JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

HOMOMORPHIC ENCRYPTION: A COMPREHENSIVE REVIEW

[1]Kavya Joshi, [2]Minal Thakare, [3]Sunita Bangal
Department of Technology
Savitribai Phule Pune University, Pune.

Abstract:
Homomorphic encryption is a revolutionary cryptographic technique that enables computations on encrypted data without the need for decryption [1]. It provides a means to perform operations on sensitive data while maintaining privacy and security.
Homomorphic encryption allows users to perform computations on encrypted data, providing a solution for secure and privacy-preserving data processing. By leveraging mathematical algorithms, homomorphic encryption enables operations such as addition and multiplication on encrypted values. The encrypted result can then be decrypted to obtain the same outcome as if the operations were performed on the original plaintext data.
This powerful technique finds applications in various domains, including cloud computing, data analytics, and machine learning, where privacy and confidentiality are paramount. With homomorphic encryption, sensitive data can be outsourced to untrusted environments, allowing computations to be carried out without exposing the data to potential adversaries.
As this field continues to evolve, homomorphic encryption holds promise for transforming the way we handle sensitive information and preserve privacy in the digital age.

Keywords: **Homomorphic encryption · Machine learning · Privacy · Cryptographic technique**

Introduction

Homomorphic encryption, an innovative cryptographic concept, can be attributed to the ground-breaking work of mathematician Craig Gentry in 2009. However, the idea of performing computations on encrypted data without decryption was discussed earlier by Rivest, Adleman, and Dertouzos in 1978. Gentry's influential paper, "A Fully Homomorphic Encryption Scheme," introduced the first fully homomorphic encryption (FHE) scheme.

Before Gentry's work, partially homomorphic encryption (PHE) schemes had already been developed. PHE allowed specific computations like addition or multiplication on encrypted data while preserving privacy. Notable PHE schemes included RSA, supporting multiplication on encrypted data, and ElGamal, supporting multiplication and limited addition.

Gentry's FHE scheme revolutionized the field by enabling arbitrary computations on encrypted data without decryption [2]. However, the initial FHE scheme had significant computational complexity and performance overhead.

Further research focused on improving the efficiency and practicality of homomorphic encryption. In 2010, Gentry, along with Halevi and Smart, introduced the first somewhat homomorphic encryption (SHE) scheme. SHE allowed a limited number of computations on encrypted data, striking a balance between functionality and efficiency.

Since then, researchers have made substantial progress in enhancing the security and efficiency of homomorphic encryption. To put it in plain English, if you have two encrypted numbers, a and b, with Homomorphic Encryption, you can do operations on them like a + b = c. You still don't know what a, b or c is, but you can pass c back to the person who has the encryption key, and they can read it [3].

Various mathematical approaches, such as lattice-based cryptography and ring learning with errors (RLWE), have been explored to advance the field.

Homomorphic encryption is a form of encryption that allows computations to be performed on encrypted data without first having to decrypt it. The resulting computations are left in an encrypted form which, when decrypted, result in an output that is identical to that produced had the operations been performed on the unencrypted data. Homomorphic encryption can be used for privacy-preserving outsourced storage and computation. This allows data to be encrypted and out-sourced to commercial cloud environments for processing, all while encrypted [4].

The significance of homomorphic encryption in secure data processing lies in its ability to address the inherent trade-off between data privacy and computation. Traditional encryption methods require data to be decrypted before performing computations, which exposes the sensitive information to potential security risks. Homomorphic encryption, on the other hand, allows computations to be

conducted directly on encrypted data, eliminating the need to reveal the plaintext.

Various homomorphic encryption schemes exist, such as partially homomorphic encryption (PHE), somewhat homomorphic encryption (SHE), and fully homomorphic encryption (FHE). PHE permits computations like addition or multiplication on encrypted data, while SHE extends the range of operations that can be executed. On the other hand, FHE enables unrestricted computations on encrypted data. In summary, PHE supports limited computations, SHE allows for more operations, and FHE enables arbitrary computations on encrypted data.

