Some Practical Applications of Fully Homomorphic Encryption

Elisa Giurgea[1,a], **Tudor Hutu**[2,b], **Emil Simion**[3,c]

[1,2] Faculty of Computer Science, Alexandru Ioan Cuza University, Iasi, Romania
[3] Polytechnic University of Bucharest, Bucharest, Romania

*

[a] elisa.giurgea@gmail.com
[b] tudorhutu2012@gmail.com
[c] emil.simion@upb.ro

Abstract

In the current context of the increasing need for data privacy and quantum computing no longer being just a novel concept, Fully Homomorphic Encryption presents us with numerous quantum-secure schemes which have the concept of enabling data processing over encrypted data while not decrypting it behind. While not entirely usable at the present time, recent research has underlined its practical uses applied to databases, cloud computing, machine learning, e-voting, and IoT computing. In this paper, we are covering the current status of research and presenting the leading implemented solutions for subjects related to data privacy in the before-mentioned areas while emphasizing their positive results and possible drawbacks subsequently discovered by the research community.

Keywords: *Fully Homomorphic Encryption; Practical Applications of FHE; Databases; Cloud Computing; Machine Learning; E-votes; IoT Computing*

1. Introduction

1.1. Background

Data is the driving force behind many if not all things around us, good data enables us to establish patterns, baselines, and benchmarks, which aid in making informed predictions and recommendations. For instance, utilizing data from a user's search history allows search engines to provide more personalized and relevant content recommendations, thereby improving the overall user experience and reducing the time and effort required for discovering new information.

This is an illustrative instance of the delicate balance between utilizing the benefits that come with providing certain services with personal data, and the potential risks associated with exposing sensitive information or utilizing it for unintended purposes.

The proliferation of data, both beneficial and harmful, has driven the demand for methods to allow access to services that utilize personal data while preserving its privacy. Currently, information security is directed toward exploring new techniques to achieve dependable data confidentiality and efficiently implement them, taking into account the move toward quantum computing.

1.2. What is Fully Homomorphic Encryption and what is its role?

A strong contender has proven to be Fully Homomorphic Encryption (FHE), referred to as the holy grail of cloud security by Gentry, Van Dijk, and many others.

But what is it all about?

FHE is a form of encryption that allows computation to be performed on encrypted data, like searching for strings within files, without the need to decrypt it first. This property is useful in situations where the data needed to be processed is sensitive and it is not desirable for the service in question (or third-party entities or employees) to have access to the decrypted data while processing it.

A report from an employee at Google in 2010 described a former engineer abusing his privileges to view private information. He used this information to stalk teenage girls and spy on their chat sessions. Many employees of cloud services have the required privileges to view the end user's data, which raises serious privacy and security concerns for the majority of users who utilize these services.

FHE brings a potential solution to this and many other problems concerning data privacy.

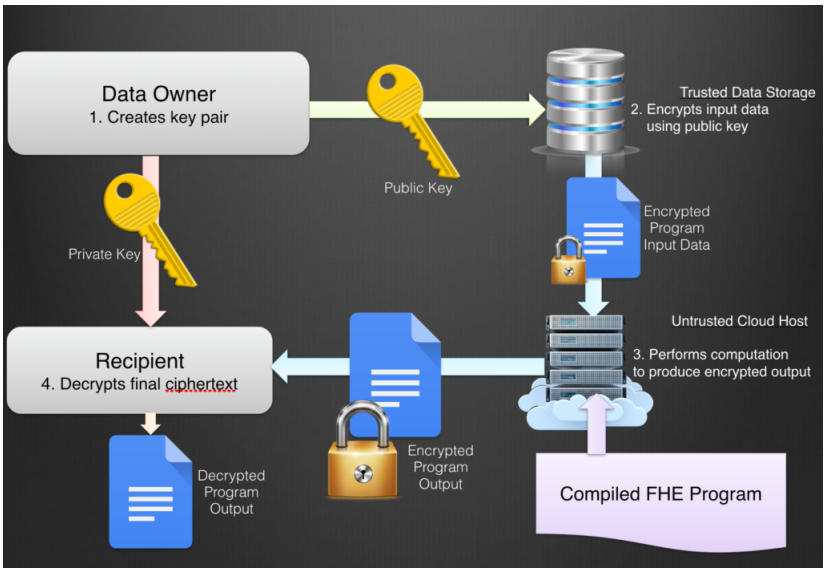

Fig. 1 - An overview of how FHE works in general [32]

Fully Homomorphic Encryption could be formalized in the following way:

$Enc(x) \rightarrow f(Enc(x)) = Enc(f(x))$, where:

- x is the sensitive data, the plaintext.
- $Enc(x)$ is the encryption applied on the said data x to obtain a ciphertext.
- $f(Enc(x))$ is the data processing applied over the encrypted data.
- $f(Enc(x)) = Enc(f(x))$ is the condition needed for the encryption scheme to be a homomorphic one, as this easily enables the person who provided the data in the first place to be the one and only one who is able to get the result of the processing which is contained in $f(x)$, being the only one with the knowledge over the encryption parameters applied over x and $f(x)$ (the encryption algorithm Enc), thus being the only one being able to successfully decrypt it.

The other condition that a cryptosystem has to have to be a Fully Homomorphic Encryption Scheme is to support arbitrary computation, which can be formalized as such:

A fully Homomorphic Encryption has to preserve a ring structure [9], meaning that we have a ring $(R, +, *)$, where R are our bits on which we operate, $(R, +)$ is an abelian group, while $(R, *)$ is a monoid. Having both addition and multiplication possible over the bits makes it possible to support and create $NAND$ gates, in which case we can derive every other boolean gate, thus, being able to do every computation on the encrypted data. If there is only one operation supported, we have Partially Homomorphic Encryption and Somewhat Homomorphic Encryption, which both present notable restrictions on the range of functions that are directly computable over ciphertexts.

1.3. Fully Homomorphic Encryption and Bootstraping

Gentry, who proposed the first plausible construction of a Fully Homomorphic Encryption scheme based on lattices [2], has also introduced in his work the concept of bootstrapping [2], which we will refer to later on in this article.

So what does bootstrapping imply?

Bootstrapping [3] is the process of refreshing a ciphertext in order to produce a new ciphertext that encrypts the same message, but with a lower level of noise so that more homomorphic operations can be evaluated on it. This operation is at the very core of any FHE schemes known to date and consists of homomorphically evaluating the decryption circuit of the scheme. In simpler terms, the process is akin to decrypting the encrypted message using the secret key and then re-encrypting it, with the difference that the secret key is not disclosed and instead replaced with an encrypted version of it, referred to as the bootstrapping key.

