Peer-to-Peer Networking and Applications (2021) 14:1666–1691
https://doi.org/10.1007/s12083-021-01076-8

Privacy-preserving neural networks with Homomorphic encryption: Challenges and opportunities

Bernardo Pulido-Gaytan[1] · Andrei Tchernykh[1,2,3] · Jorge M. Cortés-Mendoza[2] · Mikhail Babenko[4] · Gleb Radchenko[2] · Arutyun Avetisyan[3] · Alexander Yu Drozdov[5]

Received: 29 July 2020 / Accepted: 13 January 2021 / Published online: 8 March 2021

Abstract
Classical machine learning modeling demands considerable computing power for internal calculations and training with big data in a reasonable amount of time. In recent years, clouds provide services to facilitate this process, but it introduces new security threats of data breaches. Modern encryption techniques ensure security and are considered as the best option to protect stored data and data in transit from an unauthorized third-party. However, a decryption process is necessary when the data must be processed or analyzed, falling into the initial problem of data vulnerability. Fully Homomorphic Encryption (FHE) is considered the holy grail of cryptography. It allows a non-trustworthy third-party resource to process encrypted information without disclosing confidential data. In this paper, we analyze the fundamental concepts of FHE, practical implementations, state-of-the-art approaches, limitations, advantages, disadvantages, potential applications, and development tools focusing on neural networks. In recent years, FHE development demonstrates remarkable progress. However, current literature in the homomorphic neural networks is almost exclusively addressed by practitioners looking for suitable implementations. It still lacks comprehensive and more thorough reviews. We focus on the privacy-preserving homomorphic encryption cryptosystems targeted at neural networks identifying current solutions, open issues, challenges, opportunities, and potential research directions.

Keywords Cloud security · Homomorphic encryption · Machine learning · Neural networks · Privacy-preserving

1 Introduction

Cloud computing offers considerable benefits of availability, scalability, pricing, energy efficiency, almost zero upfront infrastructure investment, just-in-time service provisioning, etc. However, it also brings security and privacy concerns, where data breaches are the top threat [1]. Sensitive information can be released, viewed, stolen, and used by an

This article is part of the Topical Collection: *Special Issue on Privacy-Preserving Computing*
Guest Editors: Kaiping Xue, Zhe Liu, Haojin Zhu, Miao Pan and David S.L. Wei

✉ Andrei Tchernykh
chernykh@cicese.mx

Bernardo Pulido-Gaytan
lpulido@cicese.edu.mx

Jorge M. Cortés-Mendoza
kortesmendosak@susu.ru

Mikhail Babenko
mgbabenko@ncfu.ru

Gleb Radchenko
gleb.radchenko@susu.ru

Arutyun Avetisyan
arut@ispras.ru

Alexander Yu Drozdov
alexander.y.drozdov@gmail.com

[1] CICESE Research Center, carr. Tijuana-Ensenada 3918, 22860 Ensenada, BC, Mexico

[2] South Ural State University, Prospekt Lenina 76, 454080 Chelyabinsk, Russia

[3] Ivannikov Institute for System Programming, Alexander Solzhenitsyn 25, Moscow 109004, Russia

[4] North-Caucasus Federal University, Kulakova 2, 355029 Stavropol, Russia

[5] Moscow Institute of Physics and Technology, Institutskiy 9, Dolgoprudny 141701, Russia

unauthorized third-party. Data outsourcing implies that the user delegates direct data control and its processing. The user requires greater trust in Cloud Service Providers (CSP) because dishonest behavior can compromise the data. In general, new threats appear since more data is outsourced.

Security and privacy are critical issues for preserving integrity, reliability, and availability in a cloud computing environment. Privacy and efficient data processing are important research areas in the field of outsourcing computing. Traditionally, encryption of confidential information was the standard solution before the use of the cloud model. It may protect user data privacy from a non-trustworthy third-party, but it cannot support effective ciphertext computing.

In this respect, data in use is a phase of the data life cycle in modern data security practices that goes overlooked. Conventional cryptosystems successfully protect stored data and data in transit but do not protect the data while decrypted to be processed. The data value extraction requires decryption, creating critical exposure points. As a result, privacy-preserving techniques are emerging as a key consideration in data analytics and cloud computing domains. The general idea is to delegate data processing without giving transparent access to it.

Fully Homomorphic Encryption (FHE) has been dubbed the holy grail of cryptography, an elusive goal that could solve cybersecurity problems [2–4]. FHE allows a non-trustworthy third-party to process encrypted information without disclosing confidential data. Since the remote server only sees encryptions and never has access to the secret key, users can be assured that it does not learn anything about their data or the computation output. This is an extremely valuable opportunity in the world of distributed computing and heterogeneous networks.

FHE enables applying basic mathematical operations directly to the ciphertext so that the decrypting of the ciphertext results in the same answer as applying the operations to the original unencrypted data. In other words, FHE enables compatibility between two critical factors: computing and privacy.

Today, the promising post-quantum tool based on HE is technically feasible for real-world domains, after years of being considered a purely theoretical problem [5]. However, its implementation exhibits several limitations in performance. A long-pursue application is a privacy-preserving machine learning model for predicting or classifying confidential information. These systems promise to work with encrypted data and have the same performance as their unencrypted versions, providing security and accuracy at the same time.

A critical limitation of the extensive adoption of machine learning is the low protection of sensitive information. In most cases, access to the datasets is forbidden due to privacy concerns, e.g., accessing medical datasets is a privacy violation of the patients. In such a domain, the privacy-preserving Neural Network (NN) models via Homomorphic Encryption (NN-HE) come to place.

In this paper, we consider the current state-of-art NN-HE systems focusing on the followings aspects:

- Latest issues related to the intersection of HE cryptosystems and NN models, and ways to overcome privacy and security threats arisen in present-day computing environments;
- Fundamental concepts of NN-HE, main theoretical results, capabilities, potential applications, and limitations, discussing their state-of-the-art and the state-of-the-practice;
- Important compromises between theoretical models and their feasibility, showing tendencies of combining their potentialities;
- High-level and low-level tools, frameworks, interfaces, languages, libraries, etc.;
- Privacy-preserving NN-HE development by retrieving academic publications in the last fifteen years to detect emerging trends in research and relevant application domains;
- NN-HE implementations using high-level and low-level tools to demonstrate the advantages and disadvantages of each approach.

This survey aims to give a knowledge basis to researchers and practitioners interested in knowing, applying, and extending NN-HE models.

The paper is structured as follows. Section 2 reviews the evolution of homomorphic encryption schemes. Section 3 provides the formal definition of fully homomorphic encryption and describes fundamental concepts, such as bootstrapping and key-switching. Section 4 discusses the state-of-the-art of homomorphic cryptosystems and machine learning. Section 5 addresses privacy-preserving NN-HE, focusing on challenges and opportunities towards a blind non-interactive classification. Section 6 presents NN-HE applications in real-world problems and current development tools. Section 7 presents NN-HE implementation examples using both high-level and low-level tools. Section 8 summarizes open research problems and challenges. Finally, we conclude and discuss future works in Section 9.

2 Homomorphic encryption

In this section, we discuss basic concepts of HE and their evolution based on representative works in the area.

In the cryptography field, the term HE defines a kind of encryption system able to perform certain computable functions over ciphertexts. The output maintains the features of the function and input format. The system has no access to

information on the ciphertexts and secret keys. It only uses publicly available information without risks of the data breach. The HE concept refers to a mapping between functions on the space of messages and ciphertexts. A homomorphic function applied to ciphertexts provides the same (after decryption) result as applying the function to the original unencrypted data.

Let m_1 and m_2 be messages, c_1 and c_2 be their corresponding ciphertexts. The operation $\ddot{+}$ in an additively HE produces the ciphertext $c_+ \leftarrow c_1 \ddot{+} c_2$ that can be decrypted to $m_1 + m_2$.