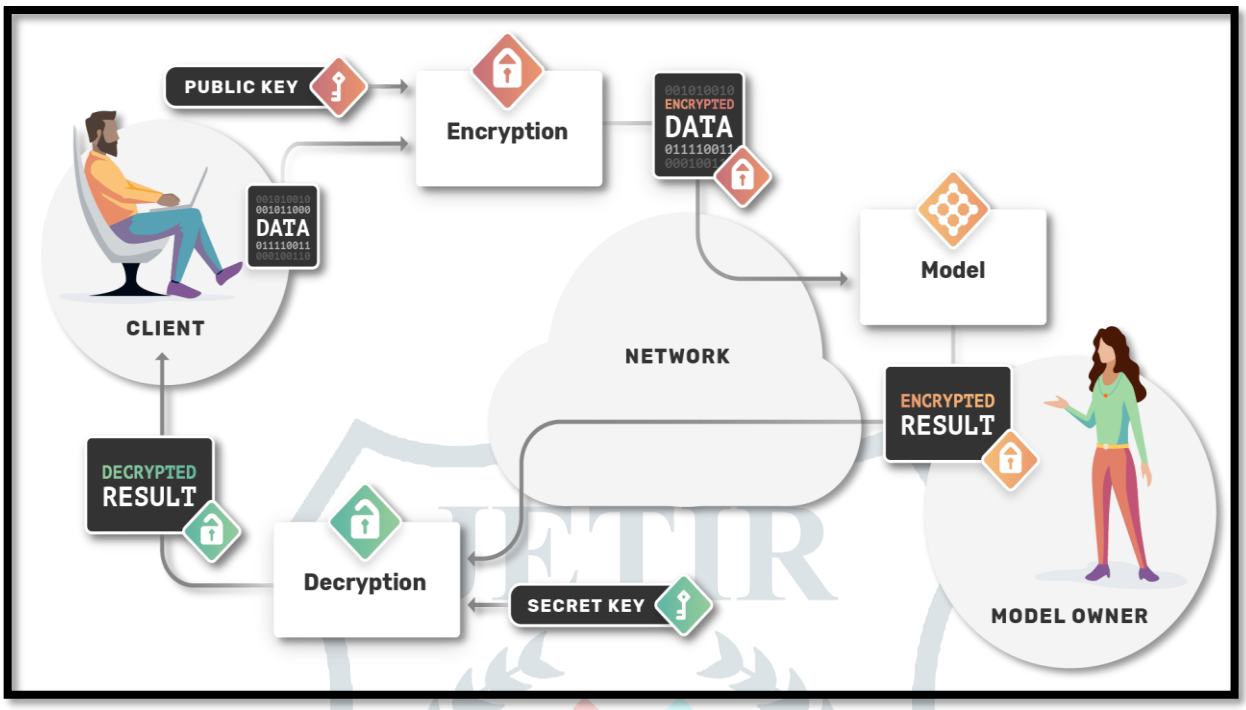


Fig.1 Homomorphic Encryption [5]

Mechanism

The homomorphic encryption mechanism involves several fundamental steps:

Key Generation:

In this step, a user generates a pair of encryption keys, which typically consists of a public key for encryption and a private key for decryption. The public key is intended for use in encrypting the data, while the private key must be kept confidential and is used for decrypting the ciphertext.

Encryption:

During the encryption phase, the data that needs to be processed is transformed into ciphertext using the public key. This encryption process ensures that the original information remains concealed and protected.

