Science and Technology Publishing (SCI & TECH)
ISSN: 2632-1017
Vol. 6 Issue 4, April - 2022

A Survey on Fully Homomorphic Encryption and Its Applications

Jollanda Shara
Dept.Mathematics&Computer Science
University "Eqrem Cabej"
Gjirokaster, Albania
e-mail jokrisha@yahoo.com

***Abstract*— Fully homomorphic encryption (FHE) has been considered as the "holy grail" of cryptography for its adaptability as a cryptographic primitive and wide range of potential applications. It opens the door to many new capabilities with the goal to solve the IT world's problems of security and trust. FHE is a new but quickly developing technology. FHE is a cryptographic primitive that allows one to compute arbitrary functions over encrypted data. Since 2009, when Craig Gentry showed that FHE can be realized in principle, there has been a lot of new discoveries and inventions in this particular field and substantial progress has been made in finding more practical and more efficient schemes, as well. Such schemes have numerous applications since it allows users to encrypt their private data locally but still outsource the computation of the encrypted data without risking exposing the actual data. The new schemes significantly reduce the computational cost of FHE and make practical deployment within reach. However, FHE is made possible with many new problems and assumptions that are not yet well studied.**

Keywords—encryption; application; FHE; scheme.

I. INTRODUCTION

The purpose of homomorphic encryption is to allow computation on encrypted data. Thus data can remain confidential while it is processed. This enables useful tasks to be accomplished with data residing in untrusted environments. In a world of distributed computation and heterogeneous networking this is a hugely valuable capability. One of the goal in cryptography had been the problem of finding a general method for computing on encrypted data. It was proposed, first, by Rivest, Adleman and Dertouzos in 1978. [8] In encryption schemes, Bob encrypts a plaintext message to obtain a ciphertext. Alice decrypts the ciphertext to recover the plaintext. In Fully Homomorphic Encryption, parties that do not know the plaintext data can perform computations on it by performing computations on the corresponding ciphertexts.[3]

The development of fully homomorphic encryption is a revolutionary advance. It extends the scope of the computations which can be applied to process encrypted data homomorphically. Fully homomorphic encryption (FHE) allows evaluation of arbitrary functions on encrypted data, and as such has a multitude of potential applications such as private cloud computing. Gentry [5, 6] was the first to show that FHE is theoretically possible. His construction consisted of three parts: first, construct an encryption scheme that is somewhat homomorphic, i.e. that can evaluate functions of limited complexity (think low degree), secondly, simplify the decryption function of this scheme as much as possible (so called squashing), thirdly, evaluate this simplified decryption function homomorphically to obtain ciphertexts with a fixed inherent noise size (so called bootstrapping).[4]

This has become a very interesting topic of study because of its numerous applications in the real world. Since Gentry published his idea in 2009 [5, 6] there has been enormous interest in the area, for improving the schemes, implementing them and applying them.

As an example, let us consider cloud computing, which we mentioned above. As more and more data is utilized into cloud storage, often unencrypted, considerable trust is required in the cloud storage providers. The Cloud Security Alliance lists data breach as the top threat to cloud security. Encrypting the data with conventional encryption avoids the problem. However, now the user cannot operate on the data and must download the data to perform the computations locally. With fully homomorphic encryption the cloud can perform computations in the interests of the user and return only the encrypted result.

II. SOME HISTORICAL NOTES

The notion of encryption schemes that permit nontrivial computation on encrypted data was first proposed by Rivest, Adleman and Dertouzos [8] in 1978, shortly after the invention of the RSA cryptosystem, in an especially provident paper titled "On Data Banks and Privacy Homomorphisms". Rivest et al. [8] proposed the exponentiation function and the RSA function as additive and multiplicative "privacy homomorphisms", respectively. Note,

however, that neither of these functions by themselves provide even chosen plaintext security. [7] Their paper states, although there are some truly essential limitations on what can be accomplished,"… we shall see that it appears likely that there exist encryption functions which permit encrypted data to be operated on without preliminary decryption of the operands, for many sets of interesting operations. These special encryption functions we call "privacy homomorphisms"; they form an interesting subset of arbitrary encryption schemes". Despite the optimism of Rivest, Adleman, and Dertouzos, fully homomorphic encryption remained unapproachable for many years.[3]

The El Gamal encryption scheme, derived from the exponentiation function, is multiplicatively homomorphic and CPA-secure. However, methods of turning the RSA function into a CPA-secure encryption scheme, either by the hardcore-bit construction [9] or the RSA-OAEP construction [10], seem to destroy its homomorphic properties. The first semantically secure homomorphic encryption scheme follows from the work of Goldwasser and Micali [9] that defined the first robust notion of security for encryption. The GM encryption scheme supports addition of encrypted bits mod 2 (that is, the exclusive OR function). A number of encryption systems that are either additively or multiplicatively homomorphic followed this example. This includes the El Gamal encryption scheme [11], the Paillier encryption scheme [12] and its generalization by Dämgard and Jurik [13], a host of lattice-based encryption schemes starting from the work of Ajtai and Dwork [14], [15], [16], and many others [17], [18], [19]. All these schemes doesn't support both homomorphic addition and multiplication of plaintexts. Constructing an encryption scheme that is both additively and multiplicatively homomorphic remained a tempting open question. Additively homomorphic encryption schemes are already quite useful in a number of applications. We mention here three such applications. Cohen and Fischer [17] proposed an additively homomorphic encryption scheme based on higher order residuosity, and showed how to use it to perform secure electronic voting. This proposal and its descendants have introduced the modern day web-based voting systems such as Helios [21]. Peikert and Waters [22] constructed lossy and all-but-one trapdoor functions from additively homomorphic encryption schemes (with some extra properties).They successively use them to construct chosen ciphertext secure (CCA-secure) public key encryption schemes. The third application is to Private Information Retrieval (PIR) protocols.

It took too much time the construction of encryption schemes that surpass simple additive (or multiplicative) homomorphisms. Boneh, Goh and Nissim [23] showed an encryption scheme based on bilinear pairings on elliptic curves that could perform arbitarily many additions as well as a single multiplication on plaintexts. Gentry, Halevi and Vaikuntanathan [24] later showed how to achieve the same ability using lattices. Another proposal for fully homomorphic encryption was that of Fellows and Koblitz [25] which is based on the hardness of the Ideal Membership Problem in the multivariate polynomial ring. An important application of homomorphic encryption derives from the work of Kushilevitz and Ostrovsky [26]. They showed how to construct (single-server) Private Information Retrieval (PIR) protocols with sub-linear communication, from any additively homomorphic encryption scheme.

Private Information Retrieval is closely connected to Fully Homomorphic Encryption. The big breakthrough came with the work of Gentry [6] in 2009. In this work he showed the first possible construction of a fully homomorphic encryption scheme that enables computation of arbitrary functions on encrypted data and producing compact ciphertexts, and not only. Gentry's work showed a general method (a "blue-print") to construct such systems, as well. This blue-print has been instantiated with a number of cryptographic assumptions, yielding progressively simpler and more efficient schemes ([27], [28], [29], [30]).

Although the blue-print was an elegant and general one, schemes constructed along these lines suffer from a number of deficiencies, including the dependence on a host of non-standard cryptographic assumptions, and severe limitations on efficiency. Gentry's construction has three components: a "somewhat homomorphic" encryption scheme that can evaluate a limited class of functions, a method of "bootstrapping", a sufficiently powerful homomorphic encryption scheme called a "bootstrappable" encryption scheme, into a fully homomorphic encryption scheme and finally, to bring it all together, a specialized method of turning the somewhat homomorphic scheme into a bootstrappable scheme.[7]

III. HE AND HE SCHEMES

We live in an era where the most people all over the world possess more than one digital device with limited local storage. Therefore, there is a growing need for services that let users easily store and access personal files. Data that was previously stored on paper is being converted to a digital file. Since most of these devices are not able to process data locally, they will often upload it to a third party for processing. However, this data may be private, the third party may not be trustworthy, or both. Therefore, the data should be encrypted before it is transferred.