Similarly for $\ddot{\times}$ in a multiplicatively HE, it generates the ciphertext $c_\times \leftarrow c_1 \ddot{\times} c_2$ that is decrypted to $m_1 \cdot m_2$. Both HEs obtain ciphertexts c_+ and $c_\times$, without knowing m_1 and m_2. Conventional encryption cannot compute $m_1 + m_2$ and $m_1 \cdot m_2$ without the decryption of c_1 and c_2 first, when users sacrifice their privacy.

The HE is categorized according to the list of specific basic mathematical operations executed over encrypted data. The efficiency and flexibility of HE are strongly related to the number of operations in the list. A HE scheme with a higher number is more flexible but less efficient. In the opposite direction, a scheme with a lower number is less flexible but more efficient.

In the following sections, we describe three types of HE cryptosystems: Partially Homomorphic (PHE), Somewhat Homomorphic (SHE), and Fully Homomorphic (FHE) encryptions, and discuss their limitations and scopes.

2.1 Partially homomorphic encryption

PHE supports an unlimited number of one type of operation. For example, additive HE allows an unbounded number of additions but no multiplications.

Ronald Rivest, Adi Shamir, and Leonard Adleman (RSA) cryptosystem is the first multiplicative PHE [6]. In general, given two messages m_1 and m_2 and their respective ciphertexts $c_1 = \left(m_1^e\right) \bmod n$, and $c_2 = \left(m_2^e\right) \bmod n$, where e is chosen such that $gcd(e,\ \phi) = 1$ for $\phi = (q_1 - 1) \cdot (q_2 - 1)$ with large primes q_1 and q_2, and $n = q_1 \cdot q_2$. The ciphertext with the product of the original plaintexts is computed as

$$c_\times \leftarrow (m_1 \cdot m_2)^e \bmod n = \left(m_1^e\right) \bmod n \cdot \left(m_2^e\right) \bmod n$$
$$= c_1 \ddot{\times} c_2$$

RSA is not semantically secure as a result of its deterministic encryption algorithm. Taher El-Gamal is another relevant multiplicative PHE [7].

Shafi Goldwasser and Silvio Micali (GM) cryptosystem is the first additively PHE [8]. According to GM cryptosystem, there are two message m_1 and m_2 and their respective ciphertexts $c_1 = \left(b_1^2 \cdot e^{m_1}\right) \bmod n$ and $c_2 = \left(b_2^2 \cdot e^{m_2}\right) \bmod n$, where b_1^2 and b_2^2 are quadratic nonresidue values such that $gcd\left(b_1^2, n\right) = gcd\left(b_2^2, n\right) = 1$, and e is one of the quadratic nonresidue modulo n values with $(x/n) = 1$.

The GM scheme has a homomorphic property, where the encryption of $m_1 + m_2$, is

$$c_+ \leftarrow \left[(b_1 \cdot b_2)^2 \cdot e^{m_1+m_2}\right] \bmod n$$
$$= \left[\left(b_1^2 \cdot e^{m_1}\right) \cdot \left(b_2^2 \cdot e^{m_2}\right)\right] \bmod n$$
$$= \left(b_1^2 \cdot e^{m_1}\right) \bmod n \cdot \left(b_2^2 \cdot e^{m_2}\right) \bmod n = c_1 \ddot{+} c_2$$

However, GM is not an efficient scheme, as ciphertexts may be several hundred times larger than the initial plaintexts.

Relevant additive PHE cryptosystems are invented by and named after Josh (Cohen) Benaloh in 1994 [9], David Naccache and Jacques Stern (NS) in 1997 [10], Tatsuaki Okamoto and Shigenori Uchiyama (OU) in 1998 [11], Pascal Paillier in 1999 [12], Ivan Damgård and Mads Jurik (DJ) in 2001 [13], Steven Galbraith in 2002 [14], and Akinori Kawachi, Keisuke Tanaka and Keita Xagawa (KTX) in 2007 [15].

The encryption process in PHE does not guarantee a given level of security. The worst-case hardness of "*noisy*" problems is one direction in the security solution. The noise term denotes a moderate quantity of error injected in the encrypted message and generates a not exact relation [16].

2.2 Somewhat homomorphic encryption

SHE supports a predetermined amount of different homomorphic operations, limiting the number of allowed operations. Each operation increases the underlying noise, so its correct evaluation depends on performing only a bounded number of actions. Message decryption fails when noise overpasses a certain threshold.

Dan Boneh, Eu-Jin Goh, and Kobbi Nissim (BGN) scheme [17] was the first approach that allowed both additions and multiplications with constant-size ciphertexts. BGN hardness is based on the subgroup decision problem [18], which decides whether an element is a member of a subgroup G_p of group G of order $n = q_1 \cdot q_2$. In BGN, ciphertexts $c_1 = g^{m_1} \cdot h^{e_1}$ and $c_2 = g^{m_2} \cdot h^{e_2}$ encrypts m_1 and m_2 messages, where g and u are two random generators from G, $h = u^{q_2}$ is a random generator of the subgroup of G of order q_1 and random numbers $e_1$and e_2 from the set $\{0, 1, \ldots, n-1\}$.

The encryption of $m_1 + m_2$ is computed as

$$c_+ \leftarrow g^{m_1+m_2} \cdot h^{e_1+e_2+e} = \left(g^{m_1} \cdot h^{e_1}\right) \cdot \left(g^{m_2} \cdot h^{e_2}\right) \cdot h^e$$
$$= c_1 \cdot c_2 \cdot h^e = c_1 \ddot{+} c_2$$

Nonetheless, BGN is impractical due to it computes $c_\times$ only once using the bilinear map property, which maps $s : G \times G = G_1$, where G_1 is a group of order $n = q_1 \cdot q_2$.

Let $g_1 = s(g, g)$ and $h_1 = s(g, h)$, where g_1 is of order n and h_1 is of order q_1. Thus, there is α such that $h = g^{\alpha q_2}$.

The encryption of $m_1 \cdot m_2$ is computed as

$$
\begin{aligned}
c_\times &\leftarrow {g_1}^{m_1 m_2} \cdot {h_1}^{m_1 e_2 + e_2 m_1 + \alpha q_2 e_1 e_2 + e} \\
&= {g_1}^{m_1 m_2} \cdot {h_1}^{m_1 e_2 + e_2 m_1 + \alpha q_2 e_1 e_2} \cdot {h_1}^{e} \\
&= {g_1}^{m_1 m_2} \cdot {g_1}^{\alpha q_2 (m_1 e_2 + e_2 m_1 + \alpha q_2 e_1 e_2 + e)} \cdot {h_1}^{e} \\
&= {g_1}^{m_1 m_2 + \alpha q_2 (m_1 e_2 + e_2 m_1 + \alpha q_2 e_1 e_2 + e)} \cdot {h_1}^{e} \\
&= s(g, g)^{(m_1 + \alpha q_2 e_1)(m_2 + \alpha q_2 e_2)} \cdot {h_1}^{e} \\
&= s(g^{m_1 + \alpha q_2 e_1}, g^{m_2 + \alpha q_2 e_2}) \cdot {h_1}^{e} \\
&= s(g^{m_1} \cdot g^{\alpha q_2 e_1}, g^{m_2} \cdot g^{\alpha q_2 e_2}) \cdot {h_1}^{e} \\
&= s(g^{m_1} \cdot h^{e_1}, g^{m_2} \cdot h^{e_2}) \cdot {h_1}^{e} \\
&= s(c_1, c_2) \cdot {h_1}^{e} = c_1 \ddot{\times} c_2
\end{aligned}
$$

where $m_1 e_2 + e_2 m_1 + \alpha q_2 e_1 e_2 + e$ is uniformly distributed in $\mathbb{Z}_N$, and $c_\times$ is uniformly distributed encryption of $(m_1 \cdot m_2)$ *mod* n, but now in G_1 rather than G. However, BGN is still additively homomorphic in G_1.

