic public-key encryption algorithm consists of a number of sub-algorithms. These incorporate the typical KeyGen, Encrypt, Decrypt, and an extra significant algorithm known as Evaluate. KeyGen, as usual, is a large odd integer (for instance p) which is chosen at random and the complexity of the algorithm depends on how easy it is to factorize this odd integer. In order to Encrypt (p, m) a bit of a message, the ciphertext is established as an integer with residue mod p and has a similar equivalence as the plaintext. Viz., set

$$c = pq + 2r + m$$

where the integers q and r are selected indiscriminately in some other recommended intervals, with the end goal that 2r is lesser than p/2 in absolute value. The r represents the noise which is adequately lesser than the private key p and therefore the Decrypt (p, c) outputs (c mod p) mod 2. Evaluate the public key pk as input. This uncomplicated algorithm is additive and multiplicative homomorphic with respect to low mathematical computations and one can also utilize bootstrapping and squashing to morph this algorithm into FHE [1].

In the work of [9], he developed a framework for this algorithm that should be easy to use, not too dependent on the security measures taken by end-clients, have the option to handle any cryptographic operations within the trusted infrastructure, be able to send encoded data to the cloud and the public clouds where the encoded data is stored ought not to possess the capability to decode its contents. This proposed implementation structure is exemplified in Figure 3.

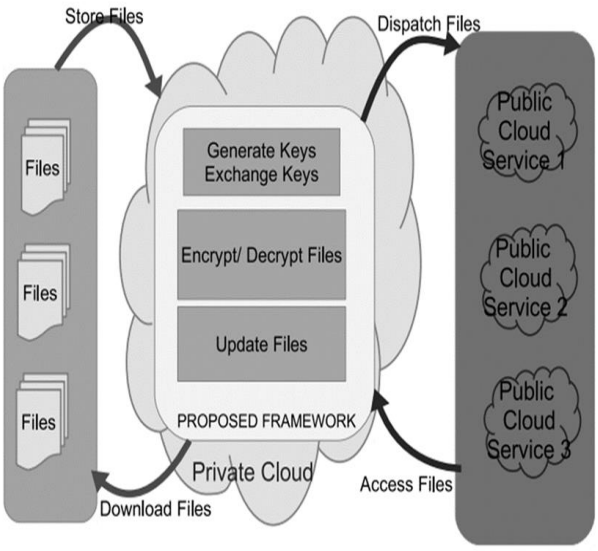

Figure 3: Framework Usage Scenario

According to the framework, the movement of the data continues as follows, data created by the clients (that includes staff personnel activity data, remote teammates, and people utilizing services supported by the customer company) are stored and transferred to the cloud environment of the company by means of a secure interface. On the off chance that the documents stay inside the reliable infrastructure of the company, they stay secure, however, when it is sent to other open cloud services, the issue of trust emerges and, in this way, the proposed system offers a helpful technique to accomplish communication with other open clouds. The framework of this structure can be utilized by companies to secure and deal with their data stored in unsecured open cloud environments.

As a major aspect of the work, he explored the chance of utilizing delta encryption concepts alongside homomorphic encryption algorithms by means of additive homomorphism to update encoded documents, rather than uploading the whole encoded versions each time after executing an update task. According to [9], in trial conditions, the created framework conveyed hopeful performance results when contrasted with other regular solutions. But the paper was silent on the ciphertext retrieval.

4. GEN10 ALGORITHM

In [5], the authors proposed an improved homomorphic encryption algorithm known as, the Gen10 algorithm in the publication of the Communications of the ACM, making a beeline for far-reaching utilization of cloud computing, which was amazingly basic and of the structure $c = pq + m$

The c represents the encrypted message (ciphertext), m represents the unencrypted message (plaintext), while p represents the key and q an arbitrary numeral [5]. This encryption procedure is, therefore, homomorphic in regard to addition, subtraction and multiplication. There exists a connection between c and m such that m is the remainder of c regarding modulus p, that is, $m = c \ mod \ p$.

For this algorithm, the encryption is such that; KeyGen is an arbitrary P-bit odd integer p and to Encrypt (p, m) a bit, let M represent an arbitrary N-bit number such that $M = m \ mod \ 2$.