Cryptography refers generally to secret writing based on the use of ciphers. A cipher is a technique used to keep out of sight information or expose it, with the help of which a plaintext is converted into a ciphertext and vice versa. Most ciphers depend on an algorithm and a key. While the algorithm establishes a way to transform data, the key is used as a parameter that modifies its behavior in a complex manner. Cryptosystems are usually classified in two categories: those where a common key is used for both ciphering and deciphering, based on private-key

cryptography, and those supported in public-key cryptography, where a public key is used to cipher messages, and the ciphertext has to be deciphered by applying the corresponding secret key.[31]

Encryption is a method used for encoding information with the goal to ensure confidentiality so that only authorised parties can access the information. There exists different types of encryption schemes that can be either symmetric or asymmetric. In the symmetric setting, the same key is used for encryption and decryption and it is commonly used when a secure channel is already established. In the asymmetric setting, there exists a public and private key for each party where the public key is used for encryption and the private key for decryption. The public key is shared between parties while the private key is kept secret so that only the holder of the secret key can decrypt the message encrypted under the corresponding public key. An analogy often used for homomorphic cryptosystems is the that of a jewellery shop. Alice owns a jewellery shop and has raw precious material that she wants her workers to get together into jewellery. The problem is that she doesn't trust her workers - she is afraid they will thieve the material if given the opportunity. She wants her workers to be able to process the materials without actually having access to them. In order to solve this problem, Alice designs a transparent, impenetrable glove box, puts the raw material inside, locks the box with a key that only she has access to and then gives the box to one of the workers. The worker can assemble the jewels inside the box using the gloves without being able to access the materials inside since it is impenetrable. Once the worker is finished, the worker gives the box back to Alice who can unlock it with her key and extract the jewellery. In this analogy, the data is represented by the material that needs to be processed and the encryption of that data is represented by the box. The special thing about this cryptosystem is that it has gloves allowing the data to be processed without accessing it. (see [2])

Homomorphic Encryption (HE) is a kind of encryption scheme that allows a third party (e.g., cloud, service provider) to perform certain computable functions on the encrypted data while preserving the features of the function and format of the encrypted data. The reason why it is called "homomorphic" is that, roughly speaking, there exists a correspondence between the space of the messages and the space of the ciphertexts, in such a way that operations performed on ciphertexts are somehow reflected in operations on the messages they encrypt. [20]

Indeed, this homomorphic encryption corresponds to a mapping in the abstract algebra. As an example for an additively HE scheme, for sample messages m_1 and m_2, one can obtain $E(m_1 + m_2)$ by using $E(m_1)$ and $E(m_2)$ without knowing m_1 and m_2 explicitly, where E denotes the encryption function. [32]

It is known that in traditional encryption schemes, Bob encrypts a plaintext message to obtain a cipher text. Alice decrypts the ciphertext to find the plaintext. So, with this standard encryption, one can only secure the following steps: establishment of communication and data transfer. In Homomorphic Encryption, parties that do not know anything about the plaintext data can perform computations on it by performing computations on the corresponding ciphertext.

An HE scheme is primarily characterized by four operations: $KeyGen, Enc, Dec$, and $Eval$. *KeyGen* is the operation that generates a secret and public key pair for the asymmetric version of HE or a single key for the symmetric version. Actually, $KeyGen, Enc$, and Dec are not different from their classical tasks in conventional encryption schemes. However, *Eval* is an HE-specific operation, which takes ciphertexts as input and outputs a ciphertext corresponding to a functioned plaintext. $Eval$ performs the function $f()$ over the ciphertexts (c_1, c_2) without seeing the messages (m_1, m_2). *Eval* takes ciphertexts as input and outputs evaluated ciphertexts. The most critical point in this homomorphic encryption is that the format of the ciphertexts, after an evaluation process, must be preserved in order to be decrypted correctly. Additionally, the size of the ciphertext should also be constant to support an unlimited number of operations. Otherwise, the increase in the ciphertext size will require more resources and this will limit the number of operations. Of all HE schemes in the literature, PHE schemes support the $Eval$ function for only either addition or multiplication, SWHE schemes support for only a limited number of operations or some limited circuits (e.g., branching programs), and FHE schemes support the evaluation of arbitrary functions (e.g., searching, sorting, max, min, etc.) for an unlimited number of times over ciphertexts. [32]

Partially Homomorphic Encryption (PHE):

A cryptosystem is partially homomorphic if it supports adding or multiplying of ciphertexts

but not both operations at the same time. The GoldwasserMicali [9] and Paillier [12] schemes supported addition operations, while the RSA [34] and ElGamal [11] schemes supported multiplication operations.

Additive Homomorphic Encryption (AHE):

A scheme is an additive Homomorphic Encryption (AHE) if giving only the public key and the encryption of m_1 and m_2, one can compute the encryption of $m_1 + m_2$.

Multiplicative Homomorphic Encryption (MHE):

A scheme is a Multiplicative Homomorphic Encryption (MHE) if giving only the public key and the encryption of m_1 and m_2, one can compute the encryption of $m_1 m_2$.

Asymmetric Homomorphic Encryption Algorithms

Paillier Scheme, an AHE:

Paillier cryptosystem is an asymmetric algorithm for public key cryptography and it is an AHE.The algorithm consists of three components: the key generator, the encryption algorithm and the decryption algorithm.

RSA Scheme, a MHE:

RSA is an asymmetric cryptosystem. It was described in 1977 by Ron Rivest, Adi Shamir and Leonard Adleman. The scheme summarizes in three steps: the

key generator, the encryption algorithm and the decryption algorithm.

ElGamal cryptosystem, a MHE:

The ElGamal encryption system is an asymmetric key encryption algorithm for public key Cryptography which is based on the DiffieHellman key exchange. It was described by Taher ElGamal in 1985.([33], [34])

IV. SOME APPLICATIONS OF HE

The main reason for the interest in homomorphic cryptosystems is that they have many theoretical as well as practical applications in different areas of cryptography. In the following, we give some of them.

1) Protection of mobile agents: One of the most interesting applications of homomorphic encryption is its use in protection of mobile agents. As we know, all conventional computer architectures are based on binary strings and only require multiplication and addition. So, such homomorphic cryptosystems would offer the possibility to encrypt a whole program so that it is still executable. Hence, it could be used to protect mobile agents against malicious hosts by encrypting them. The protection of mobile agents by homomorphic encryption can be used in two ways: (i) computing with encrypted functions and (ii) computing with encrypted data. Computation with encrypted functions is a special case of protection of mobile agents. Using homomorphic cryptosystems the encrypted function can be evaluated which guarantees its privacy.

2) Multiparty computation: In multiparty computation schemes, several parties are interested in computing a common, public function on their inputs while keeping their individual inputs private. This problem belongs to the area of computing with encrypted data. Moreover, in multi-party computation protocols, the function that should be computed is publicly known, while on the contrary in the area of computing with encrypted data it is a private input of one party.

3) Secret sharing scheme**:** In secret sharing schemes, parties share a secret so that no individual party can reconstruct the secret form the information available to it. However, if some parties cooperate with each other, they may be able to reconstruct the secret. In this framework, the homomorphic property implies that the composition of the shares of the secret is equivalent to the shares of the composition of the secrets.

4) Threshold schemes: Both secret sharing schemes and the multiparty computation schemes are examples of threshold schemes. Threshold schemes can be implemented using homomorphic encryption techniques.