This requires an additional hardness assumption, called circular security assumption, meaning that we must assume it is safe to publish an encryption of the secret key under itself.

Although this assumption is still not well studied and understood, and it is sometimes regarded with suspicion, no attacks that exploit this extra piece of information have been proposed.

1.4. What is the current state of the possible FHE applications in real life?

Currently, the practical applications of FHE are extensively researched and there are already some available implementations of possible solutions for issues encountered in several domains: Databases, Cloud Computing, Machine Learning, E-votes, and IoT Computing. In this paper, we will underline the benefits of FHE in certain given scenarios and we will cover the current status of the implementations proposed for the enumerated domains, while keeping a realistic view of the progress made, as, as promising as it is, FHE still presents a very important disadvantage in its inefficiency, a disadvantage for which possible solutions are currently researched.

2. Databases

As a starting point, the most general application of FHE is Database security, which is the root of all the other subsequent applications.

Data is stored in databases and queries are applied over them in order to retrieve and manipulate data in the database. Databases, especially in the current context of big data being stored in databases managed by cloud providers, present a growing concern regarding the potential misuse of their data, particularly in instances where data privacy regulations and confidentiality agreements are violated without the user's consent. To ensure the protection of sensitive information, it is crucial to implement robust algorithms that enforce data confidentiality, privacy, and integrity, effectively preventing unauthorized third-party access to the stored data.

FHE has shown a growing potential to enable secure databases by allowing computations to be performed on encrypted data without the need to decrypt it first or even applying encrypted computations over data in order to hide the queried parameters.

While currently there are several non-FHE-based approaches for private query processing (CryptDB [27], Monomi [28]) and searchable encryption (OpenSSE [29]) implemented, FHE has been researched extensively and several implementations have been proposed to address the beforementioned issues.

2.1. Private Query Processing

Software systems often require sensitive data from users in order to construct a query to be run on the system's database, but with the increasing issue of data privacy even querying presents a source of data that can be unlawfully retrieved and used by third parties to inflict harm.

Take for example the process of getting an internet domain for a personal website or a company. Firstly, a user has to search for an available domain with a fitting name, in order to do so, the user is required to disclose the domain name. The domain will hereby be present in the query and, for this particular scenario, we will consider that the search history of each user is stored in a large database available for the employees with production access to the application's server. One ill-intended employee might check the logged searches, see that that domain has been searched for in the last few seconds, and make a distasteful prank by preemptively registering the new domain and thereby depriving the user who searched for the then-available domain of the registration privilege. Thus, the user can no longer choose the desired domain for their website. This is a light example of how a query can disclose private information that might not be considered as being sensitive enough until proven the contrary.

In principle, FHE schemes evaluate arbitrary functions over encrypted data, which makes way for one of the most promising applications of such schemes, searching over data without compromising the confidentiality of data.

In 2014, Jung Hee Cheon, Miran Kim, and Myungsun Kim, have attempted a new approach [5] to obtaining a more general solution to the FHE-based search problem. The main idea behind the proposed solution is to find a predicate to efficiently represent a search condition (for example, equality test and greater-than comparison) and then to evaluate that predicate at each FHE ciphertext. The evaluation result is also an encryption of 1 or 0.

For instance, considering a predicate $EQTest$ for testing the equality of two ciphertexts, $c1$ and $c2$, if the plaintext of $c1$ is equal to the plaintex of $c2$, then the $EQTest$ predicate outputs an encryption of 1; otherwise, it outputs an encryption of 0.

Then, applying a few additional computations to the equality test produces only the search result of interest. Considering a list of ciphertexts $\bar{R} = c_1, ..., c_n$, where $c_i = Enc(a_i)$ for a plaintext message ai and an FHE encryption algorithm Enc. For a fixed value v, anyone can then compute

$\langle c_i \cdot EQTest(ci, Enc(v))\rangle_n^{i=1}$

Under the assumption that a relation $R(A)$ has a state $r(R) = a_1, ..., a_n$ and that $\bar{R}$ in the clear is equal to $r(R)$ as a set, the expression is semantically equivalent to the following pseudo-SQL expression:

```
select * from R where A=v
```

Arguably, this FHE-based technique allows any SQL statement to be privately processed.

The solution proposed by Jung Hee Cheon, Miran Kim, and Myungsun Kim has been a strong base with many issues of efficiency extensively covered in their research. Based on their paper, other researchers have analyzed the proposal and contributed with further optimization efficiency-wise within the FHE context, possible expansion of the plaintext space, the efficiency of other predicates, such as `greather-than` and `min-max`, the reduction of the communication cost.

Most of these issues have been underlined and possible improvements have been proposed by Myungsun Kim, Hyung Tae Lee, San Ling, and Huaxiong Wang in their further work regarding an in-depth analysis of FHE-based private queries [6]. In their paper they have successfully demonstrated that the equality test algorithm consumes the minimum multiplicative depth when the plaintext space of the FHE scheme is a field of characteristic 2. Then, they went on to develop a series of private query processing protocols for conjunctive, disjunctive, and threshold queries on encrypted databases which extend the support of the initial implementation of FHE-based queries proposed by Cheon. Moreover, they have proposed a method of reducing the communication overhead from n to δn ciphertexts when δ, the upper bound on the selectivity of a database, is known.

Another real-world scenario that is currently under development for actual practical use is

Private Text Search on Public Medical Literature Databases (PDSPMD) [7]. In this case, the scenario presents public search engines which are often used by medical professionals to check public bibliographic databases. Doctors and pharmacists search for information on these databases using clear queries, which, similarly to the first proposed scenario, data in the queries can leak sensitive corporate information related to intellectual property and not only. If the employee of a pharmaceutical company searches for information regarding the drug toxicity of a specific molecule, it might leak that the company is working on developing a new drug based on that molecule. Also, personal privacy issues are also a factor, as, having the same context, query data can leak private information which can be later used for personal identification or to extract sensitive information regarding a medical condition.

Having this picture in mind, in 2021, Yassine Abbar, Pascal Aubry, Thierno Barry, Sergiu Carpov, Sayanta Mallick, Mariem Krichen, Damien Ligier, Sergey Shpak3, and Renaud Sirdey have come up with PDSPMD.

PDSPMD is a client-server application that enables the end user to encrypt sensitive query data before sending the query to the public search server, from which the user will then obtain the ranking documents result with the minimum information leakage.

The application showcases several methods to minimize the user query data leakage to the server and to reduce the time of the FHE computation by combining it with Private Information Retrieval (PIR). The server side of the application ranks the documents with the help of Term Frequency–Inverse Document Frequency (TTF-IDF) ranking model. The built prototype showcases the use of privacy driven text search environment using FHE and also Private Information Retrieval (PIR), which, when combined, make the FHE computations subsequently reduced, and the transfer of encrypted data using FHE is also reduced, as FHE is used to compute the TF-IDF of the documents to find the most relevant results, but only returns the document id. Moreover, PIR is further used to retrieve the full documents from the server without revealing which documents were retrieved.