Figure 1 presents a timeline of the most relevant inventions in the history of HE up before the first Gentry's FHE scheme in 2009 [3].

SHE approaches are proposed by Andrew Yao [19] in 1982, Tomas Sander, Adam Young, and Moti Yung (SYY) [20] in 1999, and Yuval Ishai and Anat Paskin (IP) [21] in 2007. They have several advantages and disadvantages concerning the number of operations, ciphertexts redundancy, processing efficiency, and vulnerability to attacks. In general, they are either insecure or impractical, but they are the basis for FHE.

2.3 Fully homomorphic encryption concept

Gentry [3] presents the first FHE scheme after 30 years of countless advances in the field when researches began with the invention of the public key cryptography in 1976 [22].

He builds a bootstrappable SHE with hardness based on the *Ideal Coset Problem* (ICP). The scheme can homomorphically evaluate its decryption function. The construction of the SHE uses the notion of the ideal in the lattice algebra. The ideal I in the ring $Z[x]/(f(x))$ with $f(x)$ of degree n satisfies $a + b \in I$ and $r \times a \in I$ for all $a, b \in I$ and $\in Z[x]/(f(x))$.

The scheme encrypts plaintexts m_1 and m_2 in ciphertexts $c_1 = (\psi_1)\ mod\ B_J^{pk}$ and $c_2 = (\psi_2)\ mod\ B_J^{pk}$, where $\psi_1 \leftarrow samp(B_I, m_1)$ and $\psi_2 \leftarrow samp(B_I, m_2)$ samples from the coset $I + m_1$ and $I + m_2$, respectively, and B_J^{pk} defines a secrete base for some ideal J in a ring R with a basis B_I of I, for relative primes I and J.

Encryption of $m_1 + m_2$ is computed as

$$
\begin{aligned}
c_+ &\leftarrow (\psi_1 + \psi_2)\ mod\ B_J^{pk} = (\psi_1)\ mod\ B_J^{pk} + (\psi_2)\ mod\ B_J^{pk} \\
&= c_1 \ddot{+} c_2
\end{aligned}
$$

and $m_1 \times m_2$ is computed as

$$
\begin{aligned}
c_\times &\leftarrow (\psi_1 \cdot \psi_2)\ mod\ B_J^{pk} = (\psi_1)\ mod\ B_J^{pk} \cdot (\psi_2)\ mod\ B_J^{pk} \\
&= c_1 \ddot{\times} c_2
\end{aligned}
$$

The bootstrapping procedure reduces the noise in the ciphertext and can be applied an unlimited number of times. Therefore, both aspects allow the construction of the first FHE scheme. See Section 3.2 for more information about bootstrapping.

Gentry's lattice FHE approach was promising, but it has several bottlenecks. The high computational cost and complex implementation make it unfeasible.

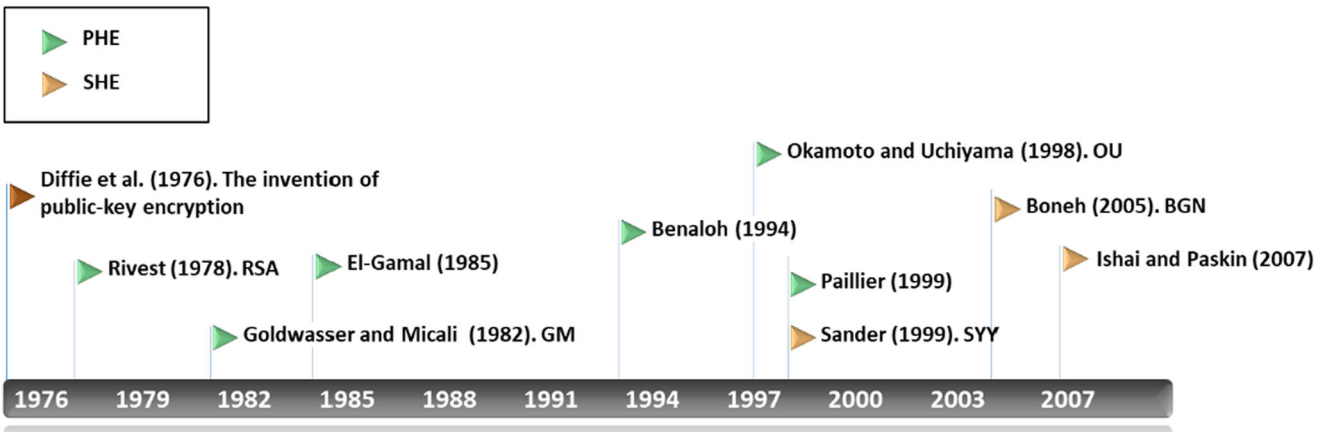

Fig. 1 Homomorphic encryption timeline

It becomes an object of optimization study and the basis of new approaches to solve performance and implementation problems [23–27].

After ten years, the advance in FHE is grouped into four main families: Ideal *Lattice-based,* over integers, Learning with Error based, and Nth-Degree Truncated Polynomial Ring Unit (NTRU)-based.

The first family works follow Gentry's original idea, whose hardness is based on the *lattice reduction problem*. The second family refers to those *integer-based* approaches [23, 25] where the hardness of the schemes is based on the *Approximate of Greatest Common Divisor (A-GCD)* problem [28]. The third family includes schemes based on *Learning with Error (LWE)* [27] and *Ring Learning with Error (RLWE)* [26, 29, 30], where both approaches are reducible to the lattice problems. Finally, the family of NTRU [31] and subsequent works [32, 33], also based on the lattice problem.

Figure 2 shows the timeline of relevant FHE approaches: Craig Gentry [3] in 2009, Craig Gentry and Shai Halevi (GH) in 2011 [4], Zvika Brakerski, Craig Gentry and Vinod Vaikuntanathan (BGV) [26] in 2012, Fan-Vercauteren variant of Zvika Brakerski's scale-invariant scheme (BFV) [29] in 2012, and Jung Cheon, Andrey Kim, Miran Kim, and Yongsoo Song (CKKS) [34] in 2017.

The continuous improvements and new approaches increase the efficiency and performance of FHE schemes. However, the contributions involve complicated designs, large keys, low computing efficiency, and high computing complexity.

The high overhead of performing additions and multiplication makes FHE impractical in real-world applications. To have a better understanding of the scopes and limitations of FHE schemes, we describe their most significant aspects in the next section.

3 Fully homomorphic encryption

The arrival of the first FHE scheme had a significant impact on the design of more secure systems, but not in their implementations. The high level of security in an FHE solution can enhance many technologies, e.g., outsourcing computing in cloud environments. But, the efficient implementation of the FHE is far away due to several limitations. This section introduces the formal definition of FHE and fundamental concepts, such as bootstrapping and key-switching.

The *privacy homomorphism* term was introduced by Rivest [35] to describe FHE formally; the main idea focuses on the arbitrarily computing of encrypted data without using the decryption key [3]. The concept of FHE does not involve the obfuscation characteristic, where a scheme is capable of hiding a sequence of l instructions called program P, $P = \{I_1, I_2, \ldots, I_l\}$. In such a way, given a plaintext input m and program P, $O(P) = \widehat{P}$ is an obfuscation transformation of P if only if $\widehat{P}$ and P have the same *observable behavior*. More precisely, if P fails or terminates with an error condition, then $\widehat{P}$ may or may not terminate; otherwise, $P(m) = \widehat{P}(m)$. For more detailed information and additional considerations on obfuscating transformations, refer to [36].

A third-party can process $\widehat{P}(m)$ without learning about $\widehat{P}$. The major disadvantage of the approach is the possibility to learn about the relation between m and $\widehat{P}(m)$. In contrast to FHE, where the third-party can process an encrypted version of $P(m)$ but cannot decrypt m and $\widehat{P}(m)$.

The general idea behind FHE is that the function f can be efficiently expressed as a circuit that processes homomorphically encrypted data, e.g., programs, mathematical operations, etc. [3].