5) Zero-knowledge proofs: This is a fundamental primitive of cryptographic protocols and serves as an example of a theoretical application of homomorphic cryptosystems. Zeroknowledge proofs are used to prove knowledge of some private information. For instance, consider the case where a user has to prove his identity to a host by logging in with her account and private password. Obviously, in such a protocol the user wants her private information (i.e., her password) to stay private and not to be leaked during the protocol operation. Zero-knowledge proofs guarantee that the protocol communicates exactly the knowledge that was expected, and no (zero) extra knowledge.

6) Election schemes: In election schemes, the homomorphic property provides a tool to obtain the tally given the encrypted votes without decrypting the individual votes.

7) Watermarking and fingerprinting schemes: Digital watermarking and fingerprinting schemes embed additional information into digital data. The homomorphic property is used to add a mark to previously encrypted data. In general, watermarks are used to identify the owner/seller of digital goods to ensure the copyright. In fingerprinting schemes, the person who buys the data should be identifiable by the merchant to ensure that data is not illegally redistributed.

8) Oblivious transfer: It is an interesting cryptographic primitive. Usually in a two-party 1-out-of-2 oblivious transfer protocol, the first party sends a bit to the second party in such a way that the second party receives it with probability $\frac{1}{2}$, without the first party knowing whether or not the second party received the bit.

9) Commitment schemes: Commitment schemes are some fundamental cryptographic primitives. In a commitment scheme, a player makes a commitment. She is able to choose a value from some set and commit to her choice such that she can no longer change her mind. She does not have to declare her choice although she may do so at some point later. Some commitment schemes can be efficiently implemented using homomorphic property.

10) Lottery protocols: Usually in a cryptographic lottery, a number pointing to the winning ticket has to be jointly and randomly chosen by all participants. Using a homomorphic encryption scheme this can be realized as follows: Each player chooses a random number which she encrypts. Then using the homomorphic property the encryption of the sum of the random values can be efficiently computed. The combination of this and a threshold decryption scheme leads to the desired functionality.

11) Mix-nets: Mix-nets are protocols that provide anonymity for senders by collecting encrypted messages from several users. For instance, one can consider mix-nets that collect ciphertexts and output the corresponding plaintexts in a randomly permuted order. In such a framework, privacy is achieved by requiring that the permutation that matches inputs to outputs is kept secret to anyone except the mix-net. In particular, determining a correct input/output pair, i.e., a ciphertext with corresponding plaintext, should not be more effective then guessing one at random. A desirable property to build such mix-nets is reencryption which is achieved by using homomorphic encryption.

12) Data aggregation in wireless sensor networks: In-network data aggregation in WSNs is a technique that combines partial results at the intermediate nodes en

route to the base station, as a result reducing the communication overhead and optimizing the bandwidth utilization in the wireless links. However, this technique raises privacy and security issues if the sensor nodes which need to share their data with the aggregator node. In applications such as healthcare and military surveillance where the sensitivity of private data of the sensor is very high, the aggregation has to be carried out in a privacy-preserving way, so that the sensitive data are not published to the aggregator. [35]

V.FHE

Gentry's FHE scheme is an asymmetric encryption scheme based on ideal lattices. Essentially one generates a secret key and then a number of public keys, each containing "noise", in a way that it is infeasible for an adversary to generate the secret key from the public keys. The problem with this first solution is that the "noise" in the ciphertext grew with each additional computation. This means that at a certain point the ciphertext will no longer decrypt to the original message because the "noise" has grown to large. Encryption schemes with this property are called Somewhat Homomorphic Encryption (SHE) schemes. The term somewhat stresses that the number of homomorphic operations one can perform is limited. In the same work [5] Gentry also provides a generic technique to transform SHE in FHE. Such technique is called Bootstrapping. Bootstrapping solves the problem of not being able to decrypt properly when the "noise" grows too large by homomorphically decrypting the ciphertext, performing a single computation on it, and then recrypting under a different public key. Unfortunately, the bootstrapping procedure is too theoretical and therefore not very efficient. Additionally, bootstrapping requires the non-standard assumption of circular security. It assumes that it is safe to encrypt the secret key under its own public key [36, 5, 37].

Since Gentry's first FHE scheme in 2009, several others have been developed [27, 28, 38, 39, 40]. In 2010, Smart and Vercauteren [28] made the first attempt of implementing FHE and since then a lot of work has been done towards more practical implementations [41, 42, 43]. FHE is only suitable in settings where the computations involve a single user. This for the reason that it requires the input from the users to be encrypted under the same key. Imagine instead a scenario where users, who have uploaded data to the cloud in encrypted form, wish to compute some joint function of their data encrypted under different keys. To handle these multi party situations, in 2012, Lopéz-Alt et al. [44] introduced a multi-key FHE scheme based on the NTRU cryptosystem [45]. This scenario is significantly more complex than the single user setting but even in this area a lot of recent improvements have been done , as well. [36, 47, 48]. Contrary to the single user setting, there exist no implementation of multi-key FHE to the best of the author's knowledge. A consequence of choosing non-trival schemes that have not already been implemented is that the difficulties are in general unpredictable. The challenges include studying and translating the scheme into code, choosing parameters, designing algorithms, choosing the benchmarks and running experiments on them. Furthermore, the schemes are based on some non-standard assumptions and Perlman and Brakerski state that their scheme is not practical. This makes implementing it a big challenge [36]:

"We stress that our scheme is not by itself practical. We use the bootstrapping machinery in a way that introduces fair amounts of overhead into the evaluation process. The goal of this work, rather, is to indicate that the theoretical boundaries of multi-key FHE, and open the door for further optimisations bringing solutions closer to the implementable world."

5.1. FHE Schemes

Let us consider now the concepts of FHE in the single and multi key setting as well as details of the implemented schemes.

Single-key Fully Homomorphic Encryption:

If we are satisfied with an encryption scheme that supports homomorphic operations of some specific operation one might question that homomorphic encryption seems easy to realise. However, some situations require more operations than only one type. If we again consider the case of cloud computing one typical reason for outsourcing computations is because it tend to be heavy and complex. This is obviously not achieved by only one specific operation which is why we need homomorphic encryption schemes that support arbitrary operations.

GSW:

Suppose that we have two matrices $\boldsymbol{C_1}$ and $\boldsymbol{C_2}$ that have the same eigenvector $\boldsymbol{s}$. Would it be possible to create an encryption scheme where $\boldsymbol{s}$ represents the secret key, $\boldsymbol{C}$ the ciphertext and m the message in form of an eigenvalue that corresponds to the eigenvector? If the eigenvector is kept secret, the message should be hidden and when we know the eigenvector it should possible to recover the message. Intuitively, the answer would be no as finding an eigenvalue can be solved by Gaussian elimination in polynomial time. However, Gentry, Sahai and Waters [49] took this idea further and instead of using eigenvectors as secret keys they used approximate eigenvectors. As long as the noise vector has a lower norm than the modulus one works with the ciphertext will be decryptable. By relaxing the condition, it becomes a hard problem to solve where the hardness derives from the LWE problem. Gentry, Sahai and Waters were able to create an encryption scheme from a simple idea that has been the base for several others [47, 36, 41, 42].

Multi-key Fully Homomorphic Encryption:

A motivation for FHE is the ability to be able to encrypt data locally but still outsource the computation of the encrypted data without risking to expose the actual data. FHE can only handle this in a single user setting where the ciphertexts are encrypted under the same key. In order to be able to compute a function on ciphertexts encrypted under different public keys

multi-key FHE was introduced by Lopéz-Alt et al. [44]. All the secret keys of the parties involved are needed in order to decrypt the ciphertext after the computation [44, 50, 48, 36, 47]. In other words, the parties involved need to jointly decrypt the ciphertext to obtain the output.