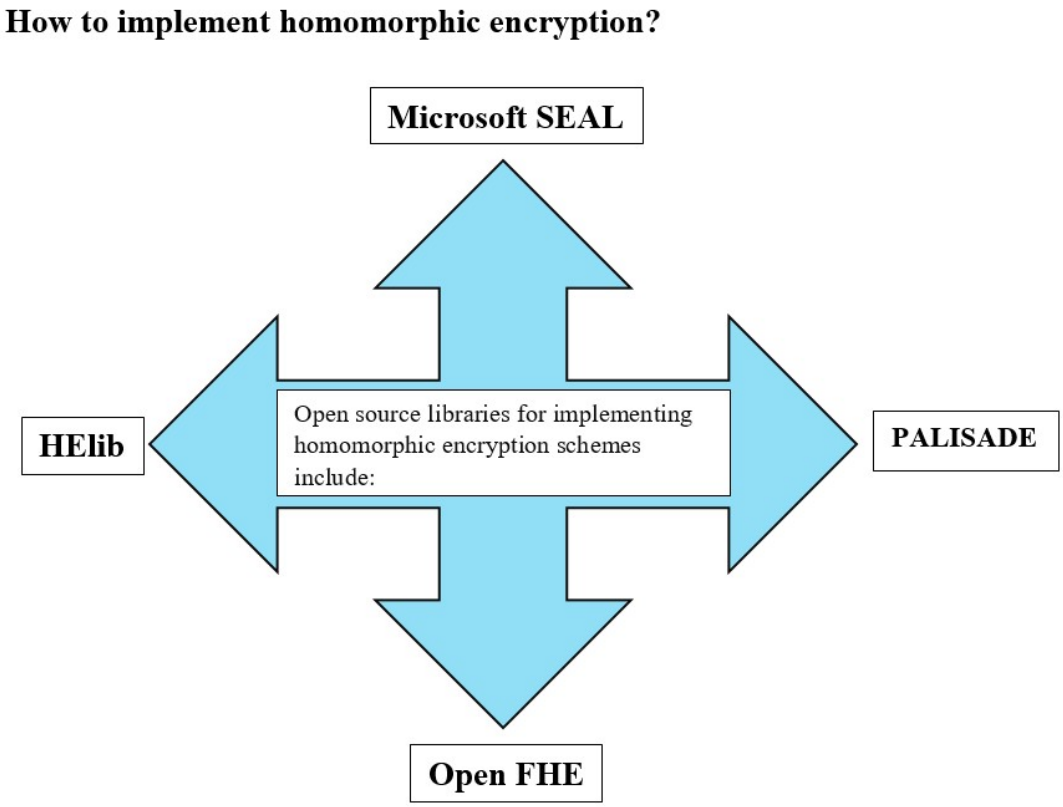
Computation:

Homomorphic encryption allows specific mathematical operations, such as addition and multiplication, to be directly applied to the encrypted data. These computations are carried out on the ciphertext, preserving the privacy and secrecy of the underlying information throughout the process.

Decryption:

Once the desired computations have been performed on the encrypted data, the resulting ciphertext is decrypted using the private key. This decryption process retrieves the final outcome in its original plaintext form, enabling the recipient to access the computed results.

To summarize, the general mechanism of homomorphic encryption involves generating encryption keys, encrypting the data, performing computations on the encrypted data, and finally decrypting the ciphertext to obtain the desired result. This mechanism enables secure computation while preserving the privacy and confidentiality of the data being processed.



Types of Homomorphic encryption

Homomorphic encryption schemes can be classified into three main types: partially homomorphic encryption (PHE), somewhat homomorphic encryption (SHE), and fully homomorphic encryption (FHE) [6]. Each type offers different levels of computational capabilities while preserving the privacy of the encrypted data.

Partially Homomorphic Encryption (PHE)

PHE schemes allow computations of a specific type to be performed on encrypted data. The most common operations supported by PHE schemes are addition and multiplication.

Addition: PHE schemes enable the addition of two ciphertexts, which corresponds to adding the corresponding plaintext values. For example, if we have encrypted values c1 and c2, we can compute c1 + c2 without decrypting the ciphertexts.

Multiplication: Some PHE schemes also support the multiplication of ciphertexts, which corresponds to multiplying the corresponding plaintext values. For example, if we have encrypted values c1 and c2, we can compute c1 * c2 without decrypting the ciphertexts.

Notable PHE schemes include the RSA cryptosystem, which supports multiplication on encrypted data, and the El Gamal cryptosystem, which supports multiplication and limited addition.

Let's consider a simple PHE scheme that supports addition on encrypted data. Suppose Catrina wants to perform a computation on her sensitive data while keeping it encrypted. She generates a key pair: a public key for encryption and a private key for decryption.

Key Generation:

Catrina generates a public key (PK) and a private key (SK).

Encryption:

Catrina encrypts her data using the public key.

For instance, if Catrina wants to encrypt the number 7, she applies the encryption algorithm with the public key: Enc (7, PK) = C1.

Computation (Addition):

Catrina wants to add the encrypted value C1 to another encrypted value C2.

She performs the addition operation on the ciphertexts: C1 + C2 = C3.

Decryption:

Catrina can only decrypt the final result using her private key. If she applies the decryption algorithm with the private key: Dec (C3, SK) = 10.

Catrina obtains the decrypted result, which is 10, without ever decrypting the individual ciphertexts.

In this example, the PHE scheme enables Catrina to perform the addition operation on the encrypted data without the need to decrypt it. The privacy of her sensitive data is preserved throughout the computation process.