FHE is considered a promising post-quantum tool [5]. Current public-key cryptography is based on the hardness of solving problems such as factoring or discrete logarithms. These widely studied problems are believed to be hard to settle

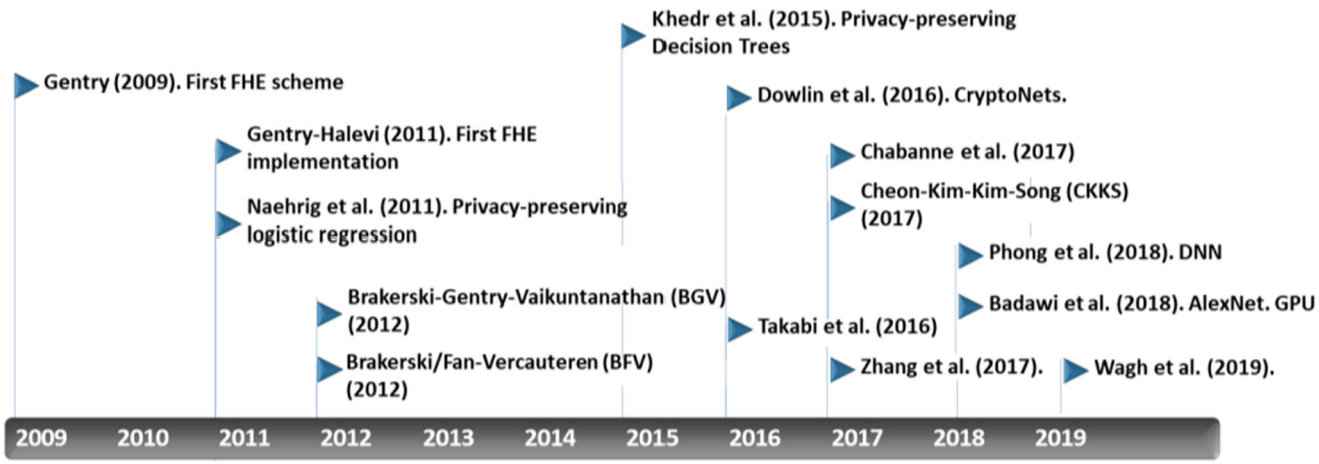

Fig. 2 Fully Homomorphic Encryption timeline

on a classical computer. However, an adversary equipped with a sufficiently large quantum computer can solve them easily. While the quantum computer does not exist today, its potential is considered a threat.

The essence of FHE is to produce the output of a ciphertext $f(c)$ for any desired function f and encrypted message c of plaintext m, as long as no information about c, $f(c)$, and m can be leaked. The function f can be efficiently computed. The expected operation of an FHE scheme ε as the classic black box model in computer systems is clarified by Fig. 3.

The challenging task is to find the appropriate mechanism $Evaluate_\varepsilon$ that satisfactorily leads to the output in a suitable time.

The following sections clarify concepts as $Encrypt_\varepsilon$ and $Evaluate_\varepsilon$ and describes additional fundamental elements, such as bootstrapping and key-switching.

3.1 Notation

Formally, an FHE scheme ε defines a conventional public-key scheme with four operations: $KeyGen_\varepsilon$, $Encrypt_\varepsilon$, $Decrypt_\varepsilon$, and $Evaluate_\varepsilon$ [3]. The computational complexity of all ε operations must be polynomial with respect to a security parameter λ, where:

- $KeyGen_\varepsilon$ takes λ as input and produces a public key pk and secret key sk as outputs, where pk maps from a plaintext space $\mathbb{P}$ to a ciphertext space $\mathbb{C}$ and sk in the opposite direction.
- $Encrypt_\varepsilon$ uses pk and a plaintext $m \in \mathbb{P}$ as inputs and produces a ciphertext $c \in \mathbb{C}$ as output.
- $Decrypt_\varepsilon$ defines the opposite process of $Encrypt_\varepsilon$, it receives sk and $c \in \mathbb{C}$ as inputs and outputs the plaintext $m \in \mathbb{P}$.
- $Evaluate_\varepsilon$ takes as input pk, a circuit $\delta \in \delta_\varepsilon$, and a tuple of ciphertexts $C = \langle c_1, \ldots, c_t \rangle$ that encrypt $M = \langle m_1, \ldots, m_t \rangle$ for the input wires of δ; it outputs a ciphertext $C' \in \mathbb{C}$, such that $Decrypt_\varepsilon(sk, C') = \delta(M)$.

Hence, given C that encrypt M, the desired functionality of $Evaluate_\varepsilon$ operation is to obtain the ciphertext $C' \leftarrow Evaluate_\varepsilon(pk, \delta, C)$ that encrypts $\delta(M)$ under pk, where $\delta(M)$ defines the output of δ on unencrypted messages of M. Additionally, the correctness and compactness properties are fundamentals in the formal definition of an FHE scheme.

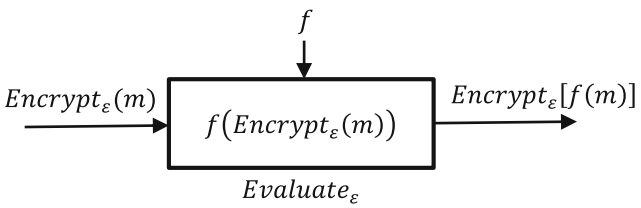

Fig. 3 Homomorphic encryption concept

They can be expressed using the four basic operations defined above.

Definition 1. Correctness A HE scheme ε is correct for circuits in δ_ε if, for any key-pair $(sk, pk) \leftarrow KeyGen_\varepsilon(\lambda)$, any circuit $\delta \in \delta_\varepsilon$, and any ciphertexts $C = \langle c_1, \ldots, c_t \rangle$ where $c_i \leftarrow Encrypt_\varepsilon(pk, m_i)$ for plaintexts $M = \langle m_1, \ldots, m_t \rangle$, it is the case that:

$$C' \leftarrow Evaluate_\varepsilon(pk, \delta, C), \text{ then } Decrypt_\varepsilon\left(sk, C'\right) = \delta(M) \quad (1)$$

Definition 2. Compactness A HE scheme ε is compact if there is a polynomial f such that, for every value of the security parameter λ, $Decrypt_\varepsilon$ can be expressed as a circuit D_ε of size at most $f(\lambda)$. Now, let ε be compact and also correct for all circuits in δ_ε, it is said that ε "compactly evaluates" δ_ε.

Definition 3. Fully Homomorphic Encryption A HE scheme ε is fully homomorphic if it compactly evaluates all circuits, i.e.:

$$Decrypt_\varepsilon(sk, Evaluate_\varepsilon(pk, \delta, C)) = \delta(M) \quad (2)$$

Since the first FHE scheme, the term of bootstrapping or "recrypt" function is fundamental. The next section highlights its importance and sketches the process.

3.2 Bootstrapping

The security of an FHE scheme resides in hiding the original message with a certain level of noise. Before the FHE introduction, the noise can be removed only by decryption; it limits the number of operations on the ciphertexts. The error grows with each homomorphic operation, and the decryption process is hopeless when the error reaches a threshold.

The notion of bootstrapping is introduced to maintain the error under the threshold. It allows creating the first FHE scheme based on a bootstrappable SHE scheme, i.e., a scheme able to homomorphically evaluate its own decryption function. The recrypt function is the core part of the bootstrapping procedure to reduce the noise in the ciphertexts. It can be applied an unlimited number of times to obtain fresh ciphertexts. So, recryption actions guarantee the correct decryption of the ciphertext after an unbounded number of operations.

In general, the recrypt function encrypts again the ciphertext (the plaintext now is double encrypted), removing the inner encryption by homomorphically evaluating the doubly encrypted plaintext using the encrypted secret key [37].