Multi-key GSW:

The GSW scheme does not support multi-key by nature but it is possible to convert the GSW FHE into a multi-key FHE. In 2014, Clear and McGoldrick showed how to extend the GSW scheme into multi-key FHE. This resulted in the first multi-key FHE based on LWE [50]. In 2016, this work was simplified and improved further by Mukherjee and Wichs [47]. The result was a Single-Hop Multi-Key (SHMK) FHE scheme. The property of single-hop in this scheme means that all relevant keys must be known at the start of the homomorphic computation and the output cannot be combined with ciphertexts encrypted under other keys in a useful way without a bootstrapping step being performed. This means that in both of these works all input needed to be known in advance before the computation starts. In 2016, this requirement was removed by Brakerski and Perlman [36] when they showed how to extend the prior work to support an unbounded number of homomorphic operations for an unbounded number of parties. In their work input from new parties can be introduced into the computation dynamically. In addition, they also improved the length of the ciphertexts and the space complexity of an atomic operation. This scheme is called Fully Dynamic Multi-Key (FDMK) FHE scheme. The fact that input from new parties can be introduced into the computation dynamically is what makes the scheme dynamic. This is achieved via bootstrapping that was introduced by Gentry [5]. (see [2])

Gentry's scheme and its implementations:

An encryption scheme is homomorphic if it supports operations on encrypted data.

As we mentioned above, in his breakthrough work, Gentry described in 2009 the first encryption scheme that supports both addition and multiplication on ciphertexts, *i.e.* a fully homomorphic encryption scheme [5]. The construction proceeds by successive steps:

First Gentry describes a "somewhat homomorphic" scheme that supports a limited number of additions and multiplications on ciphertexts. This is because every ciphertext has a noise component and any homomorphic operation applied to ciphertexts increases the noise in the resulting ciphertext. Once this noise reaches a certain threshold the resulting ciphertext does not decrypt correctly anymore; this limits the degree of the polynomial that can be applied to ciphertexts.

Secondly Gentry shows how to "squash" the decryption procedure so that it can be expressed as a low degree polynomial in the bits of the ciphertext and the secret key (equivalently a circuit of small depth). Then the breakthrough idea consists in evaluating this decryption polynomial not on the bits of the ciphertext and the secret key (as in regular decryption), but homomorphically on the encryption of those bits. Then instead of recovering the bit plaintext, one gets an encryption of this bit plaintext, i.e. yet another ciphertext for the same plaintext. Now if the degree of the decryption polynomial is small enough, the resulting noise in this new ciphertext can be smaller than in the original ciphertext; this is called the "ciphertext refresh" procedure.

Given two refreshed ciphertexts one can apply again the homomorphic operation (either addition or multiplication), which was not necessarily possible on the original ciphertexts because of the noise threshold. Using this "ciphertext refresh" procedure the number of permissible homomorphic operations becomes unlimited and we get a fully homomorphic encryption scheme.

The prerequisite for the "ciphertext refresh" procedure is that the degree of the polynomial that can be evaluated on ciphertexts exceeds the degree of the decryption polynomial (times two, since one must allow for a subsequent addition or multiplication of refreshed ciphertexts); this is called the "bootstrappability" condition. Once the scheme becomes bootstrappable it can be converted into a fully homomorphic encryption scheme by providing the encryption of the secret

key bits inside the public key. Based on Gentry's approach, two different fully homomorphic schemes are known: Gentry's scheme [5] based on ideal lattices and a scheme by van Dijk, Gentry, Halevi and Vaikuntanathan (DGHV) over the integers, that appeared at Eurocrypt 2010 [24].

Gentry described in [5] a somewhat homomorphic encryption scheme that is similar to GGH [51,52] over ideal lattices. To reduce the degree of the decryption polynomial, Gentry introduced Fully Homomorphic Encryption over the Integers with Shorter Public Keys with the following transformation [5]: instead of using the original secret key, the decryption procedure uses a very sparse subset of values that adds up to the secret key; the full set of values is made part of the public key. To apply the new decryption procedure the original ciphertext must first be "expanded" using the full set of public values. This expanded ciphertext can then be decrypted with a low-degree polynomial in the bits of the new secret key (which are the characteristic vector of the sparse subset sum); this is called the "squashed decryption" procedure.

At PKC 2010 Smart and Vercauteren [28] made the first attempt to implement Gentry's scheme using a variant based on principal ideal lattices and requiring that the determinant of the lattice be a prime number. However the authors of [28] could not obtain a bootstrappable scheme because that would have required a lattice dimension of at least $n = 227$, whereas due to the prime determinant requirement they could not generate keys for dimensions $n > 2048$.

Gentry and Halevi described in [53] the first implementation of Gentry's scheme. The authors follow the same direction as Smart and Vercauteren,

but for key generation they eliminate the requirement that the determinant is a prime. Additionally they present many clever optimizations. Four concrete parameter settings are provided, from a “toy” setting in dimension 512, to “small”, “medium” and “large” settings of dimensions 2048, 8192 and 32768, respectively. For the “large” setting public key size is 2.3 Gigabytes. The authors of [6] report that for an optimized implementation on a high-end workstation, key generation takes 2.2 hours, encryption takes 3 minutes, and ciphertext refresh takes 30 minutes.

At Eurocrypt 2010, van Dijk, Gentry, Halevi and Vaikuntanathan described a fully homomorphic encryption scheme over the integers [4]. As in Gentry’s scheme the authors first describe a somewhat homomorphic scheme supporting a limited number of additions and multiplications over encrypted bits. Then they apply Gentry’s “squash decryption” technique to get a bootstrappable scheme and then Gentry’s “ciphertext refresh” procedure to get a fully homomorphic scheme. The main appeal of the scheme (compared to the original Gentry’s scheme) is its conceptual simplicity: all operations are done over the integers instead of ideal lattices. [54]

5.2. Libraries

Beside theoretical research, the field of homomorphic encryption has been strongly active with regard to implementation efforts. In fact, given the strongly applied nature of this sort of constructions, it is appropriate that the development of new mathematical techniques and theoretical breakthrough proceeds closely associated with top quality implementation. On one hand, this contributes to estimate where we actually stand and how far the goal of practically usable homomorphic encryption is. On the other hand, this also helps to determine the limitations and the main difficulties that still remain. This is very important because it suggests to researchers and theoreticians where to concentrate their efforts.

Due to the significant advancement of proposing new efficient schemes, open-source implementations and applications and growing demands from industry, there is a standard for FHE initiated to standardize FHE schemes to have an unified and simplified API, and clear and understandable security properties for use by non-experts as well as experts. The homomorphic encryption standards meetings are held once a year at different locations. The participants to the standardization meetings are from industry, academia and government [67, 68, 69].

Over the years have been released many implementations by numerous authors, via means like GitHub, but we will consider here only a few of them, in order to give a broad idea on the current state of the art on this matter.

In particular, in this subsection, we will concetrate on HElib [55, 56, 57], SEAL [58], and TFHE [60, 61, 59].

HElib**:** An early and widely used library from IBM that supports the BGV scheme and bootstrapping.

This library, authored mainly by Shai Halevi and Victor Shoup, implements the BGV scheme [40]. It is one of the most widely used libraries in applications. This library enables for packing of ciphertexts and SIMD computations, amortizing the cost for certain tasks. It is able to perform additions and multiplications in an efficient way, but the bootstrapping operation is significantly slow. In practice, this library is often used as a somewhat homomorphic encryption scheme. One of the disadvantage of this library is that it considers the parameter selection. But this remains a complicated and error-prone operation. In fact, in [57, Appendix A], the authors note that

“The BGV implementation in HElib relies on a myriad of parameters, some of which are heuristically chosen, and so it takes some experimentation to set them all so as to get a working implementation with good performance”.