Somewhat Homomorphic Encryption (SHE)

SHE schemes expand the set of operations that can be performed on encrypted data beyond simple addition or multiplication. They offer a limited level of computational capabilities while still preserving the privacy of the data.

Multiple Addition/Multiplication: SHE schemes allow performing multiple additions and multiplications on encrypted data, either in sequence or concurrently.

Evaluation Circuit: SHE schemes often define an evaluation circuit that specifies a set of allowed computations on encrypted data. The circuit limits the types of operations that can be performed, but it can be designed to support complex computations.

SHE schemes strike a balance between functionality and efficiency, as they allow for more operations than PHE schemes while still maintaining a reasonable computational overhead. The first SHE scheme was introduced in 2010 by Gentry, Halevi, and Smart.

Consider an SHE scheme that supports both addition and multiplication on encrypted data, but with a restriction on the number of operations. Let's suppose Catrina wants to perform computations on her sensitive data using this scheme:

Key Generation:
Catrina generates a key pair: a public key (PK) and a private key (SK).
Encryption:
Catrina encrypts her data using the public key.
For instance, if Catrina wants to encrypt the number 7, she applies the encryption algorithm with the public key: Enc (7, PK) = C1.
Computation (Addition and Multiplication):
Catrina wants to perform an addition and a multiplication operation on the encrypted data.
First, she adds C1 to another encrypted value, let's say C2: C1 + C2 = C3.
Next, she multiplies C3 by a constant, for example, 3: 3 * C3 = C4.
Decryption:
Catrina can only decrypt the final result using her private key. If she applies the decryption algorithm with the private key: Dec (C4, SK) = 39.
Catrina obtains the decrypted result, which is 39, without ever decrypting the individual ciphertexts.
In this example, the SHE schemes allows Catrina to perform addition and multiplication operations on the encrypted data. However, the number of operations she can perform may be limited by the specific SHE schemes. Once again, it's important to note that the example represents a simplified scenario, and real-world SHE schemes may have different characteristics and constraints.

Fully Homomorphic Encryption (FHE)
FHE schemes enable arbitrary computations to be performed on encrypted data, allowing for unlimited operations without decryption. FHE is the most powerful and versatile type of homomorphic encryption.
Arbitrary Computations: FHE schemes support a wide range of computations, including addition, multiplication, comparison, branching, and other complex operations.
Turing-Complete: FHE schemes are Turing-complete, meaning they can perform any computation that can be expressed as an algorithm.
FHE was initially proposed by Craig Gentry in 2009. However, early FHE schemes were computationally intensive and impractical. Subsequent research has focused on improving the efficiency and security of FHE, making it more feasible for real-world applications.
It's worth noting that as we move from PHE to FHE, the computational complexity and performance overhead generally increase. However, advancements in homomorphic encryption continue to drive the development of more efficient and practical schemes, expanding the possibilities for privacy-preserving computations on encrypted data.
Consider an FHE scheme where Catrina wants to perform computations on her sensitive data while keeping it encrypted:
Key Generation:
Catrina generates a key pair: a public key (PK) and a private key (SK).
Encryption:
Catrina encrypts her data using the public key.
For example, if Catrina wants to encrypt the number 10, she applies the encryption algorithm with the public key: Enc (10, PK) = C1.
Computation (Arbitrary):
Catrina wants to perform an arbitrary computation, such as computing the square of the encrypted value.
She applies the computation function directly on the ciphertext: Square(C1) = C2.
Decryption:
Catrina can only decrypt the final result using her private key. If she applies the decryption algorithm with the private key: Dec (C2, SK) = 100.
Catrina obtains the decrypted result, which is 100, without ever decrypting the individual ciphertexts.
In this example, the FHE scheme allows Catrina to perform an arbitrary computation (squaring) on the encrypted data without the need to decrypt it. The privacy of her sensitive data is maintained throughout the computation process.