In other words, recryption refers to the process of executing $Evaluate_\varepsilon$ function on $Decrypt_\varepsilon$, i.e., $Evaluate_\varepsilon(pk_2, D_\varepsilon, \langle\langle \overline{sk}_1 \rangle, \langle \overline{c}_1 \rangle\rangle)$, where pk_2 defines the new public key, D_ε is the $Decrypt_\varepsilon$ function, $\langle \overline{sk}_1 \rangle$ is sk_1 encrypted under pk_2, and $\langle \overline{c}_1 \rangle$ the noisy ciphertext encrypted under pk_1 and pk_2.

Algorithm 1 defines a sufficient process to build an FHE scheme out of SHE; see [3] for more details.

Algorithm 1. Recrypt function in bootstrapping

Input: $pk_2, D_\varepsilon, \langle \overline{sk}_1 \rangle, c_1$
Output: c_2
$\langle \bar{c}_1 \rangle \leftarrow Encrypt_\varepsilon(pk_2, c_1)$
$c_2 \leftarrow Evaluate_\varepsilon\left(pk_2, D_\varepsilon, \langle \langle \overline{sk}_1 \rangle, \langle \bar{c}_1 \rangle \rangle\right)$

The bootstrapping is a homomorphic encryption scheme able to decrypt itself. The next section provides additional information about this characteristic.

3.3 Key-switching

A second secret key is fundamental in the bootstrapping process to encrypt/decrypt a ciphertext homomorphically. In algorithmic terms, bootstrapping can be defined as:

$$C' \leftarrow Encrypt_\varepsilon(pk_2,\ \ Decrypt_\varepsilon(sk_1, C)) \tag{3}$$

where the fresh ciphertext C' contains less noise than the original C. sk_1 and C are encrypted under a public key pk_1 and C' under pk_2. The encryption of sk_1 is usually referred to as bootstrapping key bk.

Key selection is a fundamental piece for the correct operation of the process; the quality in the selection and development of keys is directly proportional to the performance carried out by bootstrapping. There are two alternatives to define bk: encrypt the secret key sk_1 under itself $Encrypt_\varepsilon(pk_1, sk_1)$, or another key $Encrypt_\varepsilon(pk_2, sk_1)$.

In the self-encryption key sk_1, ciphertexts C and C' are encrypted under the same key, so the *circular security* [26] avoids the use of several keys.

In contrast, the *key-switching* alternative has the advantage of not requiring *circular security,* but it has to deal with multiple keys. A critical limitation of the *key-switching* approach is the number of available keys, i.e., n keys allow to achieve only a leveled homomorphism because they allow performing n bootstrapping operations.

C*ircular security with key-switching* is a combination of both approaches. It repeatedly uses a collection of n iterative keys: $sk_1, sk_2, \ldots, sk_n, sk_1 \ldots, sk_n \ldots$

The major disadvantage of the bootstrapping method is the computational cost. The overhead becomes the main drawback of the practicality of all FHEs. Most of the bootstrapping routines are complex and slow. Even with these limitations, researchers try to affront these disadvantages with techniques of high-performance, distributed, and parallel computing. All the circumstances make cloud computing ideal to receive the benefits of the FHE schemes.

The next section delves into a promising long-pursue application for privacy-preserving using homomorphic ciphers in cloud environments.

4 State-of-the-art

In this section, we review the latest advances in HE and Machine Learning-as-a-Service (MLaaS) fields. First, we highlight the main topics discussed in the published HE reviews. Later, we identify potential areas of opportunity, general limitations of MLaaS, introduce the latest approaches, and discuss open research directions in the area.

4.1 HE surveys

HE surveys consolidate significant contributions focusing on performance improvement, new approaches, applications, among others. They provide knowledge foundation and general panorama to researchers interested in applying and extending HE approaches.

Armknecht et al. [37] present the latest advances in the field and discuss relevant terminology and notions. They investigate fundamental concepts related to the implementation and development of HE, particularly in FHE. Naehrig et al. [38] exhibit the advantages of SHE in the medical, financial, and advertising domain; the authors implement a *proof-of-concept* based on *RLWE* to evaluate the efficiency and size of ciphertexts. Moreover, Archer et al. [39] list the benefits of FHE or SHE in real-world applications. The authors analyze its use in genomics, health, security, and education domains and present the significant importance of FHE.

Acar et al. [40] provide an exhaustive literature review and open research directions to essential contributions in the field. The survey gives the fundaments and future trends in the domain of HE systems. Martins et al. [41] present the topic from an engineering perspective. The last approaches in the field are analyzed and compared concerning performance and security. Vaikuntanathan [2] covers the development of HE for readers with knowledge in the mathematic field rather than for practitioners, similarly to Parmar et al. [42], Soitha and Shunmuganathan [43], and Gentry [44].

Aguilar-Melchor et al. [45] discuss the implementation of algorithms for signal processing in SHE. Hrestak and Picek [46] describes practical scenarios in the domain of cloud computing for HE. Moore et al. [47] propose an optimization hardware implementation solution of FHE.

Table 1 lists the main topics discussed in the HE reviews. It indicates a lack of specialized reviews focused on MLaaS-HE and NN-HE. This is one of the motivations of our work.

The next section presents recent advances in MLaaS-HE. We analyze contributions related to the design of ML modules for processing confidential information.

Table 1 Main topics of HE reviews

Topic Reference	Technical	Limitations	Applications	Tools	Cloud-based	Implementations
Vaikuntanathan [2]	•		•			
Armknecht et al. [37]	•	•	•	•		•
Naehrig et al. [38]	•	•	•		•	•
Archer et al. [39]			•		•	
Acar et al. [40]	•	•	•	•		
Martins et al. [41]	•	•	•			
Parmar et al. [42]	•		•			
Shunmuganathan [43]	•		•			
Gentry [44]	•	•				
Aguilar-Melchor [45]	•		•			•
Hrestak and Picek [46]	•			•	•	
Moore et al. [47]	•					•

4.2 MLaaS-HE

MLaaS refers to services in cloud computing for deploying machine learning tools [48]. It has emerged as a flexible and scalable solution for the training and prediction remotely. However, its most serious limitation is security and privacy concerns [49]. For example, prediction and classification models can involve extremely sensitive data: medical, advertising, financial, and behavioral, among others.

HE offers an elegant solution to the apparent paradox of security in the cloud. It allows a blind process of encrypted data in a remote server, i.e., the third-party does not learn anything about the input data and output. According to the notation in previous sections: given an HE scheme ε, a model α, and an input pattern p with its correspondent ciphertext $e \leftarrow Encrypt_{\varepsilon}(p)$, the scheme ε returns a ciphertext s with the evaluation of the model α on the input pattern p, where $Decrypt_{\varepsilon}(s) = \alpha\,(p)$.

Under this assumption, many machine learning models have tried to implementing cryptographic systems for the prediction and classification of confidential information using homomorphic ciphers [50]. For instance, in health, to process encrypted patient data and obtain an encrypted diagnosis.

Naehrig et al. [38] consider the training of a Logistic Regression (LR) model with data protection during the regression coefficients generation. The model enables an efficient message encoding of an approximate polynomial of degree N-1.

Khedr et al. [51] implement Bayesian filters and Decision Trees (DT) for encrypted data using an FHE scheme. The classification model supports ciphertexts multiplication without require key-switching.

Several researchers focus their contributions on HE frameworks capable of enriching the MLaaS paradigm and designing an efficient environment for the arbitrary evaluation of complex NNs over encrypted data.

Dowlin et al. [52] propose CryptoNets to address the challenge of achieving a blind non-interactive classification. The NN uses a SHE scheme for inputs and propagates signals across the network homomorphically. Its performance is limited due to the replacement of the sigmoidal activation function and the computational overhead. Several subsequent works in the literature focus on improving its constraints.

Chabanne et al. [53] solve the limitations of CryptoNets by adding a normalized layer before each activation layer. The implementation is the first to enable a homomorphic evaluation of Deep Neural Networks (DNN). The NN achieved an accuracy similar to the best non-secure versions through an FHE scheme.