The revolutionary part introduced in PDSPMD is the implementation of TF-IDF over an FHE scheme. In their work [7], the formalization of TF-IDF in FHE form is closely detailed in the *Implementing TF-IDF over FHE* chapter and its advantages efficiency-wise do not take away from the security level of the chosen FHE scheme.

PDSPMD was implemented using the Cingulata toolchain [30], which is a compiler toolchain and RTE for running C++ programs over encrypted data by means of FHE. For this proposed application, Cingulata was configured to use the Fan-Vercauteren (Fan and Vercauteren, 2012) IND-CPA secure schemes as the FHE backend. Under this scheme, the execution performances are highly multiplicative depth dependent.

By using these methods together, PDSPMD manages to perform rather well in terms of execution time, averaging 40 seconds for all the tests, which is a great improvement from previous works and brings the application closer to being a usable tool. Its only drawback so far seems to be the high compilation time, but given that it is a one-time offline operation for the system to be working, this is not a breaking disadvantage.

2.2. Searchable Encryption

While still not being fully usable as practical implementations, due to efficiency issues, several promising solutions to searchable encryption using FHE have been proposed and further analyzed.

In 2011, Youssef Gahi, Mouhcine Guennoun, and Khalil EL-Khatib have presented the implementation [4] of a secure database system using FHE schemes proposed by Gentry to test the feasibility of querying encrypted databases and the time complexity of the information retrieval process.

Their implementation shows that it is possible to perform SQL queries over an encrypted database. For example, the user can specify a search criterion through a database. Then, the client software encrypts the parameters of the query, corresponding to the search criterion, and sends it to the appropriate server. The server retrieves the requested record (blind processing) from

the database and returns it to the client. The client software decrypts the record and displays it to the user. Thus, it successfully proves the possibility of performing successful SQL queries over data encrypted using FHE schemes, yet falls short performance-wise, showing that the current schemes do not provide a suitable time of statement execution for real-time transactions that involve large databases.

The previously mentioned work [7] of Yassine Abbar, Pascal Aubry, Thierno Barry, Sergiu Carpov, Sayanta Mallick, Mariem Krichen, Damien Ligier, Sergey Shpak3, and Renaud Sirdey where they have proposed the PDSPMD application, also contains a practical implementation of searchable encryption using the Fan-Vercauteren FHE scheme.

While private query processing does hide the personal information of the user who performs the query, a database with plain confidential data also poses a clear security risk, so, in addition to the private query processing, they decided to take their work a step further and have the platform support end-to-end encryption (E2EE) which means that the database over which the private queries are performed should also be encrypted.

By analyzing how a server that wants to rank documents where both the corpus database and the queries are encrypted using FHE, they have sketched its behavior: Firstly, the server needs to calculate the importance of terms from the query to every document in its database, which is done using TF-IDF to estimate the importance of a specific term in a document. Secondly, the server needs to sort the documents by the statistic calculated in the first step in the order of decreasing importance. As a result, the server returns the n first most relevant documents to the client.

To try to optimize this process they have come up with the proposal to also include an SSE-based pre-processing step which would consist in first retrieving the documents which contain the query terms and, only after, passing them to the FHE server for ranking. To isolate different parts of the system they proposed an architecture model where the SSE server and the FHE server do not communicate: If the indices are disclosed, then the FHE server can compute the TF-IDF only for a small subset of the database records and then perform the sorting only on that subset. This is the most favorable case in terms of decreasing the FHE computation footprint, but it induces some information leakage – for example, it reveals if two queries concern the same (encrypted) records at the SSE step. The main drawback of this model is the leak of very important information while the main advantage is that the FHE Server could directly select the documents for which it will compute TF-IDF.

After the implementation, the following ideas were extracted: Upon receiving a request from the client, which contains encrypted terms and encrypted document indices obtained during the SSE phase, the FHE server performs the TF-IDF computation for all the documents in the database. However, it only retains and organizes the TF-IDF results corresponding to the number of document indices returned by the SSE phase. Consequently, the execution time gain derived from the use of SSE depends on the number of document indices returned.

After seeing the time execution performance, the authors then tried to enhance the ranking time through the implementation of a modified parallel merge sort for document ranking. Their approach entailed dividing the SSE response into smaller segments, sorting them in parallel, and retrieving only the first k elements from a response of length h, with $n > k$. However, the computational limitations of FHE precluded the extraction of just the first k elements during the merge phase, rendering the approach computationally intensive and negating any performance benefits.

As a result, utilizing Cingulata's [30] standard sorting function for sorting the entire response proved to be more efficient, albeit not to the extent required for practical application on large databases.

Presently, with regard to the challenge of searchable encryption, it can be concluded that while implementation is feasible, the current FHE schemes do not demonstrate sufficient computational efficiency to serve as a viable solution for real-world databases containing millions of records due to their excessive processing time. Despite this current limitation, the increasing interest in FHE bodes well for potential advancements in the near future.

3. Cloud Computing

Cloud computing refers to the practice of accessing IT resources like computing power, storage, and databases via the internet, paying only for what is used. Instead of owning and maintaining physical servers, these services are provided by cloud providers such as Amazon Web Services, Google Cloud Provider, Azure, etc.

With the widespread application of cloud computing, more and more sensitive information and private data are stored in the cloud by users.

Cloud storage security is one of the most crucial security issues in cloud computing. In order to protect the privacy of user data, cloud data should be stored in the form of ciphertext. However, encryption adds computational overhead, so the ideal scenario for which the researchers aim is that the confidentiality of the data will be guaranteed at the lowest possible cost.

Since cloud service providers are unreliable third parties, how to keep data confidential to cloud service providers and allow cloud service providers to complete various operations on data, it is difficult for traditional encryption methods to solve the above-mentioned problems. HE technology supports the management of ciphertext data under privacy protection. It can directly retrieve, calculate and count ciphertext in the cloud and return the results to users in the form of ciphertext without exposing the processed data to the cloud provider.