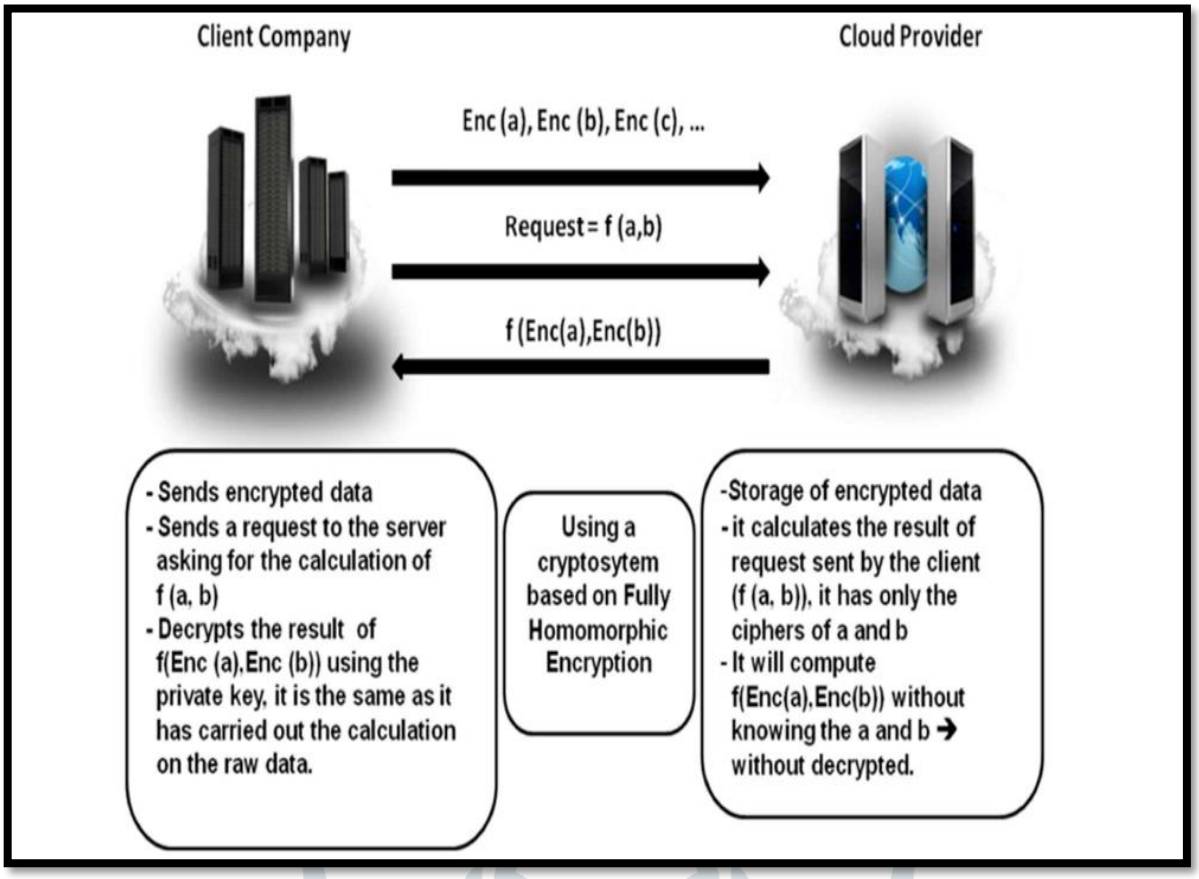


Fig. 2 Fully Homomorphic Encryption applied to the Cloud Computing [7]

Brief comparison between PHE, FHE and SHE:

Parameter	Partial HE	Somewhat HE	Fully HE
Type of operation	Either addition or multiplication	Multiple addition and multiplication	Both addition and multiplication
No. of Computations	Limited	Limited	Unlimited
Performance	Faster and more compact	Depends on the computations	Slower
Versatility	Low	Restricted functionality	high
Computational efforts	Less	Higher than PHE	More
Ciphertext size	Small	Larger compared with PHE and FHE	Large
Example	Unpadded RSA,ElGamal	Paillier encryption scheme	Gentry scheme

Improve Homomorphic Encryption performance:

Under the DARPA DPRIVE (Data Protection in Virtual Environments) program, Intel plans to design an application-specific integrated circuit (ASIC) accelerator to reduce the performance overhead currently associated with fully homomorphic encryption. When fully realized, the accelerator could deliver a massive improvement in executing FHE workloads over existing CPU-driven systems, potentially reducing cryptograms' processing time by five orders of magnitude.