Hesamifard et al. [54] develop CryptoDL to prove the possibility to find a lowest degree polynomial approximation of an activation function within a specific error range. Rectified Linear Unit (ReLU), Sigmoid, and hyperbolic Tangent (Tanh) functions are approximated by polynomials.

Badawi et al. [55] present a Convolutional Neural Network (CNN) for image classification with FHE properties on Graphics Processing Units (GPU). The AlexNet accelerates the classification process and maintains security and accuracy; it classifies the MNIST dataset in 1% of the time CryptoNets takes.

Zhang et al. [56] propose a privacy-preserving deep learning model for big data learning in a cloud computing environment. The model is trained with a back-propagation algorithm and uses the BGV homomorphic scheme [26].

Brutzkus et al. [57] develop Low-Latency CryptoNets (LoLa) to improve latency and memory usage over its predecessors. While CryptoNets encodes each image's feature as a separate message, Lo-La encrypts the input vector as a single

message. This modification allowed evaluating the same network in a pair of seconds.

Takabi et al. [58] consider decentralized scenarios with distributed datasets across multiple parties. The NN uses a polynomial approximation as an activation function. Nonetheless, the recryption process is carried out directly by the client because the implementation has an interactive approach.

Phong et al. [59] introduce an asynchronous Stochastic Gradient Descent (SGD) on a DNN with an additively HE. The approach holds the accuracy with an acceptable increase of the overhead considering the conventional deep learning systems.

Wagh et al. [60] develop the building blocks for a novel protocol of NN with a secure three-party. The model enables the training and inference of several NNs architectures without learning about the data.

There is a key limitation of current HE schemes. They cannot support division operations and comparisons, such as the test of equality/inequality. Number comparison and sign determination are essential operations for implementing cryptographic algorithms in MLaaS [61]. As a consequence, without their substantial development, practical adoption is bounded [62].

One strong direction focuses on designing approximate methods to address these limitations. Babenko et al. [61] introduce a numerical comparison technique in the Residue Number System (RNS) without requiring resource-consuming non-modular operations.

Table 2 presents a comparison of MLaaS-HE related works.

It emphasizes objectives, operational characteristics, approaches, and implemented schemes. Many studies analyze efficiency, additionally to security. A higher level of security increases the use of computational resources, and therefore, implies less efficiency.

The literature review exposes three main challenges of NN-HE research and development:

a. *Low efficiency*. Significant optimization is required in the practical implementation of NN-HEs. High complexity operations, such as bootstrapping and large encrypted messages, are the main sources of low efficiency.
b. *Reduced number of primitives*. Technical characteristics limit the applicability of NN-HE schemes in real-world applications. A major number of operations, increased multiplicative depth, efficient number comparison (sign detection) can expand the adoption of privacy-preserving systems.
c. *Real-world applications*. The HE schemes are being used even with problems of performance and the limited number of primitives. ML models can train, predict, and classify confidential information using HE schemes.

In the next section, we focus on relevant aspects of privacy-preserving neural networks.

5 Privacy-preserving neural networks

NNs are achieving remarkable results and are extensively used in multiple domains. However, their implementation can be difficult for inexperienced users. The training process with relatively big datasets can consume many resources and time.

Cloud services can make it easier. The models can be implemented, trained, and deployed on third-party infrastructures with a relative facility [69, 70].

The use of a third-party infrastructure reduces the problems of resources and complexity but introduces privacy issues of sensitive information [49, 71].

The access of the NN to the raw data can create potential privacy risks. This section presents solutions for solving the apparent paradox by allowing encrypted data to be blindly processed by a remote server.

This section addresses the importance and shortcomings of the homomorphic processing of NNs. The idea is to explain the difficulties of achieving an efficient blind non-interactive classification. First, we provide basic notions in the field of NNs. Later, we present relevant aspects of the implementation of privacy-preserving NN via HE (NN-HE), focusing on the techniques for constructing NN-HE models.

5.1 Preliminaries

In the last decades, ML emerged as an essential topic in Artificial Intelligence (AI). Moreover, ML algorithms based on NNs are the primary research potential line. They provide suitable solutions in a wide field of human knowledge. Its massive adoption and generalized use reveal several possible data privacy problems in the construction of a model. An underlying notion of NN is required to understand its issue of privacy.

NN is a computing model system that mimics the behavior of a biological brain. It attempts to identify (to learn) underlying relationships in the information provided by a dataset. The architecture of NN groups a population of neurons in layers and defines the connection between them. The neurons are basic units to process information coming from the external world.

Each neuron consists of n_I inputs $x = (x_1, \ldots, x_{n_I})$ and an output $f(y)$, where $y = \sum_{i=1}^{n_I} w_i \cdot x_i + \beta$. The value of y defines a weighted sum of the inputs considering the weights $w = (w_1, \ldots, w_{n_I})$ and a bias β.

The non-linear activation function f generates the final output of the neuron. The definition of w and f are fundamental problems of NN.

Table 2 Main properties of MLaaS-HE

Reference	Year	Operations			ML					Schemes				Party		Objective	
		Addition	Multiplication	Other	Logistic regression	Neural Networks	Deep Neural Networks	Decision Trees	Naive Bayes	(R) LWE	Integer-based	NTRU	Ideal Lattice-based	Two	Multi	Security	Efficiency
[6]	1978		•											•		•	
[7]	1985		•											•		•	
[12]	1999	•												•		•	
[3]	2009	•	•										•	•		•	
[38]	2011	•	•		•					•				•		•	
[31]	2014	•	•									•		•			•
[63]	2014	•	•		•						•			•		•	
[50]	2015	•	•					•		•				•		•	
[51]	2015	•	•					•		•				•		•	
[58]	2016	•	•				•				•				•	•	•
[52]	2016	•	•			•					•			•		•	
[64]	2016	•	•								•			•		•	
[65]	2016	•	•		•					•				•		•	
[56]	2016	•	•				•				•				•	•	
[5]	2017	•	•							•				•		•	•
[53]	2017	•	•				•				•			•		•	
[54]	2017	•	•				•				•			•		•	•
[66]	2018	•	•		•					•				•		•	
[59]	2018	•					•				•				•	•	•
[67]	2018	•	•				•				•			•		•	
[55]	2018	•	•				•				•			•		•	•
[16]	2018	•	•				•				•			•		•	
[68]	2019	•	•						•	•				•		•	
[57]	2019	•	•			•					•			•		•	•
[60]	2019	•	•			•					•				•	•	
[61]	2019	•	•	•										•		•	•

The HE version of a neuron substitutes operators + and × by $\ddot{+}$ and $\ddot{\times}$, respectively. The evaluation of the HE-neuron requires the encrypted values of x, w, and β, such that:

$$\overline{c}_0 \leftarrow \ddot{f}\left(\sum_{i=1}^{n_I}\left(\overline{w}_i \ddot{\times} \overline{x}_i\right) \ddot{+} \overline{\beta}\right) \tag{4}$$

where $\overline{x}$, $\overline{w}$, and $\overline{\beta}$ are the corresponding ciphertexts of x, w, and β, and $\ddot{f}$ is the homomorphic version of f, for instance, a polynomial approximation that only consists of operations $\ddot{+}$ and $\ddot{\times}$. $\overline{c}_0$ contains the encrypted output of the neuron computation, it guarantees the privacy of the result even if $\overline{c}_0$ is disclosed.

The interaction between neurons is essential for NN performance, the network structure defines the interaction between layers, subsets of grouped neurons. In the network sequence, the layer position establishes a specific role in the processing of data. Typically, NN is composed of three types of layers. The first layer is the input layer. It receives information from outside the network. Internal layers, also called hidden, are not directly accessible from the exterior. The last segment in the NN is the output layer. It transfers information outside of the network.