So, the output of the ciphertext becomes c←M+pq. where q is an arbitrary Q-bit number. The Decrypt (p, c) Outputs c mod p, where (c mod p) is the integer C in (-p/2, p/2) such that p| (c – C).

The researchers in [10], focused on how to store data in cloud environments in an encoded format with FHE (Gen10) in Amazon Web Service (AWS) public cloud. The data was stored specifically stored in DynamoDB of AWS and computations were demonstrated in it. Their methodology involved a user first establishing a connection with the AWS DynamoDB service within the Eclipse IDE for Java EE Developers. Which at that point permits the client to log in depending on his/her credentials and afterwards the client can perform some computations on their data dependent on necessities [11]. Once the client is finished with all the jobs, he/she may decide to log out of the framework. The simplified flowchart is shown in Figure 4.

In the proposed algorithm, J and K denote a private key P0 and P1 denote a public key while N represents the number to be encoded and acknowledged as the client response. The purported algorithm is streamlined, productive, applied in AWS open cloud and can, accordingly, be utilized for different applications, for example, web-based auctioning, medical reasons and business reasons, but the ciphertext of this algorithm is too long for efficient processing.

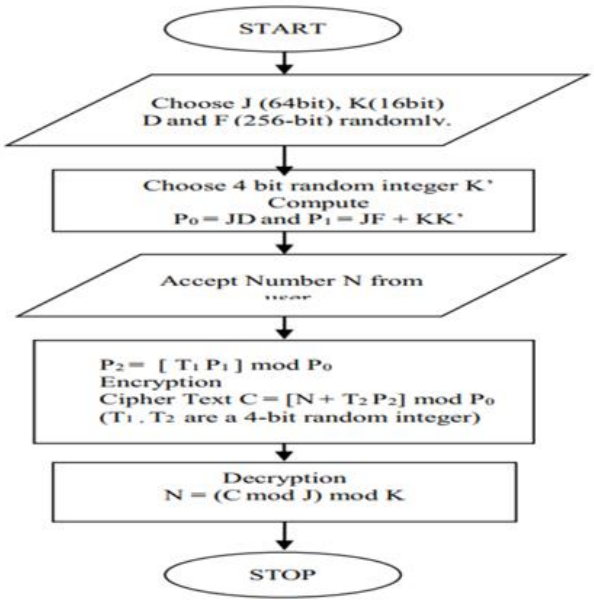

Figure 4: Flowchart for the Fully Homomorphic Encryption

5. SDC ALGORITHM

The Gentry cryptosystem, DGHV and Gen10 encourage the addition and multiplication of encrypted data, nevertheless, neither of the three has made references to the cyphertext recovery algorithms [11]. Therefore, [1] proposed a simple FHE known as the SDC algorithm, which is also based on the Gentry cryptographic encryption algorithm to safeguard data confidentiality in cloud environments. An illustration of this SDC algorithm is described below:

The KeyGen(p): Where p is an arbitrary P-bit odd integer and to Encrypt (p, m) a message or bit {0,1}

$$c = m + p + rpq$$

where r is an arbitrary R-bit numeral q is consistent with Q-bit enormous whole integer and c the ciphertext. The Decrypt (p, c) Output (c mod p) and the Retrieval(c):

$$R_i = (c_i - c_{index})\ mod\ q$$

As soon as the customer wishes to retrieve the message m_{index}, he encodes the keywords

$$c_{index} = m_{index} + p + rpq$$

and transports the c_{index} to the server. In receipt of c_{index}, the server inspects the ciphertexts, computing

$$R = c_i - c_{index} mod\ q$$

once R = 0, ciphertext retrieval works and Ci is the anticipated outcome [7]. A full description of this is shown in [11].

According to [4], Partially Homomorphic Encryption (PHE) such as RSA and Paillier algorithms are inadequate to protect cloud computing on the grounds that these algorithms permit performing just a single activity (being addition or multiplication) on the encoded data of the customer. But on a brighter side, utilizing the Fully Homomorphic Encryption (FHE) algorithm to encode data in the cloud server is the best guarantee for security and confidentiality issues, since this type of algorithm permits the execution of some essential computations on the encoded data. In their work, the various FHE algorithms were discussed and the utilization of the most effective one, the SDC algorithm, to protect cloud computing data was also discussed but not implemented and its performance evaluation was also not analyzed. The framework for FHE as proposed by [7] is shown in Figure 5.