Applications:

Homomorphic encryption finds diverse applications across various domains due to its unique capability of performing computations on encrypted data without the need for decryption. Some notable use cases include:

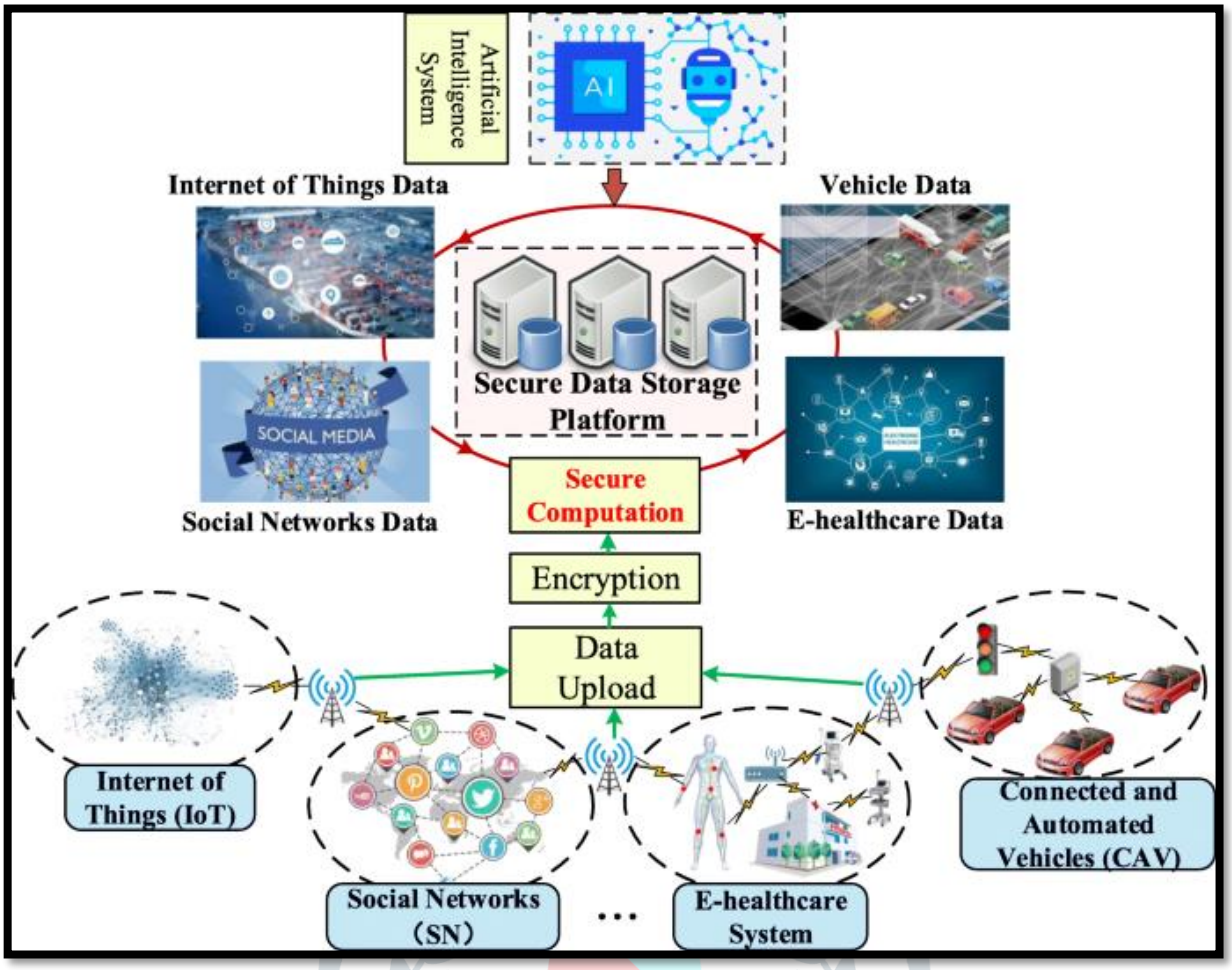


Fig. 3 Applications of Homomorphic Encryption [8]

- **Secure Cloud Computing**

Homomorphic encryption allows for secure data processing and computation outsourcing in cloud environments, ensuring the confidentiality of sensitive information.

- **Privacy-Preserving Machine Learning**

By enabling machine learning models to operate on encrypted data, homomorphic encryption facilitates collaborative data analysis while preserving the privacy of individual data contributors.

- **Secure Computation Outsourcing**

Homomorphic encryption enables secure delegation of computations to external parties, protecting the privacy of sensitive data while allowing for efficient and confidential data processing.