Neurons can be connected to the neurons of the same layer (self-recurrent), a successive layer (feed-forward), or a previous layer (feedback/recurrent). Hence, a NN could be a recurrent system, as opposed to the purely feed-forward ones, where each neuron is evaluated only once. Throughout this manuscript, we consider feed-forward networks because they are widely used, easier to understand, and simpler to evaluate homomorphically.

The associate weight of a path between two neurons defines the importance of the input in the neuron. Moreover, the degree of connection establishes the number of inputs in the neuron. The most widely used architectures are fully connected, convolutional, max pooling, and mean pooling. The NN-HE does not apply any modification in the structure of the NN.

The selection of an activation function is an essential element in the construction of an effective NN model. It determines the reaction of a neuron to the inputs and information forwarded to the following layers. The most common activation functions are step, sign, sigmoid, ReLU, and Tanh [72]. They contribute to the underlying relationships in a set of data by the learning process.

The definition of an adequate $\ddot{f}$ is an open problem. Standard activation functions use operations not supported by HE, so to find cryptographically compatible replacement functions is necessary to operate over encrypted data. Related works provide several examples of approximate activation functions for multiple domains.

5.2 Homomorphic training of neural networks

The training process consists of developing a mapping from the input to the output space based on the modification of the weights w of each neuron. The network is able to learn and generalize information based on several examples. The presence of an external entity that controls the learning process is defined as Supervised Learning (SL). Meanwhile. The absence of the entity is denoted as Unsupervised Learning (UL).

In SL, the supervisor adjusts w when the expected output $\widehat{\vartheta}$ and actual output ϑ are different. The loss function $L\left(\widehat{\vartheta}, \vartheta\right)$ measures the error between both outputs. In general, the training process is the following: 1) the network receives an input pattern x, 2) it calculates the output $\vartheta = Net(x)$ by feeding x forward and performing all the computations until the output layer; 3) the network computes $L\left(\widehat{\vartheta}, \vartheta\right)$, and 4) it modifies the weights to reduce the error. Finally, 5) the network repeats the process several times for all the samples in the dataset. The aim is to find a collection of weights that minimize that error.

The NN should provide correct answers after the training process, even in the presence of patterns that are not used to train the model. In the testing phase, the NN is evaluated under a set of examples distinct to the training phase samples. The idea is to avoid the overfitting of the model and measure its efficiency with new information. The general process is an analogy of the human brain evolution during a person's life.

The training consists of computationally intensive tasks, even for non-HE models. Training a NN itself involves a large number of operations toward finding the collection of weights that minimize $L\left(\widehat{\vartheta}, \vartheta\right)$. With HE, it becomes more challenging even with advanced technologies.

In the HE domain, the training process implies large encrypted messages and several bootstrapping executions. Its computational complexity is in orders of magnitude of the unencrypted training. For instance, the computational cost of seven-layer CNN training is around one hour with a conventional CPU, while to train the same CNN with HE requires around a year [73]. Hence, training DNNs that can contain tens, hundreds, or sometimes thousands of layers is impractical.

Two options are common to deal with the bootstrapping execution during NN-HE training: its acceleration and exclusion.

High-performance, distributed, and parallel computing provide tools for training over large encrypted datasets. The main tendencies are to use hardware accelerators such as high throughput computing units (GPU, FPGA, etc.) and customized chips (ASIC).

These technologies significantly reduce runtime and can make them comparable to similar unencrypted versions. The use of cloud computing is a cornerstone in this direction.

The first approach of avoiding bootstrapping operations focuses on their substitution by decrypting the ciphertext

inside a secure entity (client-server, secured HPC computers, etc.).

For instance, Takabi [58] presents the first homomorphically trained NN-HE using an interactive approach, where the decryption and encryption process must be carried out directly by the client. It is a hybrid model between Secure Multi-party Computing (SMC) and HE.

The second approach focuses on pre-trained NN-HEs, where the weights of an already trained NN are public. To avoid the overhead of the training phase, it is performed over unencrypted data keeping the evaluation on the encrypted data. It is widely used in current practice.

5.3 Homomorphic evaluation of neural networks

The training and testing processes of NNs do not limit access to the raw data. Consequently, legal and ethical requirements may prevent cloud-based solutions for many applications [74, 75]. For example, the processing of DNA sequencing remains a complex task, but now it is faster and less expensive due to the methods developed over the last two decades [76].

Two opposing considerations should be evaluated. On the one hand, sharing and processing the genetic sequence with complex NN models could offer much value, such as disease diagnosis. On the other hand, from a privacy perspective, when a person shares his DNA sequence, his siblings or children's DNA is partially shared; it falls into severe problems if such information reaches than unreliable third-parties. We can find analogies of this example with medical, financial, or behavioral data.

The above problem detonated the interest of evaluating arbitrarily complex NNs over encrypted data. One strong direction focuses on the benefit of HE schemes for the processing of information in NN. NN-HE is a natural extension of NN models. The methodology consists of applying HE to the network inputs and propagating the signals across the network homomorphically. The training and inference phases are fundamental in the process of NN with privacy-preserving.

In the inference phase, the model's dimensions are known beforehand, so the number of operations can be estimated apriori.

From a HE perspective, the network represents a leveled circuit where levels are called layers. The cryptosystem allows implementing an encryption scheme with a predefined noise budget, avoiding bootstrapping or any recrypt function. In other words, a leveled HE scheme is enough for the inference phase since the amount of noise supported by the ciphertext is known, and the polynomial functions have a fixed maximal degree on the encrypted data.

Nonetheless, deep learning needs an FHE scale-invariant scheme because it implies many hidden layers in the network. Therefore, a large amount of noise has to be controlled by bootstrapping or any recrypt function. Moreover, only polynomial functions can be computed due to the HE schemes supports only additions and multiplications.

The construction of NN-HE involves two challenges: a computational design of the homomorphic processing of inner network functions and low-degree polynomials manipulation. The following sections delve into both challenges and solutions proposed in the literature.

5.4 Homomorphic neurons

NN-NE design involves several aspects of efficient implementations. For example, each neuron performs a *weighted-sum* y and non-linear activation function $f(y)$. With the HE scheme, multiplication operation is slower and adds large amounts of noise. A bootstrapping operation should be performed when a ciphertext contains too much noise.

The computing of *weighted-sum* consists of a set of additions and multiplications between known constant weights and ciphertexts. The processing of y can be improved since one of the operands is plaintext and the size of the resultant ciphertext remains the same as the input ciphertext.

Bos et al. [77] encryption scheme can help to clarify this point. It maps plaintext messages from the ring $R_t^N = \mathbb{Z}_t[x]/(x^N+1)$ to the ring $R_q^N = \mathbb{Z}_q[x]/(x^N+1)$. The encryption scheme chooses random polynomials $f, g \in R_q^N$, $f = (tf' + 1) \bmod q$. The public key h is defined as $(tgf^{-1}) \bmod q$, while f is the secret key. Since not every element in R_q^N is invertible, these steps are iterated until the corresponding f has an inverse and h can be computed. Encryption of a message $m \in R_t^N$, can be defined as $c = \left[\left\lfloor \frac{q}{t} \right\rfloor (m) \bmod t + e + hs\right] \bmod q$, where e and s are random noise polynomials in R_q^N with coefficients of small absolute value. Decrypting is done by computing $m = \left\lfloor \frac{t}{q} \cdot (f \cdot c) \bmod q \right\rfloor \bmod t$. Here the product $f \cdot c$ is first computed in R_q^N, the coefficients are interpreted as integers, scaled by t/q, and rounded to the nearest integers. Finally, they are interpreted as modulo t.