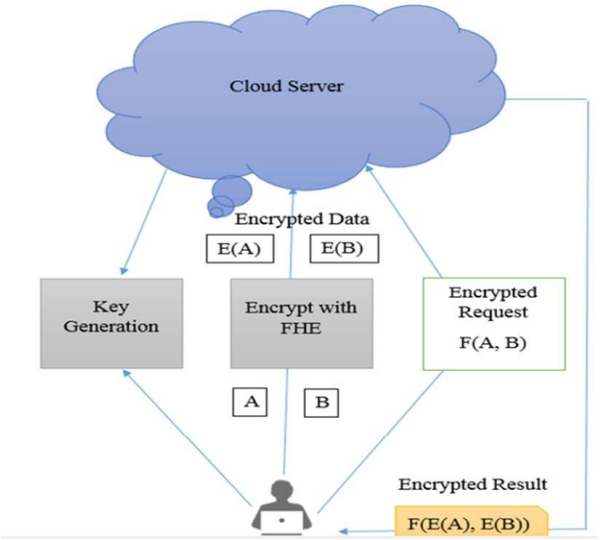

Figure 5: Utilizing FHE to protect cloud data.

Within their proposed framework: the user or client first logs in and generates the private or secret key using the key generation given by the server. Secondly, the client then encodes the data which will be sent to the cloud environment with the generated key and probably utilize a digital signature algorithm to guarantee the integrity as well as the non-repudiation of the data during transmission. Last but not least, computations in the server such as the search on the encrypted data can be done by sending an encrypted request to the server, and then finally, the client decrypts data using the private key to retrieve the actual results.

6. GSW ALGORITHM

In [5], a novel encryption algorithm known as the GSW for developing FHE algorithms which evade the high-priced relinearization phase in homomorphic multiplication was presented. As noted by [12], in some categories of circuits, the GSW algorithm possesses a lesser noise growth rate as compared with the others and, thus better proficiency and more grounded security.

This algorithm was later upgraded to obtain ring modifications of the GSW algorithm which are the fast fully homomorphic encryption in the west (FHEW) in 2014 as well as fast fully homomorphic encryption over the torus (TFHE) in 2016. The FHEW was the principal algorithm to illustrate bootstrapping(refreshing) of the ciphertext such that after each and every computation, it is conceivable to lessen the bootstrapping time to a small amount of a second and also immensely simplified bootstrapping by implementing another strategy to calculate Boolean gates on scrambled data [13]. Unlike the other algorithms, the FHEW encryption has KeyGen, Encrypt, NAND and Decrypt meant for key generation, encryption, homomorphic encryption as well and decryption respectively [14].

The KeyGen creates the secret key and the corresponding evaluation key that is utilized in encoding the message. Subsequently, the ciphertext is then operated upon by the homomorphic NAND operator using the evaluation key and then finally, the resulting ciphertext is outputted. Two years later, the efficacy of the FHEW was enhanced by the TFHE

algorithm, which also employs a variant of the bootstrapping procedure with methods similar to the FHEW. The TFHE allows implementing a very fast gate-by-gate bootstrapping by evaluating a random Boolean circuit made out of binary gates, such as, AND, OR, XOR, and NAND, just as Negation and Multiplexer gate over encrypted data without breaching data confidentiality.

The strapping approach of the TFHE does any limitation on the number of gates nor its compositions and thus enables any form of computations on the encrypted data [15]. The KeyGen generates a private-keyset and a cloud-keyset which both provide encryption and decryption capabilities but the cloud-keyset can be transferred into the clouds so that it can allow for secure homomorphic computations on encoded data. The private keyset enables encryption, as well as decryption on the data while the cloud keyset, evaluates a netlist of binary gates homomorphically at a pace of around 76 gates for each core, deprived of decoding its input. It is therefore adequate to deliver the input bits in place of the sequence of gates and the output ciphertexts of the output bits.