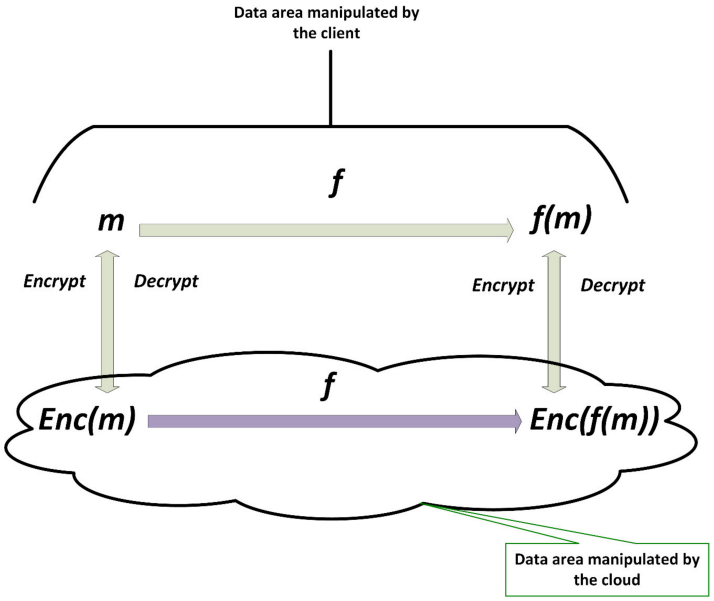

Fig. 1 - An overview of how would FHE work with cloud [33]

Compared to conventional encryption methods, HE reduces the cost of communication and computation by eliminating the need for frequent encryption and decryption between the cloud and users. Theoretically, HE is the key technology to ensure data confidentiality in cloud environments. By leveraging its homomorphic capabilities, HE resolves key security concerns in cloud services and drives further advancements in the technology through its application in cloud computing.

Until now, several FHE schemes have been proposed and implemented for several areas of cloud computing, such as Homomorphic Proxy Re-Encryption, Homomorphic Authenticated Encryption, and Multi-party Computation. However, some cloud service scenarios have requirements that make simple HE schemes unsuitable at the moment. Several schemes have undergone analysis and improvement with the aim of developing practical implementations, however, the work remains at a theoretical level due to various limitations, including inefficiencies in terms of time.

In 2022, a comprehensive overview of HE, FHE, and their applications was presented by Chiara Marcolla, Victor Sucasas, Marc Manzano, Riccardo Bassoli, Frank H.P. Fitzek, and Najwa Aaraj in [1]. The authors succinctly summarized the advancements made in several subareas of cloud

computing. Our analysis will be based on their work, supplemented with additional insights from the works of other researchers.

3.1. Homomorphic Proxy Re-Encryption

Homomorphic Proxy Re-Encryption (HPRE) is required in scenarios where a cloud service handles data from multiple users. Traditional Homomorphic Encryption (HE) only supports operations on ciphertexts encrypted with the same public key, so ciphertexts from different users must be converted to a common key.

Proxy Re-Encryption (PRE) is a widely used technique in cloud computing for traditional (non-homomorphic) encryption. PRE enables a proxy to transform the ciphertext from one user (the delegator) into one for a different user (the delegatee), without accessing the plaintext or the users' secret keys. This process allows the delegatee to decrypt the ciphertext without knowing the delegator's secret key. In Homomorphic Proxy Re-Encryption (HPRE), this capability is further enhanced by allowing the cloud service to perform homomorphic operations on the transformed ciphertexts.

Fortunately, Gentry's thesis proposes a simple construction to achieve HPRE [2]. The delegator generates two ciphertexts: one which encrypts the secret key with the delegatee's public key homomorphically; and one which encrypts the data with its own public key. Then, the proxy can evaluate the decryption circuit of the homomorphic scheme to re-encrypt the ciphertext with the delegatee's public key (this technique is identical to bootstrapping).

Unfortunately, Gentry's method has been shown to be vulnerable to weak collusion attacks where the delegatee and proxy could work together to obtain the delegator's secret key. Recent works using key-switching techniques have proposed HPRE schemes that can resist collusion. For example, Y. Kawai, T. Matsuda, T. Hirano, Y. Koseki, and G. Hanaoka, have come up with the proposal of a single-hop (only one re-encryption is allowed) HPRE scheme [15] that is partially homomorphic, meaning that only certain homomorphic operations are possible after converting the ciphertext.

Other works cater to fully homomorphic single-hop proxy re-encryption schemes, such as the work of C. Ma, J. Li, and W. Ouyang in [11]. Y. Polyakov, K. Rohloff, G. Sahu, and V. Vaikuntanathan provided two IND-CPA secure constructions for multi-hop HPRE for BV and NTRU schemes [12], which notably outperform previous lattice-based proxy re-encryption schemes based on NTRU and BV respectively. Also, another important contribution is the one of Z. Li, C. Ma, and D. Wang, who have provided the model for the construction of multi-hop HPRE schemes that are fully homomorphic via branching programs in [16].

All these works provide solutions to solve the collusion attack that Gentry's approach presents, however, they are still not resilient to strong collusion attacks. Namely, the proxy and the delegatee cannot obtain the delegator's secret key but they can still obtain some information about the delegator's secret key as proven by J. Li, Z. Qiao, K. Zhang, and C. Cui, in their research regarding strong anti-collusion for lattice-based HPRE in [17]. Moreover, as stated and demonstrated in [18], known HPRE schemes are only CPA secure, which is inadequate in some scenarios, as seen in Cohen's work [19].

Although it is well known that HE schemes cannot achieve CCA2 security (according to its standard definition), some can be CCA1-secure, such as the ones proposed by R. Canetti, S. Raghuraman, S. Richelson, and V. Vaikuntanathan. Unfortunately, that is not true for HPRE, as all currently known CCA1-secure HPRE schemes are only partially homomorphic.

3.2. Homomorphic Authenticated Encryption

In some scenarios, privacy alone is not enough. The user may pay for a specific service or use remote data processing in safety-critical applications, where ensuring the integrity of the data is important. Thus, a guarantee of correct processing by the cloud service may be necessary, even if it is not malicious, as it may try to avoid the heavy computational burden of processing homomorphically encrypted data by submitting incorrect data.

In these certain cases, the user should be able to verify that the decrypted data is the result of a specific arithmetic circuit over the transmitted encrypted data. This specific feature can be achieved with Homomorphic Authenticated Encryption (HAE). Homomorphic authenticated encryption[24] allows implicit computation on plaintexts using corresponding ciphertexts without losing data privacy and provides the authenticity of the computation and the resultant plaintext of the computation when performing decryption on a ciphertext. HAE can be obtained by combining HE and HA, obtaining homomorphic authenticated encryption [24] (Noteworthy, combining HE and HA results in the property that if both HE and HA are CPA secure, the resulting HAE scheme is also CCA1 secure [23]).

The process goes as follows in FHAE's case: The user transmits ciphertexts along with homomorphic signatures (HS) as authenticators. The HS can be evaluated homomorphically in a similar manner to ciphertexts, generating a valid signature for the processed data.