- **Private Information Retrieval**

Homomorphic encryption can be applied in systems that retrieve specific data from databases without revealing the user's query or granting direct access to unencrypted data.

- **Privacy-Preserving Data Sharing**

Homomorphic encryption enables secure collaboration and data sharing among multiple parties while maintaining the privacy of individual data. It allows for joint data analysis and processing while preserving the confidentiality of sensitive information.

- **Secure Multiparty Computation**

SMPC represents a subfield of cryptography that performs data computation by distributing the data between different parties [9]. Each party applying the algorithm to its secure data without knowing the rest of the data results in privacy preservation. This is particularly useful in scenarios such as voting protocols, auctions, and data aggregation, where privacy and confidentiality are crucial.

- **Secure Internet of Things (IoT)**

Homomorphic encryption can be employed in IoT systems to safeguard the privacy of sensitive data generated by IoT devices. It ensures that data can be securely processed and analysed while preserving the confidentiality of IoT-generated information.

- **Privacy-Enhanced Data Storage**

By utilizing homomorphic encryption, data storage privacy can be enhanced. The encryption of data prior to storage in a remote or untrusted environment ensures that the information remains confidential even if the storage system is compromised.

Challenges:

Homomorphic encryption, despite its significant advantages in secure computation on encrypted data, presents several hurdles that need to be tackled:

- **Performance Overhead**

Homomorphic encryption methods often introduce substantial computational overhead compared to traditional encryption techniques. Computation on encrypted data necessitates intricate mathematical operations, resulting in slower processing times and increased resource demands.

- **Key Management**

Effective management of both public and private keys is imperative in homomorphic encryption schemes. Secure key generation, distribution, and storage are vital for preserving the integrity and confidentiality of the encrypted data. As the number of participants and the complexity of computations grow, key management becomes increasingly challenging.

- **Noise Accumulation**

Homomorphic encryption introduces noise during computation, and this noise accumulates with each operation. The accumulation of noise can impact the accuracy of the computed results, necessitating the application of noise reduction algorithms or other techniques to mitigate its effects.

- **Limited Functionality**

Although fully homomorphic encryption (FHE) schemes allow for arbitrary computations, they often have practical limitations concerning computation depth or input size. These limitations can restrict the types of computations that can be efficiently performed using homomorphic encryption.

- **Security Assumptions**

Homomorphic encryption relies on specific mathematical assumptions to ensure security. If these assumptions are compromised, the security of the encrypted data may also be compromised. Continuous research is essential to fortify homomorphic encryption schemes against emerging threats and attacks.

- **Scalability**

As data size and the number of participants increase, scalability becomes a challenge for homomorphic encryption. Ensuring efficient processing of large-scale computations on encrypted data while maintaining acceptable performance levels can be intricate.

Addressing these challenges necessitates ongoing research and development in the field of homomorphic encryption. Advancements in algorithm optimization, key management practices, noise reduction techniques, security analysis, and scalability solutions are crucial to overcoming these obstacles and facilitating broader adoption of homomorphic encryption in practical applications.

Recent Developments and Future Directions

Homomorphic encryption has witnessed significant advancements in recent years, driven by ongoing research and emerging trends in the field. Two major optimisations are possible and are work in progress, namely the bootstrapping step can be highly improved by using a much better SIMD approach than the current state-of-the-art; and the actual homomorphic system itself should be replaced by our "t-adic approach to FHE", which keeps the size of the ciphertext minimal w.r.t. the size of the noise it contains [10]. Here are some notable advancements:

- **Improved Efficiency**

Significant strides have been made by researchers in improving the efficiency of homomorphic encryption schemes. These advancements encompass the creation of more streamlined algorithms, more effective techniques for managing keys, and methods for reducing the impact of noise. The objective is to minimize the computational burden and make homomorphic encryption more feasible and applicable in real-world scenarios.

- **Lattice-Based Cryptography**

Lattice-based cryptography has emerged as a highly promising method for constructing secure and efficient homomorphic encryption schemes. These schemes built on lattice-based cryptography offer robust security assurances and demonstrate resilience against quantum attacks, which has led to a significant focus on them in ongoing research. Lattice-based cryptography provides a solid groundwork for the development of more effective and practical homomorphic encryption schemes.