7. MEDICAL CLOUD COMPUTING

More healthcare providers are opting to work with vendors that provide cloud computing solutions for their digital records than ever before for the efficiency of the industry while decreasing costs [16]. Cloud computing makes medical record-sharing easier and safer, automates backend operations and even facilitates the creation, and maintenance of, telehealth apps. Suppose, a patient comes complaining of cough, chest pains and a headache. The doctor would then use his or her knowledge to diagnose what is wrong with the patient and record all the relevant details. Nonetheless, only the primary diagnosis would show up on the patient's chart. That is where cloud-based data analysis comes in handy, extracting data that would otherwise remain hidden. One of the main apprehensions of healthcare providers, irrespective of size, is healthcare data security. Data breaches and other cybercrimes can overwhelm a firm's revenue, threaten patient safety and damage the provider's reputation [17] [18].

A few researchers also proposed different applications for homomorphic encryption. For instance, [19], introduced a safe and sound image retrieval strategy for cloud computing dependent on homomorphic characteristics of the Paillier algorithm. A technique for protection of medical cloud computing, utilizing the fully homomorphic encryption was also introduced [20]. Their novel method is illustrated in Figure 6.

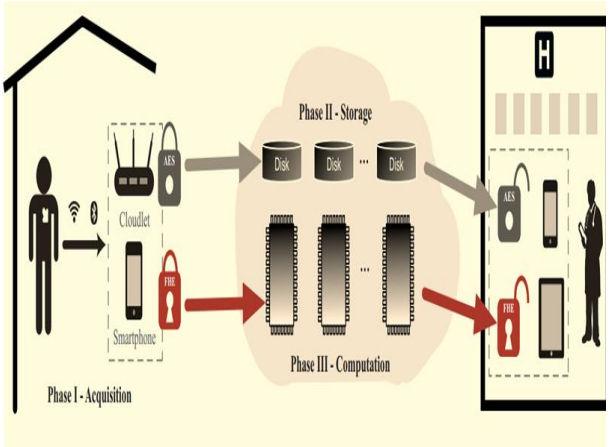

Figure 6: Medical Cloud Patient Monitoring System

According to [21], it is basic to keep up the protection of a scrambled database while performing computations on the encrypted data. Thus, they devised a framework that uses primitive circuits for encryption which was done for search and computes operations to maintain protection for the exposure of confidential data. It further described the advantages of using FHE over the usage of multiple encryption algorithms to maintain the privacy policy.

A Bayesian Classifier for a clinical decision support system to help in making critical decisions was also proposed in [22]. An addictive homomorphic proxy segregation algorithm can be used to encrypt patients' medical records. The method of retrieval used to highlight the top matching records for this algorithm is the Top-k method and the client-server connection terminates by the message-passing phenomenon [22].

The challenges encountered while dealing with encrypted data such as variable definition translation, instruction execution, loop handling and conditional terminations were highlighted and the idea of the encrypted auxiliary stack with the methods of encrypted pop and encrypted push used to handle them [23].

In [24], the researchers provided a homomorphic algorithm that can handle large message space by putting much emphasis on advances in existing algorithms by encoding it as a coefficient of polynomial and then performing the encryption on the encoded polynomial's coefficient.

8. Conclusion

The ciphertext recovery algorithm in the DGHV requires the movement of the secret key to the server, which appears to be horrendously shaky. While, the ciphertext recovery algorithm of Gentry's work is named Gen10 and GSW requests to present q to the server in spite of that using c mod q, where q is an irregular number and c is the ciphertext, yet the plaintext spills out.

In [1], the authors proposed an algorithm that solves the problem of ciphertext recovery successfully, without plaintext spill out, in light of the fact that the procedure of decoding utilizes the secret key p although the recovery procedure utilizes the whole number q, which is completely unique. Consequently, fulfils both the interest for ciphertext recovery and information security.

There are quite several types or evolutions of the Fully Homomorphic encryption algorithms with each of them addressing some peculiar problems of its predecessor. Some of the works reviewed so far used various implementation frameworks for securing cloud data using fully homomorphic encryption algorithms. In this research work, an implementation framework for the SDC algorithm will be proposed and its performance evaluation analyzed to protect data from unauthorized access, disclosure, modification and monitoring.

9. REFERENCES

[1] J. Li, S. Song, S. Chen and X. Lu, "A Simple Fully Homomorphic Encryption Scheme Available In Cloud Computing," IEEE, pp. 214-217, 2012.