SEAL**:** A widely used open source library from Microsoft that supports the BFV and the CKKS schemes. This library is written in C**++**11 and implements the FV encryption scheme [4].

It should be noted that this scheme is already implemented in [62, 63], both of which use the ideal lattice library NFLlib [64]. As opposed to HElib, with the SEAL library it is considerably easier to set good parameters for the performance and the security of the implemented scheme. Microsoft SEAL supports the BFV and CKKS schemes. From a development perspective, SEAL recently released a .NET standard wrappers.

PALISADE: A widely-used open source library from a consortium of DARPA-funded defense contractors that provides lattice cryptography building blocks and supports leading homomorphic encryption schemes. PALISADE supports the BGV, BFV, CKKS, and FHEW schemes.

FHEW/TFHE (Torus-FHE): GSW-based libraries with fast bootstrapped operations. TFHE is designed from FHEW, which is no longer actively being developed. This library implements a generalized version of the GSW encryption scheme [65, 66]. The library features a very efficient operation bootstrapping, with timings that are in the order of fractions of a second on average machines. As a drawback, this bootstrapping operation has to be applied after computing every gate of the circuit. When used for realizing an FHE scheme, this library is more efficient than HElib. However, for simple tasks requiring small computational depth, HElib used as a somewhat homomorphic encryption scheme will generally perform better. Moreover, TFHE is currently not capable of amortizing large SIMD computations as well as HElib does. However, it should be noted that the code of TFHE is still largely under development, and more features (e.g., multithreading) are likely to be added soon. [20]

HeaAn: This library implements the CKKS scheme with native support for fixed point approximate arithmetic.

Λ ∘ Λ (pronounced “L O L”): This is a Haskell library for ring-based lattice cryptography that supports FHE.

NFLlib: This library is an outgrowth of the European HEAT project to explore high-performance

homomorphic encryption using low-level processor primitives.
HEAT: This library focuses on an API that bridges FV-NFLib and HeLIB.
HEAT is a HW accelerator implementation for FV-NFLlib.
cuHE: This library explores the use of GPGPUs to accelerate homomorphic encryption.
Lattigo: This is a lattice-based cryptographic library written in Go. Lattigo also implemented
the BFV and CKKS and supports secure multiparty computation based on the threshold
or distributed key. [1]

VI.APPLICATIONS OF FHE

Fully Homomorphic Encryption schemes can be either public key (where the encryptor knows the decryptor's public key but not her private key) or symmetric key (where the encryptor and decryptor share a key that is used for both encryption and decryption). [3]
Although still slow, homomorphic encryption has been proposed for several practical uses. This section explores the numerous applications of the various flavours of homomorphic encryption. Some require fully homomorphic encryption, while others just need somewhat homomorphic encryption.
1- A major application of FHE is to cloud computing. Alice can store her data in “the cloud”, for example, on remote servers that she accesses via the Internet. The cloud has more storage capabilities and computing power than does Alice, so when Alice needs computations to be done on her data, she would like those computations to be done by the cloud. However, Alice doesn't trust the cloud. Her data might be sensitive (for example, Alice might be a hospital and the data might be patients' medical records). So, Alice would like the cloud to know as little as possible about her data, and about the results of the computations. Hence, Alice sends encrypted data to the cloud, which can perform arithmetic operations on it without learning anything about the original raw data, by performing operations on the encrypted data.
2- Consumer Privacy in Advertising
Though often unwanted, advertising can be useful when adapted to user needs, e.g. through recommender systems or through location-based advertising. However, many users have to do with the privacy of their data, in this case their preferences or location. There have been several approaches to this problem.
Jeckmans et al. [70] sketch a framework where a user wants recommendations for a product. The framework is composed around a social network where recommendations are based on the tastes of the user's friends with the condition of confidentiality. The proposed system applies homomorphic encryption to allow a user to obtain recommendations from friends without the identity of the recommender being discovered.
Armknecht and Strufe [71] presented a recommender system where a user gets encrypted recommendations without the system being aware of the content. This system builds upon a very simple but highly efficient homomorphic encryption scheme which has been developed for this purpose. This allows a function to be computed which chooses the advertisement for each user while the advertising remains encrypted.
In another approach to personalized advertising [72] a mobile device sends a user's location to a provider, who sends customized public notices of sale, such as discount vouchers for nearby shops, back to the user. Of course, this potentially allows the provider to monitor everything about the user's habits and preferences.
However, this problem can be solved by homomorphic encryption, provided the advertisements come from a third party (or several) and there is no secret cooperation with the provider.
3- Medical Applications
Naehrig et al. [72] propose a scenario where a patient's medical data is (continuously) uploaded to a service provider in encrypted form. Here, the user is the data owner, so the data is encrypted under the user's public key and only the user can decrypt. The service provider then computes on the encrypted data. These data could consist of things like blood pressure, heart rate, weight or blood sugar reading to predict the possibility of certain conditions occurring or more generally to just remain fully aware of the user's health. The main benefit here is to allow real-time health analysis based on readings from various sources without having to reveal this data to any one source. Lauter [73] described an actual implementation of a heart attack prediction by Microsoft.
4- Data Mining
Mining from large data sets offers great value, but the price for this is the user's privacy. While Yang, Zhong and Wright [74] are often cited as using homomorphic encryption as a solution to this problem, the scheme actually uses functional encryption. However, applying homomorphic encryption is certainly feasible as a solution.
5- Financial Privacy
Let us think a framework where a corporation has sensitive data and also proprietary algorithms that they do not want revealed, e.g. stock price prediction algorithms in the financial sector. Naehrig et al. [72] propose the use of homomorphic encryption to upload both the data and the algorithm in encrypted form in order to delegate the computations to a cloud service. However, keeping the algorithm secret is not something that homomorphic encryption offers, but is rather part of obfuscation research.
The attribute that comes closest in fully homomorphic schemes is called circuit privacy, but this only guarantees that no information about the function is leaked by the output, not that one can encrypt the function itself. What homomorphic encryption offers is the solution to a related problem. Imagine that a corporation A has sensitive data, like a stock portfolio, and another company B has secret algorithms that

make predictions about the stock price. If A would like to use B's algorithms (for a price, of course), either A would have to disclose the stock portfolio to B, or B has to give the algorithm to A. However, with homomorphic encryption, A can encrypt the data with a circuit private scheme and send it to B, who runs the proprietary algorithm and only sends back the result. This result can only be decrypted by A's secret key. This way, B does not learn anything about A's data, and A does not learn anything about the algorithms used.

6- Forensic Image Recognition

Fully Homomorphic Encryption (FHE) [8,5,6] is an encryption scheme with the special property of enabling computing on the encrypted data, while simultaneously protecting its secrecy. (see for example [55]) Specifically, FHE allows computing any algorithm on encrypted input (ciphertexts), with no decryption or access to the secret key that would compromise secrecy, yet succeeding in returning the encryption of the desired outcome.

Bösch et al. [75] describe how to utilize forensic image recognition. Tools similar to this are being used by the police and other law enforcement agencies to detect illegal images in a hard drive, network data streams and other data sets. The police use a database containing hash values of "bad" pictures.