HS were initially proposed for linear arithmetic circuits. Scenarios such as secure random linear network coding (RLNC) adopted HS [25] to counteract tag pollution attacks. Subsequent works provided HS schemes that accept polynomial homomorphic operations [26]. Gorbunov, Vaikuntanathan, and Wichs [13] provided constructions with the desirable feature of being fully homomorphic, hence enabling fully homomorphic authenticated encryption (FHAE).

However, previous HS schemes (except for [13]) are selectively secure (i.e. the attacker is only provided with signatures of chosen messages before the challenge is available). Adaptive security was first achieved by Boyen, Fan, and Shi [14] and researched further. Unfortunately, previous HS constructions have a limitation in terms of efficiency for circuits of polynomial depth [14].

Another solution, potentially more efficient, is the adoption of verifiable computation (VC) schemes that work over encrypted data [**?**]. A VC scheme provides proof that each arithmetic gate of the arithmetic circuit has had its inputs processed. Moreover, it is even possible to provide such a proof of computation over a partially private circuit (known only to the cloud service), since the cloud service could prove that part in zero-knowledge.

3.3. Multi-party Computation

Secure Multiparty Computation (MPC) is a cryptographic technique that allows multiple parties to jointly compute a function over their inputs while keeping the inputs private. In MPC, each party inputs their private data, and the computation is performed in a way that ensures the confidentiality of the data and the validity of the result. The computation is performed in a secure environment, such as a trusted third party or a cryptographic protocol, that ensures that the result is accurate and that the private data of the parties is protected from unauthorized access or manipulation.

An FHE scheme allows the evaluation of arbitrary computations on encrypted data without decrypting it. In theory, realizing MPC through a HE scheme is a simple and efficient approach. However, despite its promising theoretical power, the practical side of the approach remains underdeveloped.

HE provides a solution for the centralization of private computations. However, in a scenario where several parties aim to interact, the direct application of HE is not so intuitive. Specifically, several cloud services may want to evaluate a function combining their private datasets without leaking any information about the inputs (except for what can be inferred from the output). Such a scenario can be addressed with secure multi-party computation (MPC).

There are different kinds of MPC protocols optimized for arithmetic circuits, based on secret-sharing techniques, and for boolean circuits, based on garbled circuits, respectively. Notably, some of these protocols follow a pre-processing model where the computation is divided into two phases. The first phase happens before the parties' inputs are defined, and consists of the generation of cryptographic resources (secret-shared elements or garbled circuits), which will be used to enhance the performance of the second phase. During the second phase, the participants define their inputs and perform the computation securely by evaluating the circuit privately.

Although less efficient than previous approaches that use HE (SHE-BMR, Overdrive), MPC

can also be constructed directly with Multi-Key FHE (MKFHE).

The work in [20] proposes a construction that requires only two communication rounds, see Fig. 16. In the first round, each party encrypts its inputs under multiple keys and broadcasts the ciphertexts to all parties. Then, each party evaluates the circuit homomorphically. Finally, in the second round, each party partially decrypts the result and broadcasts its share of the output. All shares can be combined locally by each party to obtain the output.

In MKFHE, decryption of a ciphertext is only possible with all partial decryptions from the secret key holders, ensuring input privacy. However, if a single party fails to provide its share of the output, the entire Secure Multiparty Computation (MPC) protocol would fail.

The issue of single-party failure for MKFHE was further addressed with a three-round MPC protocol that uses Threshold Multi-Key Fully Homomorphic Encryption (TMFHE). With TMFHE, a smaller group of parties can decrypt a ciphertext, enabling the reconstruction of the output by any subset of sufficient size after the first round of input sharing. This makes the protocol more resilient to failures, but it also requires trust as smaller groups can access the private inputs of the parties.

It's noteworthy that MKFHE suffices for providing MPC with passive security, i.e. malicious parties follow the protocol specification and try to extract information from the transmitted messages. But in a setting where parties can deviate from the protocol specification (e.g. transmit wrong shares of the output), active security is needed and can be achieved by integrating zero-knowledge proofs.

The field of MKFHE has been the subject of increased research and study in recent years, with several proposals based on the learning with errors (LWE) paradigm leading the way. However, the difficulty of the implementations and the inefficiency in terms of time have kept the research in a theoretical phase. Nevertheless, significant progress has been made and continues to be made, bringing the realization of FHE for cloud computing closer to practical reality.

4. Machine Learning

Machine learning is an area of artificial intelligence concerned with the creation of algorithms and statistical models that allow computers to perform tasks without explicit programming. The models are trained using data and mathematical algorithms to enable the systems to make informed predictions or decisions based on previous experiences and identified patterns. The ultimate goal of machine learning is to improve the accuracy of these predictions or decisions through experience.

The potential conflict between machine learning and privacy arises due to the nature of machine learning: Machine learning operates by analyzing data to identify patterns and processes, leading to concerns about machines potentially acquiring excessive knowledge or becoming too dominant. Despite these apprehensions, machine learning has already become ubiquitous in various applications, including but not limited to regression analysis, classification, recommendation systems, clustering, and anomaly detection. Machine learning is heavily used in various technologies such as driving aids, language translation tools, and facial recognition software.

One real-life example of this is the Cambridge Analytica scandal, in which a data analytics firm used personal data from millions of Facebook users to create targeted political advertisements. This data was obtained through a combination of scraping and obtaining data from users who installed a personality quiz app. Although the data was collected for ML purposes, it was not properly secured and was used for unethical purposes. Another example is the Equifax data breach, in which the personal information of over 147 million people, including Social Security numbers, birth dates, and addresses, was stolen. This data could be used for malicious purposes such as identity theft or financial fraud. In both of these cases, the data was collected for ML purposes, but the security of the data was not properly considered. This highlights the importance of not only using FHE, but also implementing additional security measures such as secure data storage, transmission, and access control to protect sensitive data.

It is crucial to consider the security of the data when collecting it for ML purposes, even when using FHE. The collection, storage, and usage of sensitive data must be done in a responsible and

secure manner to prevent potential harm to individuals and organizations. It is evident that the nature of data processed in machine learning is frequently highly sensitive or personal. The use of such data through machine learning can range from comparatively innocuous purposes, such as targeted advertising, to malicious intents, such as blackmail. However, this does not need to be the case.

The distinctiveness of FHE lies in its ability to perform computations on encrypted data without the need for decryption. Unlike conventional encryption algorithms that only secure data during transmission or storage, FHE allows for computation to be performed on encrypted data. This unique property makes FHE particularly well-suited for use in machine learning applications[1].

One barrier to the widespread utilization of machine learning though is the requirement for substantial data inputs to attain substantial accuracy, resulting in potential privacy and security issues. FHE enables the computation of arithmetic operations on encrypted real numbers, thus enabling the development of machine learning training algorithms that preserve privacy. FHE is highly important in the context of distributed machine learning as it supports secure confidential computing scenarios.