[2] S. Goldwasser and S. Micali, "Probabilistic Encryption and How to Play Mental Poker Keeping Secret all Partial Information," ACM Symposium on Theory of Computing, 1982.

[3] M. Tebaa, S. El Hajji and A. El Ghazi, "Homomorphic Encryption Applied to the Cloud Computing Security.," in World Congress on Engineering, London, 2012.

[4] I. Jabbar and S. Najim, "Using Fully Homomorphic Encryption to Secure Cloud Computing," Internet of Things and Cloud Computing, pp. 13-18, 2016.

[5] C. Gentry, "Computing Arbitrary Functions of Encrypted Data," ACM, vol. 53, no. 3, pp. 97-105, 2010.

[6] C. Gentry, "A Fully Homomorphic Encryption Scheme," 2009.

[7] I. Jabbar and S. Najim, "Using Fully Homomorphic Encryption to Secure Cloud Computing," Internet of Things and Cloud Computing, pp. 13-18, 2016.

[8] C. Gentry, "Fully Homomorphic Encryption Using Ideal Lattices," ACM, pp. 169-178, 2009.

[9] K. M. Mohanty, "Secure Data Storage on the Cloud using Homomorphic Encrypyion.," in National Institute of Technology., Rourkela, 2013.

[10] M. M. Potey, C. A. Dhote and H. D. Sharma, "Homomorphic Encryption for Security of Cloud Data," in International Conference on Communication, Computing and Virtualization, 2016.

[11] D. R. Korda, E. D. Ansong and D. K. M. Hodowu, "Securing Data in the Cloud using the SDC Algorithm," International Journal of Computer Applications , pp. 24-29, 2021.

[12] V. Brakerski and V. Vaikuntanathan, "Lattice-Based FHE as Secure as PKE," in ITCS 2014, 2014.

[13] J. Alperin-Sheriff and C. Peikert, "Faster Bootstrapping with Polynomial Error," in In CRYPTO 2014, 2014.

[14] L. Ducas and D. Micciancio, "FHEW: Bootstrapping Homomorphic Encryption in less than a second," University of California, San Diego, 2014.

[15] I. Chillotti, N. Gama, M. Georgieva and M. Izabachène, "Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds.," Asiacrypt , pp. 3-33, 2016.

[16] BuiltIn, "BuiltIn," 17 April 2019. [Online]. Available: https://builtin.com/cloud-computing/cloud-computing-in-healthcare.

[17] Forsee Medical, "Cloud Computing in Healthcare," 28 September 2020. [Online]. Available: https://www.foreseemed.com/blog/cloud-computing-in-healthcare.

[18] D. K. M. Hodowu, D. R. Korda and E. D. Ansong, "An Enhancement of Data Security in Cloud Computing with an Implementation of a Two-Level Cryptographic Technique, using AES and ECC Algorithm," International Journal of Engineering Research & Technology, vol. 09, no. 09, pp. 639-650, 2020.

[19] O. Kocabas and T. Soyata, "Utilizing Homomorphic Encryption to Implement Secure and Private Medical Cloud Computing.," IEEE, 2015.

[20] O. Kocabas and T. Soyata, "Utilizing Homomorphic Encryption to Implement Secure and Private Medical Cloud Computing," IEEE, 2015.

[21] J. H. Cheon and M. Kim, "Optimized Search and Compute Circuits and their Application to Query Evaluation on Encrypted Data.," IEEE Transactions on Information Forensics and Security, pp. 188-199, 2016.

[22] C. Ayantika and I. Sengupta, "Translating Algorithms to handle Fully Homomorphic Encrypted Data on the Cloud," IEEE Transactions on Cloud Computing, vol. 6, no. 1, 2018.

[23] D. S. Dhokey and A. V. Deorankar, "Review on Usage of Homomorphic Encryption Technique," International Journal of Current Research, pp. 76377-76379, 2018.

[24] K. Aganya and I. Sharma, "Symmetric Fully Homomorphic Encryption Scheme with Polynomials Operations.," in International Conference on Electronics, Communication and Aerospace Technology. , Coimbatore: IEEE., 2018.