A major concern is that perpetrators could obtain this database, check if their images would be detected and, if so, change them. This scheme uses a somewhat homomorphic encryption scheme proposed by Brakerski and Vaikuntanathan [30] to realise a scenario where the police database is encrypted while at the same time the company's legitimate network traffic stays private. The company compares the hashed and encrypted picture data stream with the encrypted database created by the police. The service provider learns nothing about the encrypted database itself, and after a given time interval or threshold, the temporary variable is sent to the police. [76]

7- Fully Homomorphic Encryption can be used to query a search engine, without disclosing what is being searched for (here, the search engine is doing the computations on encryptions of information that it doesn't know). Secure search using FHE has been the indication example for useful FHE applications since Gentry's break through result construction the first FHE candidate [5]. Use case examples are abundant: secure search for a document matching a retrieval query in a corpus of sensitive documents, such as private emails, classified military documents, or sensitive corporate documents; secure SQL SELECT WHERE query to a database, e.g., searching for a patient's record in a medical database based on desired attributes; secure search engine; etc. In all these use cases security means that both the searched data (documents, DB, etc.) and the search query are encrypted with semantically secure FHE, and that the data access pattern reveal no information on the searched data or query, as well.

The secure search problem at the core of all previously mentioned use case examples can be represented as searching for an encrypted lookup value in an encrypted array (the array representing, for example, an encrypted table/column in a relational database, or a word-by-word encryption of a document for full text search).

8- One of the most popular applications of FHE has been Machine Learning. There are many works which are focused on Neural Networks and different variants of regression. To our knowledge, all works in this line are concerned with supervised learning. This means that there is a training set with known outcomes, and the algorithm tries to build a model that matches the desired outputs to the inputs as well as possible. When the training phase is done, the algorithm can be applied to new instances to predict unknown outcomes.

Machine Learning as an application of FHE was first proposed in [72], and subsequently there have been numerous works on the subject, to our insight all concerned with supervised learning. The most popular of these applications seem to be (Deep) Neural Networks (see [78], [79], [80], [81], and [82]) and (Linear) Regression (e.g., [83], [84], [85] or [86]), though there is also some work on other algorithm classes like decision trees and random forests ([87]), or logistic regression ([88],[89] and [90]). [91]

Fully homomorphic encryption (FHE) is one of the prospective tools for privacy-preserving machine learning (PPML), and several PPML models have been proposed based on various FHE schemes and approaches. Although the FHE schemes are known as suitable tools to implement PPML models, previous PPML models on FHE such as CryptoNet, SEALion, and CryptoDL are limited to only simple and non-standard types of machine learning models. These non-standard machine learning models are not proven efficient and accurate with more practical and advanced datasets. Previous PPML schemes replace non-arithmetic activation functions with simple arithmetic functions instead of adopting approximation methods and do not use bootstrapping, which enables continuous homomorphic evaluations. Thus, they could not use standard activation functions and could not employ a large number of layers.

The privacy-preserving issue is one of the most practical problems for machine learning recently. Fully homomorphic encryption (FHE) is the most appropriate tool for privacy-preserving machine learning (PPML) to ensure strong security in the cryptographic sense and satisfy the communication's conciseness.

Privacy-preserving machine learning on fully homomorphic encryption (FHE) is one of the most influential applications of the FHE scheme.

The maximum classification accuracy of the existing PPML model with the FHE for the CIFAR-10 dataset was only 77% until now.

FHE is an encryption scheme whose ciphertexts can be processed with any deep Boolean circuits or arithmetic circuits without access to the data. The security of FHE has been usually defined with indistinguishability under chosen-plaintext attack (IND-

CPA) security, which is a standard cryptographic security definition. If the client sends the public keys and the encrypted data with an FHE scheme to the PPML server, the server can perform all computation needed in the desired service before sending the encrypted output to the client. Therefore, the application of FHE to PPML has been researched much until now.

9- Genome sequencing is now practical on a large scale, and the cost of data storage continues to decrease. As a result, genome sequencing is now commercially available for personalized genetic analysis. Genomic data has been used for applications in multiple domains including healthcare, biomedical research, disease risk tests, and forensics [93]. Because we are uniquely identified by our genetic code, a privacy breach compromising this data could have an unforeseen negative impact upon a person's life.

Homomorphic encryption has been used for a variety of applications in bioinformatics. A large portion of this work is driven by the annual iDASH workshop's homomorphic encryption competition tasks. Lauter et al. provided some of the first applications of FHE for computation over encrypted genomic data [69]. They provided algorithms for homomorphic computation of basic genomic algorithms such as the Pearson goodness-of-fit test, the D' and r^2-measures of linkage disequilibrium, the Estimation Maximization (EM) algorithm for haplotyping, and the Cochran-Armitage test for trend. [46]

VII.CONCLUSIONS

We live in an era where several billion devices are connected to the Internet, and this number will continue to grow. This is a consequence of not only more people becoming interested in consumer electronics but also more sensors and actuators being incorporated into everyday electronics, household appliances, and the general infrastructure. Therefore, there is a growing need for services that let users easily store and access personal files. Data that was previously stored on paper is being converted to a digital file. However, some data needs to be kept secret and is not meant for public consumption. But most encryption schemes no longer produce the correct decrypted value once computations have been performed on the data. As we mentioned above, this is not the case with homomorphic encryption schemes. In this survey, we explain shortly what FHE is, describing some of its schemes and applications, as well. The interested reader can consult the literature in this paper and not only, because there exists, already, a very rich one, to expand the knowledge in this very attractive area of current research.

ACKNOWLEDGMENT *(Heading 5)*

I want to express my gratitude to many authors such as [1, 5, 20] etc. for their excellent help and inspiration they offer to me in this research and my sister for her great moral support in its fulfillment.

REFERENCES

[1] Yao Chen , Applying Fully Homomorphic Encryption: Practices and Problems, 2021.

[2] Emelie Widegren, Fully Homomorphic Encryption: A Case Study, 2018.

[3] Alice Silverberg, Fully Homomorphic Encryption For Mathematicians, 2013.

[4] Junfeng Fan and Frederik Vercauteren, Somewhat Practical Fully Homomorphic Encryption, 2012.

[5] Craig Gentry. A fully homomorphic encryption scheme. PhD thesis, Stanford University, 2009, crypto.stanford.edu/craig.

[6] Craig Gentry. Fully homomorphic encryption using ideal lattices, ACM, 2009, pp.169-178.

[7] Vinod Vaikuntanathan, Computing Blindfolded: New Developments in Fully Homomorphic Encryption.

[8] R. Rivest, L. Adleman, and M. Dertouzos, On data banks and privacy homomorphisms, Foundations of Secure Computation. Academic Press, 1978, pp. 169–177.

[9] S. Goldwasser and S. Micali, Probabilistic encryption, Journal of Computer and System Sciences, vol. 28, no. 2, 1984, pp. 270–299.

[10] M. Bellare and P. Rogaway, Optimal asymmetric encryption, in EUROCRYPT, 1994, pp. 92-111.

[11] T. E. Gamal, A public key cryptosystem and a signature scheme based on discrete logarithms, in CRYPTO, 1984, pp. 10–18.

[12] P. Paillier, Public-key cryptosystems based on composite degree residuosity classes, in EUROCRYPT, 1999, pp. 223–238.

[13] I. Dämgard and M. Jurik, A generalisation, a simplification and some applications of paillier's probabilistic public-key system, in Public Key Cryptography, 2001, pp. 119–136.

[14] M. Ajtai and C. Dwork, A public-key cryptosystem with worst-case/average-case equivalence, in STOC, 1997, pp. 284–293.

[15] O. Regev, New lattice-based cryptographic constructions, J. ACM, vol. 51, no. 6, 2004, pp. 899-942.

[16] O. Regev, On lattices, learning with errors, random linear codes, and cryptography, in STOC, 2005, pp. 84–93.

[17] J. D. Cohen and M. J. Fischer, A robust and verifiable cryptographically secure election scheme (extended abstract), in FOCS. IEEE, 1985, pp. 372–382.

[18] D. Naccache and J. Stern, A new public key cryptosystem based on higher residues, ACM

Conference on Computer and Communications Security, 1998, pp. 59–66.