4.1. Privacy-Preserving Machine Learning With FHE for Deep Neural Network

In addition to ongoing research efforts, various commercial products have been proposed to address real-world problems across industries.[10]

Zama's[31] open-source technology allows for the inference of trained machine learning models on encrypted user data through the use of HE, regardless of the architecture or training method. The technology's potential applications include medical data, image classification, autonomous systems, and smart cities' data processing. Intel and Ant Group have collaborated to construct Privacy-Preserving Machine Learning on Intel's Software Guard Extensions and Occlum, utilizing cryptographic techniques such as HE and differential privacy. Duality Technologies, a company that provides privacy-preserving data collaboration platforms using HE, was selected by the Defense Advanced Research Projects Agency (DARPA) along with other top research institutes to further the application of FHE as part of DARPA's Data Protection in Virtual Environments program, which aims to develop hardware acceleration for FHE computations.

The CKKS[31] scheme is an FHE system that enables arithmetic operations on encrypted data with real or complex numbers. The encrypted data is in the form of a one-dimensional vector with each component called a slot. Public key users can process encrypted real or complex numbers with the CKKS scheme without accessing any private information. The security of CKKS is based on the ring-LWE assumption. Supported homomorphic operations include addition, scalar, and non-scalar multiplication, rotation, and complex conjugation. Each operation, except rotation, is applied component-wise. Scalar multiplication is with plaintext, while non-scalar multiplication is with ciphertext. The rotation operation performs a cyclic shift of the vector in multiple steps homomorphically. Non-scalar multiplication, rotation, and complex conjugation require additional evaluation keys and key-switching procedures. The data is scaled with a large integer called the scaling factor and rounded to an integer before encryption. Homomorphic multiplication of two CKKS encrypted data sets also multiplies their scaling factors, which need to be reduced to their original value through rescaling.

To employ HE, various non-arithmetic operations of ResNets and bootstrapping must be approximated through high-degree polynomials. The efficiency of polynomial evaluation for encrypted inputs using the RNS-CKKS scheme lies in minimizing non-scalar multiplications and reducing depth consumption. The baby-step giant-step polynomial evaluation method is a popular, efficient technique for polynomial evaluation that cuts down on non-scalar multiplications and depth consumption.

The rescaling operation reduces the scaling factor and ciphertext modulus, both necessary for each homomorphic multiplication. After repeated multiplications, the ciphertext modulus can no longer be reduced. CKKS scheme's bootstrapping transforms a ciphertext with a small modulus into a new ciphertext with a large modulus, preserving the message. Bootstrapping allows for constructing any arithmetic circuit with a high multiplicative depth.

The CKKS bootstrapping process begins with an increase in the ciphertext modulus, resulting in the message polynomial becoming

$m + q_0 I$, where:

- q_0 is the modulus before bootstrapping
- I is an unknown integer polynomial

Homomorphic modular reduction must be performed to remove the $q_0 I$ component from the coefficients of the message polynomial.

The implementation of homomorphic encryption (HE) and privacy-preserving machine learning (PPML) is a technically complex process that requires a combination of expertise in cryptography and machine learning. This complexity, along with the associated computational overhead and performance trade-offs, presents challenges for widespread adoption. Additionally, the lack of standardization in the field of HE and PPML, combined with the requirement for compliance with data privacy regulations such as GDPR and HIPAA, presents further obstacles to the widespread implementation of the RNS-CKKS scheme for PPML in deep neural networks.

5. IoT Computing

Fog computing, also known as edge computing, was introduced by CISCO to handle large-scale IoT deployments. It serves as a layer between IoT devices and cloud services, placed as close as possible to the device to pre-process data. This reduces both bandwidth use and latency in IoT applications. Tasks performed in fog computing are limited to processing information in a single data packet, such as evaluating relevance/category, formatting, encoding, filtering, etc.

The Smart City scenario exemplifies the usefulness of fog computing and the significance of HE. Intelligent transportation, resource management, safety and security, and environmental monitoring are a few applications in Smart Cities that rely on large-scale IoT deployments consisting of small sensors sending data to smart city data collectors. The fog layer processes the data in intermediate gateways, crucial for maintaining scalability.

The constant evolution and indispensability of IoT devices which are currently widely used: smartphones, smartwatches, and household appliances, have enabled the acquisition of diverse sensor data, including action history and heart rate information. This data is expected to be analyzed and utilized through cloud services. However, due to the existence of sensitive information in the sensor data collected by IoT devices, there is a need to protect personal information to prevent unauthorized access and information breaches in cloud services that may not be fully secure.

FHE can be implemented to secure citizens' privacy by encrypting the data from the sensors with the data collector's public key, allowing pre-processing operations in the fog layer to be performed on encrypted data. Only the data collector can access the decrypted data with its secret key. This approach not only protects data privacy in fog nodes but also ensures IoT nodes' privacy towards the data collector as the data is aggregated.

Although HE and FHE have potential, their complexity and large ciphertext size make them a challenge for IoT implementation. The ciphertext expansion leads to increased communication overhead, and the computational demands result in delays and strain on the limited resources of IoT devices and communication standards.

To overcome these issues, hybrid protocols combining HE with symmetric key encryption (SKE) have been researched [1], including hybrid homomorphic encryption (HHE). In HHE, IoT devices use an SKE scheme with a random key to encrypt data, then encrypt the key with a HE scheme using the data collector's public key. SKE is less complex and doesn't cause ciphertext expansion. Fog nodes can homomorphically evaluate the SKE decryption circuit and convert the encrypted data to HE-encrypted data, which can then be processed and sent to the data collector.

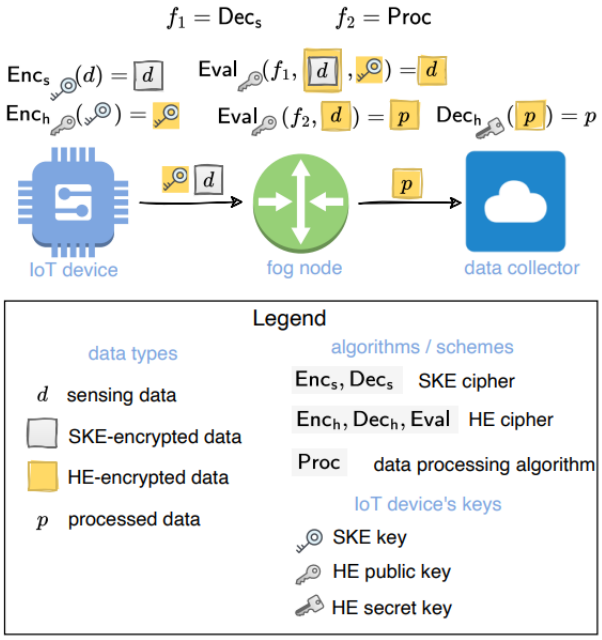

Fig. 2 - Hybrid FHE scheme applied to fog computing for IoT [1]

Aside from regular HE, research has also been conducted regarding the practical applications of FHE for IoT Computing, but implementing FHE on IoT devices with limited computing power remains a challenge due to the intensive data processing requirements and large size of the ciphertext, leading to increased communication overhead.