- **Hardware Acceleration**

Researchers are actively investigating methods of hardware acceleration to enhance the performance of homomorphic encryption. Specifically, they are exploring the utilization of dedicated hardware architectures like FPGAs (Field-Programmable Gate Arrays) and GPUs (Graphics Processing Units) to expedite the computational processes involved in homomorphic encryption. These hardware accelerators facilitate quicker computations and alleviate the computational burden associated with homomorphic encryption.

- **Multi-Party Computation (MPC) and Homomorphic Encryption**

The combination of homomorphic encryption and secure multi-party computation (MPC) is a rising trend in the field. MPC techniques allow multiple parties to collaborate and collectively carry out computations on their private data while maintaining privacy. By integrating homomorphic encryption with MPC, the security and efficiency of secure multiparty computations can be improved.

- **Quantum-Safe Homomorphic Encryption**

As quantum computing becomes a reality, there is an increasing emphasis on the development of homomorphic encryption schemes that can withstand attacks from quantum computers. These quantum-safe schemes are specifically designed to ensure the long-term security of encrypted data. Ongoing research in this field aims to create efficient homomorphic encryption schemes that are resilient against quantum attacks.

- **Standardization Efforts**

Standardization plays a vital role in promoting the acceptance and compatibility of homomorphic encryption schemes. Currently, there are ongoing initiatives to establish standardized protocols and frameworks for homomorphic encryption. These efforts aim to ensure that different implementations of homomorphic encryption can work seamlessly together and facilitate the integration of homomorphic encryption into various applications.

Conclusion:

In summary, the paper extensively covered the fundamental principles of homomorphic encryption, encompassing its three primary categories: partially homomorphic encryption, somewhat homomorphic encryption, and fully homomorphic encryption. The advantages and limitations of each type were thoroughly explored, emphasizing the range of computational capabilities and security assurances they offer.

Furthermore, the review delved into the diverse applications of homomorphic encryption across different domains, such as secure cloud computing, privacy-preserving machine learning, and secure outsourced computation. These applications exemplify how homomorphic encryption can effectively tackle the crucial challenges associated with data privacy and security in today's interconnected world.

The study also provided an overview of the latest advancements and state-of-the-art techniques in homomorphic encryption. Notably, it discussed recent developments in lattice-based encryption schemes, optimizations to enhance computational efficiency, and methods for mitigating security vulnerabilities. These advancements have significantly enhanced the feasibility and

performance of homomorphic encryption, bringing it closer to practical implementation in real-world scenarios.

To conclude, homomorphic encryption shows immense potential in addressing the privacy and security challenges that arise when dealing with sensitive data. It presents a ground-breaking method that enables computations to be conducted on encrypted data, ushering in fresh opportunities for secure and privacy-preserving data analysis and computation. As ongoing research and development progress, homomorphic encryption has the capability to become an indispensable component of the data security landscape, empowering individuals and organizations to harness the benefits of computation while ensuring the confidentiality of their data.

References

[1] A. Shende. [Online]. Available: https://adityashende17.medium.com/the-revolutionary-encryption-technique-enabling-secure-computation-on-encrypted-data-a4837fc5326b.

[2] [Online]. Available: https://academic-accelerator.com/encyclopedia/homomorphic-encryption.

[3] [Online]. Available: https://venafi.com/blog/what-are-latest-developments-homomorphic-encryption-ask-experts/.

[4] [Online]. Available: https://venafi.com/blog/what-are-latest-developments-homomorphic-encryption-ask-experts/.

[5] [Online]. Available: https://blog.openmined.org/what-is-homomorphic-encryption/.

[6] K. Munjal and R. Bhatia. [Online]. Available: https://link.springer.com/article/10.1007/s40747-022-00756-z.

[7] M. TEBAA and S. E. H. . [Online]. Available: https://arxiv.org/ftp/arxiv/papers/1409/1409.0829.pdf#:~:text=Homomorphic%20Encryption%20Applied%20to%20Cloud%20Computing%20Security&text=The%20client%20will%20need%20to,data%20stored%20in%20the%20Cloud..

[8] [Online]. Available: https://cybersecurity.springeropen.com/articles/10.1186/s42400-020-00057-3.

[9] N. Khalid. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S001048252300313X.

[10] J. Fan and F. Vercauteren. [Online]. Available: https://www.researchgate.net/publication/267862690_Somewhat_Practical_Fully_Homomorphic_Encryption.