[19] T. Okamoto and S. Uchiyama, A new public-key cryptosystem as secure as factoring, in EUROCRYPT, 1998, pp. 308–318.

[20] Michele Minelli, Fully homomorphic encryption for machine learning, 2018.

[21] B. Adida, Helios: Web-based open-audit voting, in USENIX Security Symposium, 2008, pp. 335-348.

[22] C. Peikert and B. Waters, Lossy trapdoor functions and their applications, in STOC, 2008, pp. 187–196.

[23] D. Boneh, E.-J. Goh, and K. Nissim, Evaluating 2-DNF formulas on ciphertexts, in Theory of Cryptography - TCC'05, ser. Lecture Notes in Computer Science, vol. 3378. Springer, 2005, pp. 325-341 .

[24] C. Gentry, S. Halevi, and V. Vaikuntanathan, A simple BGN-type cryptosystem from LWE, in EUROCRYPT, 2010, pp. 506–522.

[25] M. Fellows and N. Koblitz, Combinatorial cryptosystems galore!, Finite Fields: Theory, Applications and Algorithms, 1993, pp. 51–61.

[26] E. Kushilevitz and R. Ostrovsky, Replication is not needed: Single database, computationally-private information retrieval, in FOCS, 1997, pp. 364–373.

[27] M. Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, Fully homomorphic encryption over the integers, in EUROCRYPT, 2010, pp. 24–43, http://eprint.iacr.org/2009/616.pdf.

[28] N. P. Smart and F. Vercauteren, Fully homomorphic encryption with relatively small key and ciphertext sizes, Public Key Cryptography, ser. Lecture Notes in Computer Science, P. Q. Nguyen and D. Pointcheval, Eds., vol. 6056. Springer, 2010, pp. 420–443.

[29] D. Stehlé and R. Steinfeld, Faster fully homomorphic encryption, in ASIACRYPT, 2010, pp. 377-394.

[30] Z. Brakerski and V. Vaikuntanathan, Fully homomorphic encryption from ring-LWE and security for key dependent messages, in CRYPTO, vol. 6841, 2011.

[31] Paulo Martins, Leonel Sousa and Artur Mariano, A Survey on Fully Homomorphic Encryption: An Engineering Perspective, *ACM Comput. Surv.* 50, 6, Article 83, 2017, https://doi.org/10.1145/3124441.

[32] Abbas Acar, Hidayet Aksu, A. Selcuk Uluagac, Mauro Conti, Survey on Homomorphic Encryption Schemes: Theory and Implementation, 2018, ACM 0360-0300/2018/07-ART79, https://doi.org/10.1145/3214303.

[33] Samiha Jlilab, Hassan Satori, Khalid Satori, Computing on Encrypted Data into the Cloud through Fully Homomorphic Encryption, TMLAI-Transactions on Machine Learning and Artificial Intelligence, Volume 5, No 4, 2017.

[34] M. Seetha and A. K. Koundinya, Comparative Study and Performance Analysis of Encryption in RSA , ECC and GoldwasserMicali Cryptosystems, vol. 3, no. 1, 2014, pp. 111–118.

[35] Jaydip Sen , Homomorphic Encryption: Theory & Applications, 2013.

[36] Z. Brakerski and R. Perlman, Lattice-based fully dynamic multi-key fhe with short ciphertexts, in *Annual Cryptology Conference*, Springer 2016, pp. 190–213.

[37] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, Fully homomorphic encryption without bootstrapping. Cryptology ePrint Archive, Report 2011/277, 2011, http://eprint.iacr.org/2011/277.

[38] N. P. Smart and F. Vercauteren, Fully homomorphic simd operations, *Des. Codes Cryptography*, vol. 71, 2014, pp. 57–81.

[39] Z. Brakerski and V. Vaikuntanathan, Efficient fully homomorphic encryption from (standard) lwe, *SIAM Journal on Computing*, vol. 43, no. 2, 2014, pp. 831–871.

[40] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, (leveled) fully homomorphic encryption without bootstrapping, in *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, ACM, 2012, pp. 309–325.

[41] L. Ducas and D. Micciancio, Fhew: Bootstrapping homomorphic encryption in less than a second, in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, Springer, 2015, pp. 617–640.

[42] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène, Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. Cryptology ePrint Archive, Report 2016/870, 2016, http://eprint.iacr.org/2016/870.

[43] S. Halevi and V. Shoup, Helib. https://github.com/shaih/HElib. 2017.

[44] A. López-Alt, E. Tromer, and V. Vaikuntanathan, On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption, in *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, ACM, 2012, pp. 1219–1234.

[45] J. Hoffstein, J. Pipher, and J. H. Silverman, Ntru: A ring-based public key cryptosystem, in *International Algorithmic Number Theory Symposium*, Springer, 1998, pp. 267–288.

[46] Alexander Wood, Kayvan Najarian, Delaram Kahrobaei, Homomorphic Encryption for Machine Learning in Medicine and Bioinformatics, ACM Comput. Surv. 0, 0, Article 0 (0000), https://doi.org/10.1145/1122445.1122456.

[47] P. Mukherjee and D. Wichs, Two round multiparty computation via multi-key fhe, in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, Springer, 2016, pp. 735-763.

[48] C. Peikert and S. Shiehian, Multi-key fhe from lwe, revisited, in *Theory of Cryptography Conference*, Springer, 2016, pp. 217–238.

[49] C. Gentry, A. Sahai, and B. Waters, Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based, in *Advances in Cryptology– CRYPTO 2013*, Springer, 2013, pp. 75–92.

[50] M. Clear and C. McGoldrick, Multi-identity and multi-key leveled fhe from learning with errors, Cryptology ePrint Archive, Report 2014/798, 2014. http: /eprint.iacr.org/2014/798.

[51] Goldreich, O., Goldwasser, S., Halevi, S.: Public-key cryptosystems from lattice reduction problems. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, Springer, Heidelberg 1997, pp. 112–131.

[52] Micciancio D, Improving Lattice Based Cryptosystems Using the Hermite Normal Form. In Silverman, J.H. (ed.) CaLC 2001. LNCS, vol. 2146, Springer, Heidelberg, 2001, pp. 126–145.

[53] Gentry, C., Halevi, S. Implementing Gentry's Fully-Homomorphic Encryption Scheme. In Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, Springer, Heidelberg 2011, pp. 129–148.

[54] Jean-Sébastien Coron, Avradip Mandal, David Naccache, and Mehdi Tibouchi, Fully Homomorphic Encryption over the Integers with Shorter Public Keys, P. Rogaway (Ed.) CRYPTO 2011, LNCS 6841, pp. 487–504.

[55] Shai Halevi and Victor Shoup. Algorithms in HElib. In: *CRYPTO 2014, Part I*. Ed. by Juan A. Garay and Rosario Gennaro. Vol. 8616. LNCS. Springer, Heidelberg, 2014, pp. 554-571, doi: 10.1007/978-3-662-44371-2_31.

[56] Shai Halevi and Victor Shoup. HElib - An implementation of homomorphic encryption, 2014, https://github.com/shaih/HElib.

[57] Shai Halevi and Victor Shoup. Bootstrapping for HElib. In: *EUROCRYPT 2015, Part I*. Ed. by Elisabeth Oswald and Marc Fischlin. Vol. 9056. LNCS. Springer, Heidelberg, 2015, pp. 641-670, doi: 10.1007/978-3-662-46800-5_25.

[58] Hao Chen, Kyoohyung Han, Zhicong Huang, Amir Jalali, and Kim Laine, Simple Encrypted Arithmetic Library (SEAL) v2.3.0. https://www.microsoft.com/en-us/research/project/simple-encrypted-arithmetic-library/, 2016.

[59] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène. TFHE: Fast FullyHomomorphic Encryption Library over the Torus, 2016, https : / / github . com /tfhe/tfhe.