In 2021, in order to reduce the high computational load imposed by FHE on IoT devices with limited computing power, two techniques were proposed by Marin Matsumoto and Masato Oguchi in their work [22].

The first approach analyzed as a proposal for their implementation, referred to as SHE+FHE, involves encrypting plaintext using a lighter encryption system (SHE) on the IoT device and converting the SHE ciphertext to FHE ciphertext on the cloud service. Compared to FHE, SHE is notably faster and the generated ciphertext much shorter (as a comparison, on Google Pixel 3, FHE has a 0.609s encryption time and a 21.993MB ciphertext, while SHE has a 0.004s encryption time and its ciphertext 0.004MB).

After further analysis of the possible options, the authors opted to also use TRIVIUM in their implementation, coming up with a complex scheme called SHE+TRIVIUM+FHE. Using the combination of SHE, TRIVIUM, and FHE provides a benefit in that the IoT device is not forced to employ FHE. Additionally, the plaintext encryption utilizing a common key cryptosystem leads to a decrease in the load on the IoT device compared to that of the SHE and FHE integration. However, this design is more complex and imposes a heavier load on the cloud service.

Experiments were conducted using a smartphone as an IoT device to compare the proposed techniques with existing techniques. The results indicated that the proposed techniques significantly reduced the load on the IoT device, with SHE+TRIVIUM+FHE being the most efficient in terms of balancing the load on the IoT device and cloud service.

Showing great promise in the progress made in reducing the computational overhead, HE is proving to be a strong contender in the area of IoT computing too, although further research is required to improve the conversion process to FHE and to evaluate the techniques using high-performance computers instead of laptops on the cloud service side.

6. E-voting

Electronic voting (e-voting) refers to the use of electronic devices and systems to cast and count votes in an election. This technology provides an alternative to traditional paper-based voting systems and offers several benefits, such as increased accuracy and speed in vote counting, reduced costs, and increased accessibility for people with disabilities. However, e-voting systems also raise concerns about the security and transparency of the voting process, and there is an ongoing debate about the use of electronic voting systems in elections. To ensure the integrity and fairness of the voting process, e-voting systems must be designed and implemented with robust security measures in place to prevent hacking, tampering, and other forms of manipulation.

FHE has the potential to revolutionize the field of e-voting by enabling the secure and private casting and counting of votes without the need to decrypt the data. With FHE, encrypted votes can be processed and counted while remaining encrypted, providing a high level of security and privacy for voters. This is particularly important for e-voting systems, where there are concerns about the potential for hacking and manipulation. By using FHE, voters can cast their votes with confidence, knowing that their privacy and the integrity of the voting process are protected.

Unfortunately, the use of FHE for e-voting remains just a concept at present. However, there have been experimental implementations of e-voting systems based on partially homomorphic encryption schemes, such as the Paillier cryptosystem [21], which have demonstrated a great deal of promise and, hopefully, there will be new implementations using the full potential of arbitrary computations specific to FHE proposed in the near future.

7. Conclusion

In the present study, we provide a comprehensive overview of the currently researched applications of Fully Homomorphic Encryption (FHE) in various domains, including Databases, Cloud Computing, Machine Learning, IoT Computing, and E-voting. At this time, its applications in these areas are not yet mature enough for practical deployment. However, the high level of confidentiality and privacy offered by FHE has stimulated extensive research and highly promising optimization efforts, thus advancing its prospects for practical deployment.

8. Acknowledgements

First and foremost, we would like to mention the tremendous help we got from the well-put-together work [1] of Chiara Marcolla, Victor Sucasas, Marc Manzano, Riccardo Bassoli, Frank H.P. Fitzek, and Najwa Aaraj, in regards to FHE and its applications, from which we were able to get a clear view on what we should search for and read about in order to get a clear picture on the current status of FHE.

Last, but not least, we would like to thank our Master's Programme Coordinator, Professor Ferucio Laurențiu Țiplea, Ph.D., for providing us with an extensive list of research papers on the subject which helped us understand the formalized implementations of FHE in regard to Cloud Computing and Private Query Processing.

References

[1] Chiara Marcolla, Victor Sucasas - Member, IEEE, Marc Manzano, Riccardo Bassoli - Member, IEEE, Frank H.P. Fitzek - Senior Member, IEEE, and Najwa Aaraj, "Survey on Fully Homomorphic Encryption, Theory, and Applications", *Proceedings of the IEEE*, 2022, vol. 110, no. 10, pp. 1572-1609. [Online] Available: https://eprint.iacr.org/2022/1602.pdf

[2] Craig Gentry, "A fully homomorphic encryption scheme", *Ph.D. thesis, Stanford University*, 2009. [Online] Available: https://crypto.stanford.edu/craig/craig-thesis.pdf

[3] Rémi Leluc, Elie Chedemail, Adéchola Kouande, Quyen Nguyen, Njaka Andriamandratomanana, "Fully Homomorphic Encryption and Bootstrapping", *Research Report*, IRMAR - Université Rennes 1. 2022, pp.1-12. hal-03676650. [Online] Available: https://hal.science/hal-03676650/document

[4] Youssef Gahi, Mouhcine Guennoun, and Khalil El-Khatib, "SA Secure Database System using Homomorphic Encryption Schemes", 2011. [Online] Available: https://arxiv.org/ftp/arxiv/papers/1512/1512.03498.pdf

[5] Jung Hee Cheon, Miran Kim, and Myungsun Kim, "Search-and-compute on Encrypted Data", *Financial Cryptography and Data Security*, 2015, pp. 142–159. [Online] Available: https://eprint.iacr.org/2014/812.pdf

[6] Myungsun Kim, Hyung Tae Lee, San Ling, and Huaxiong Wang, "On the Efficiency of FHE-based Private Queries", *EEE Transactions on Dependable and Secure Computing*, 2015, vol. 15, no. 2, pp. 357-363. [Online] Available: https://eprint.iacr.org/2015/1176.pdf