[60] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Faster Fully Homomorphic Encryption: Bootstrapping in Less Than 0.1 Seconds. In: *ASIACRYPT 2016, Part I*. Ed. by Jung Hee Cheon and Tsuyoshi Takagi. Vol. 10031. LNCS. Springer, Heidelberg, 2016, pp. 3–33. doi: 10.1007/978-3-662-53887-6_1.

[61] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène, Faster Packed Homomorphic Operations and Efficient Circuit Bootstrapping for TFHE, In: ASIACRYPT 2017, Part I. Ed. by Tsuyoshi Takagi and Thomas Peyrin. Vol. 10624. LNCS. Springer, Heidelberg, 2017, pp. 377–408.

[62] Jean-Claude Bajard, Julien Eynard, Anwar Hasan, and Vincent Zucca. A Full RNS Variant of FV like Somewhat Homomorphic Encryption Schemes. Cryptology ePrint Archive, Report 2016/510, http://eprint.iacr.org/2016/510, 2016.

[63] T. Lepoint, FV-NFLlib, GitHub repository. https : / / github . com /CryptoExperts/FV- NFLlib, 2016.

[64] Carlos Aguilar Melchor, Joris Barrier, Serge Guelton, Adrien Guinet, Marc-Olivier Killijian and Tancrède Lepoint. NFLlib: NTT-Based Fast Lattice Library. In: *CT-RSA 2016*. Ed. by Kazue Sako. Vol. 9610. LNCS. Springer, Heidelberg, 2016, pp. 341–356. doi: 10.1007/978-3-319-29485-8_20.

[65] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic Encryption from Learning with Errors: Conceptually-Simpler, Asymptotically-Faster, Attribute-Based. In: *CRYPTO 2013, Part I*. Ed. by Ran Canetti and Juan A. Garay. Vol. 8042. LNCS. Springer, Heidelberg, 2013, pp. 75–92. doi: 10.1007/978-3-642-40041-4_5.

[66] Jacob Alperin-Sheriff and Chris Peikert. Faster Bootstrapping with Polynomial Error. In *CRYPTO2014, Part I*. Ed. by Juan A. Garay and Rosario Gennaro. Vol. 8616. LNCS. Springer, Heidelberg, 2014, pp. 297–314. doi: 10.1007/978-3-662-44371-2_17.

[67] Martin R Albrecht, Melissa Chase, Hao Chen, Jintai Ding, Shafi Goldwasser, Sergey Gorbunov, Shai Halevi, Jeffrey Hoffstein, Kim Laine, Kristin E Lauter, Homomorphic encryption standard. IACR Cryptol. ePrint Arch., 2019:939, 2019.

[68] Martin R Albrecht, Melissa Chase, Hao Chen, Jintai Ding, Shafi Goldwasser, Sergey Gorbunov, Shai Halevi, Jeffrey Hoffstein, Kim Laine, Kristin E Lauter, 2018.

[69] Kristin Lauter, Adriana Lopez-Alt, and Michael Naehrig. 2015. Private Computation on Encrypted Genomic Data. Springer International Publishing, Cham.

[70] Arjan Jeckmans, Andreas Peter, and Pieter H. Hartel. Efficient privacy-enhanced familiarity- based recommender system. In Jason Crampton et al. editors, Computer Security ESORICS 2013, volume 8134 of Lecture Notes in Computer Science, Springer, 2013, pages 400-417.

[71] Frederik Armknecht and Thorsten Strufe. An efficient distributed privacy-preserving recommendation system. In The 10th IFIP Annual Mediterranean Ad Hoc Networking Workshop, Med-Hoc-Net 2011, IEEE, 2011, pages 65-70.

[72] Michael Naehrig, Kristin E. Lauter, and Vinod Vaikuntanathan. Can homomorphic encryption be practical? In Christian Cachin and Thomas Ristenpart, editors, Proceedings of the 3rd ACM Cloud Computing Security Workshop, CCSW, ACM, 2011, pages 113-124.

[73] Kristin Lauter. Practical applications of homomorphic encryption, 2015.

[74] Zhiqiang Yang et al. Privacy-preserving classification of customer data without loss of accuracy, In Hillol Kargupta et al., editors, Proceedings of the 2005 SIAM International Conference on Data Mining, SDM 2005, pages 92-102.

[75] Christoph Bösch, Andreas Peter, Pieter H. Hartel, and Willem Jonker. SOFIR: securely outsourced forensic image recognition. In IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2014, pages 2694-2698.

[76] Frederik Armknecht, Colin Boyd, Christopher Carr, Kristian Gjosteen, Angela Jäschke, Christian A. Reuter, and Martin Strand, A Guide to Fully Homomorphic Encryption, 2015.

[77] Adi Akavia, Dan Feldman, and Hayim Shaul, Secure Search via Multi-Ring Fully Homomorphic Encryption, 2018.

[78] Jäschke, A., Armknecht, F.: Accelerating homomorphic computations on rational numbers, In: ACNS 2016.

[79] Gilad-Bachrach, R., Dowlin, N., Laine, K., Lauter, K.E., Naehrig, M., Wernsing, J. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In: ICML 2016.

[80] Chabanne, H., de Wargny, A., Milgram, J., Morel, C., Prouff, E. Privacypreserving classification on deep neural network. IACR Cryptology ePrint Archive (2017/035).

[81] Phong, L.T., Aono, Y., Hayashi, T., Wang, L., Moriai, S. Privacy-preserving deep learning via additively homomorphic encryption. IACR Cryptology ePrint Archive (2017/715).

[82] Bost, R., Popa, R.A., Tu, S., Goldwasser, S., Machine learning classification over encrypted data. In: NDSS 2015.

[83] Lu, W., Kawasaki, S., Sakuma, J., Using fully homomorphic encryption for statistical analysis of categorical, ordinal and numerical data, IACR Cryptology ePrint Archive 2016/1163).

[84] Esperança, P.M., Aslett, L.J.M., Holmes, C.C., Encrypted accelerated least squares regression, In: Singh, A., Zhu, X.J. (eds.) AISTATS 2017.

[85] Barnett, A., Santokhi, J., Simpson, M., Smart, N.P., Stainton-Bygrave, C., Vivek, S., Waller ., Image classification using non-linear support vector machines on encrypted data. IACR Cryptology ePrint Archive (2017/857).

[86] Graepel, T., Lauter, K.E., Naehrig, M., ML confidential: Machine learning on encrypted data, In: ICISC 2012.

[87] Wu, D.J., Feng, T., Naehrig, M., Lauter, K.E., Privately evaluating decision trees and random forests, PoPETs (4), 2016.

[88] Kim, M., Song, Y., Wang, S., Xia, Y., Jiang, X., Secure logistic regression based on homomorphic encryption, IACR Cryptology ePrint Archive (074), 2018.

[89] Kim, A., Song, Y., Kim, M., Lee, K., Cheon, J.H., Logistic regression model training based on the approximate homomorphic encryption, IACR Cryptology ePrint Archive (254), 2018.

[90] Bonte, C., Vercauteren, F., Privacy-preserving logistic regression training, IACR Cryptology ePrint Archive 233, 2018.

[91] Angela Jäschke and Frederik Armknecht, Unsupervised Machine Learning on Encrypted Data, 2018.

[92] 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 No, Privacy-Preserving Machine Learning with Fully Homomorphic Encryption for Deep Neural Network, 2021.

[93] Muhammad Naveed, Erman Ayday, Ellen W. Clayton, Jacques Fellay, Carl A. Gunter, Jean- Pierre Hubaux, Bradley A. Malin, and Xiaofeng Wang, Privacy in the Genomic Era, ACM computing surveys 48, 1, 2015.