[7] Yassine Abbar, Pascal Aubry, Thierno Barry, Sergiu Carpov, Sayanta Mallick, Mariem Krichen, Damien Ligier, Sergey Shpak, and Renaud Sirde, "Cloud-based Private Querying of Databases by Means of Homomorphic Encryption", *Proceedings of the 6th International Conference on Internet of Things, Big Data and Security - Volume 1: IoTBDS*, 2021, pp. 123-131. [Online] Available: https://www.scitepress.org/Papers/2021/103788/103788.pdf

[8] Min Zhao E and Yang Geng, "Homomorphic Encryption Technology for Cloud Computing", *Procedia Computer Science*, 2019, vol. 154, pp. 73-83. [Online] Available: https://www.sciencedirect.com/science/article/pii/S1877050919307811

[9] Jack Aiston, "Homomorphic Encryption in Algebraic Settings", *Ph.D. thesis, School of Mathematics & Statistics, Newcastle University*, 2019. [Online] Available: https://theses.ncl.ac.uk/jspui/bitstream/10443/5041/1/Aiston J 2020.pdf

[10] Joon-Woo Lee, Hyungchul Kang, Yongwoo Lee, Woosuk Choi, Jieun Eom, Maxim Deryabin, Eunsang Lee, Junghyun Lee, Donghoon Yoo, Young-Sik Kim and Jong-Seon*Privacy-Preserving Machine Learning With Fully Homomorphic Encryption for Deep Neural Network*

[11] C. Ma, J. Li, and W. Ouyang, "A homomorphic proxy re-encryption from lattices", *International Conference on Provable Security*, Springer, pp. 353–372, 2016. [Online] Available: https://www.researchgate.net/publication/309029227_A_Homomorphic_Proxy_Re-encryption_from_Lattices

[12] Jack Aiston, "Homomorphic Encryption in Algebraic Settings", *Ph.D. thesis, School of Mathematics & Statistics, Newcastle University*, 2019. [Online] Available: https://theses.ncl.ac.uk/jspui/bitstream/10443/5041/1/AistonJ2020.pdf

[13] Sergey Gorbunov, Vinod Vaikuntanathan, and Daniel Wichs, "Leveled fully homomorphic signatures from standard lattices", *Proceedings of the Forty-Seventh Annual ACM Symposium on Theory of Computing*, pp. 469-477, 2015. [Online] Available: https://eprint.iacr.org/2014/897.pdf

[14] Xavier Boyen, Xiong Fan, and Elaine Shi, "Adaptively Secure Fully Homomorphic Signatures Based on Lattices", *Cryptology ePrint Archive*, 2014. [Online] Available: https://eprint.iacr.org/2014/916.pdf

[15] Y. Kawai, T. Matsuda, T. Hirano, Y. Koseki, and G. Hanaoka, "Proxy Re-Encryption That Supports Homomorphic Operations for Re-Encrypted Ciphertexts", *IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences*, vol. 102, no. 1, pp. 81–98, 2019.

[16] Zengpeng Li, Chunguang Ma, and Ding Wang, "Towards multi-hop homomorphic identity-based proxy re-encryption via branching program", *IEEE Access*, vol. 5, pp. 16 214–16 228, 2017. [Online] Available: https://www.researchgate.net/publication/319168677_Towards_Multi-Hop_Homomorphic_Identity-Based_Proxy_Re-Encryption_via_Branching_Program

[17] J. Li, Z. Qiao, K. Zhang, and C. Cui, "A Lattice-Based Homomorphic Proxy Re-Encryption Scheme with Strong Anti-Collusion for Cloud Computing", *Sensors*, vol. 21, no. 1, 2021. [Online] Available: https://doi.org/10.3390/s21010288

[18] F. Luo, S. Al-Kuwari, W. Susilo, and D. H. Duong, "Chosen-ciphertext secure homomorphic proxy re-encryption", *IEEE Transactions on Cloud Computing*, pp. 1–1, 2020.

[19] Aloni Cohen, "What about Bob? The Inadequacy of CPA Security for Proxy Reencryption", *Public-Key Cryptography – PKC 2019*, 2019. [Online] Available: https://eprint.iacr.org/2017/785.pdf

[20] Pratyay Mukherjee, and Daniel Wichs, "Two Round Multiparty Computation via Multi-key FHE", *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, 2016. [Online] Available: https://www.researchgate.net/publication/301670743_Two_Round_Multiparty_Computation_via_Multi-key_FHE

[21] Saksham Saproo, Vedant Warke, Shreyas Pote, Rashmi A. Dhumal, "Online Voting System using Homomorphic Encryption", *ITM Web of Conferences*, 2020. [Online] Available: https://www.itm-conferences.org/articles/itmconf/pdf/2020/02/itmconf_icacc2020_03023.pdf

[22] Marin Matsumoto, and Masato Oguchi, "Speeding Up Encryption on IoT Devices Using Homomorphic Encryption", *IEEE International Conference on Smart Computing (SMARTCOMP)*, 2021. [Online] Available: http://www.is.ocha.ac.jp/ oguchi_lab/Publications/paper2021/BITS2021_marin.pdf

[23] Chihong Joo, and Aaram Yun, "Homomorphic Authenticated Encryption Secure Against Chosen-Ciphertext Attack", *Advances in Cryptology – ASI - ACRYPT*, 2014. [Online] Available: https://eprint.iacr.org/2013/726.pdf

[24] Jeongsu Kim, and Aaram Yun, "Secure Fully Homomorphic Authenticated Encryption", *IEEE Access*, vol. 9, pp. 107 279–107 297, 2021. [Online] Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9500126

[25] Nuttapong Attrapadung, and Benoît Libert, "Homomorphic Network Coding Signatures in the Standard Model", *Public Key Cryptography – PKC 2011*, pp. 17-34, 2011. [Online] Available: https://link.springer.com/chapter/10.1007/978-3-642-19379-8_2

[26] Dario Catalano, Dario Fiore, and Bogdan Warinschi, "Homomorphic Signatures with Efficient Verification for Polynomial Functions", *Advances in Cryptology – CRYPTO 2014*, pp. 371-389, 2014. [Online] Available: https://link.springer.com/chapter/10.1007/978-3-662-44371-2_21

[27] CryptDB, [Online] Available: https://github.com/CryptDB/cryptdb

[28] Monomi, [Online] Available: https://github.com/stephentu/monomi-optimizer

[29] OpenSSE, [Online] Available: https://opensse.github.io/

[30] Cingulata, [Online] Available: https://github.com/CEA-LIST/Cingulata

[31] Zama, [Online] Available: https://zama.ai.

[32] Image - FHE General, [Online] Available: https://galois.com/blog/2017/12/revolution-evolution-fully-homomorphic-encryption/

[33] Image - FHE Cloud, [Online] Available: https://www.mdpi.com/technologies/technologies-07-00021/article_deploy/html/images/technologies-07-00021-g001.png