To compute an operation of addition or multiplication between plaintext and ciphertext there are two alternatives: 1) The naive process is computationally intensive and increments the noise. It encrypts the constant and then executes the operation, and 2) an optimized process encrypts the constant without noise and performes the standard homomorphic operation. Let $c = \left[\left\lfloor \frac{q}{t} \right\rfloor \cdot (m) \bmod t + e + hs\right] \bmod q$ be the encrypted message and w the plaintext. Addition can be achieved by multiplying w by q/t, and the result is adding to c, which produces

$$c_+ \leftarrow \left[\left\lfloor \frac{q}{t} \right\rfloor \cdot (m + w) \bmod t + e + hs\right] \bmod q$$

$$= \left[\left\lfloor \frac{q}{t} \right\rfloor \cdot (m) mod\ t + \left\lfloor \frac{q}{t} \right\rfloor \cdot (w) mod + e + hs\right] mod\ q$$

$$= \left[\left\lfloor \frac{q}{t} \right\rfloor \cdot (m) mod\ t + e + hs\right] mod\ q + \left[\left\lfloor \frac{q}{t} \right\rfloor \cdot (w) mod\right] mod\ q$$

$$= c \ddot{+} w$$

Homomorphic multiplication is done similarly.

The processing of *activation functions* in HE is more complicated because most of the functions are not polynomials, such as sigmoid and ReLU functions. Some approaches replace standard functions with the non-linear low-degree square function [52, 57]. Nonetheless, the square function's unbounded derivate induces a strange behavior during the training phase with more than two non-linear layers [53].

Multiple approaches address the limitation of activation function represented by polynomial approximation through Taylor series, Chebyshev polynomials, among others [53–55, 58, 78–81]. Since all NN inner functions are continuous, the network can be viewed as a continuous function. If the domain (the input space) is a compact set, then the Stone-Weierstrass theorem [82] establishes that it can be approximated uniformly by polynomials. The challenge is to approximate NN with polynomials of the lowest possible degree.

Let *Net*() be a network with l layers. If a polynomial of degree d approximates the activation function in each layer, then the polynomial approximation of *Net*() will be a polynomial of degree d^l. Hence, to achieve a low degree polynomial, d and l must be small. Minimizing d is a standard exercise in approximation theory. Even so, optimizing l implies reducing the layers in *Net*(); this goes against the deep learning current trend.

5.5 Data manipulation

Data representation and manipulation in the network is another challenge of the construction and implementation of NN-HE. Each ciphertext c has some noise that hides the message. Hence, $c = m + e$, where m represents the original message and e defines the noise. Regardless of the data nature to be submitted in these computational models, their encryptions are entries in a polynomial ring.

Mapping integers, floats, binaries, or another number system to polynomials is a standard and widely studied exercise. For example, a conventional mapping technique for integers uses their binary representation; the binary expansion of an integer generates the polynomials' coefficients. However, it is still necessary to achieve a low-degree polynomial representation.

Multiple approaches provide low-degree polynomial representation by applying the Chinese Remainder Theorem (CRT) [52, 54, 55, 57, 83], where k primes $q_1, \ldots, q_k$ and a polynomial $\sum a_i x^i$ define k polynomials in such a way that the *jth* polynomial is $\sum[(a_i)\ mod\ t_j]x^i$.

CryptoNets [52] and AlexNet [55] follow the CRT approach with polynomials of a high degree. They implement a Single Instruction Multiple Data (SIMD) technique where a single polynomial encodes a feature of multiple instances. For example, if the input data are images with a dimension of 28 × 28 pixels, it generates 784 ciphertexts (one per pixel) where the number of images determines the ciphertexts' size. The constructions use the CRT to break the ciphertexts into multiples chunks and process them in parallel. The main contribution of CRT is decreasing the processing time of high degree polynomials.

The SMID approach has several limitations of latency and memory. A single prediction implies many operations since each feature represents a message, and the high number of messages generates memory bottlenecks. It makes unfeasible its implementation in DNN models. The biggest disadvantage is inefficiency. The consumption of computational resources is the same in processing one or a thousand images.

Lo-La [57] employs an alternative representation to reduce the latency and memory usage of its predecessors. It encodes an input vector v of d dimensions as a single message m where $m_i = v_i$, for $i = 1, \ldots, d$. The private prediction of a linear classifier on v requires only a single message and $O(\log d)$ operations, in contrast to the d messages and $O(d)$ operations of previous systems for the same classifier. However, the scheme is still dependent on the dimension of the message, and therefore, impractical for real-world applications with high-dimension data.

In a nutshell, CryptoNets and Lo-La contributions are the foundations of many types of solutions. Consequent works combine different techniques for the reduction and manipulation of polynomials such as CRT and SIMD, among other methods. Besides, the evolution towards an NN-HE friendly model suggests incorporating GPUs, Field-Programmable Gate Arrays (FPGA), and Application-Specific Integrated Circuits (ASIC). We are dealing with a nascent research area with innumerable potentials and commensurable benefits.

5.6 Noise and errors

As shown in previous sections, sometimes HE uses the concepts of noise and error as interchangeable notions. In this section, we differentiate them from each other for better understanding. Broadly, the noise is information injected into data for the generic construction of a public-key cryptosystem. In contrast, the error occurs as part of a rounding error occurring during approximate computations. This section presents the noise and error notions in the context of ML, specifically in NN models.

5.6.1 Noise

The design of NN-HE has to take into account the number of arithmetic operations (addition and multiplication) necessary for its implementation. In BFV [29] and CKKS [34] schemes, the quantity of additions is unlimited, but the amount of multiplications is limited.

The multiplicative depth is the maximal number of homomorphic multiplications, which can be performed on ciphertexts to retrieve the result of these multiplications correctly. The computational complexity of each multiplication and size of the public key depends on the multiplicative depth. Both parameters are increased with increasing depth.

As practice shows, the volume of noise depends on two factors: level keys and implementation.

- *Level keys*. The upper limit of the multiplicative depth relies on the selected level keys. For example, the PALISADE library offers an algorithm for selecting level keys to change the amount of noise that cannot be removed without decryption or bootstrapping.
- *Implementation*. The maximum number of the multiplicative depth also depends on the algorithm implementation [84]. For instance, the theoretical boundary for multiplicative depth in Halevi et al., [85] and Bajard et al., [86] schemes is the same, but the multiplicative depth of [85] is larger than [86] in practice. Also, the approximate calculations of implementation [85] increase the noise in comparison with exact calculations.

Gentry [3] proposes a mechanism to remove excessed noise through the bootstrapping procedure (See Section 3.2), which can be roughly construed as re-encryption or decryption in encrypted form. Some noise can only be removed with decryption. The amount of noise after the bootstrapping process is greater than in the original ciphertext for FHE over integers [87].

Considering the great computational complexity of bootstrapping, Badawi et al. [55] use a scheme that does not require bootstrapping to implement CNN.

A general NN model consists of mathematical convolutions and activation functions. In the best case, the multiplicative depth of the mathematical convolution is equal to one. For the realization of activation, the function is approximated by a polynomial, as HE supports only addition and multiplication.

In the worst case, the multiplicative depth is equal to the binary logarithm of the polynomial plus one. To reduce the multiplicative depth in the polynomial calculating, various techniques are used based on function compositions [88].

The correctness of the result in NN-HE depends on two factors described in the next section.

5.6.2 Errors

Errors in data processing can compromise the correctness of the results. The implementation of NN-HE distinguishes two error classes: algorithmic errors and running errors.

Algorithmic errors. The representation of real numbers with integer polynomials can lead to an incorrect result. An error can occur in numbers in the vicinity of zero, even without adding noise. Converting a value vector to polynomial $m(X)$ can lead to a severe error in calculations. For example, let us consider the CKKS [34] scheme:

Let M-th cyclotomic polynomial with $M = 8$ (i.e. $\Phi_8(X) = X^4 + 1$), a scaling factor $\Delta = 64$, and $T = \{\xi_8, \xi_8^3\}$ for the root of unity $\xi_8 = \exp(2\pi i/8)$. For a given vector $z = (0.1, -0.01)$, its corresponding real polynomial is $-0.039X^3 + 0.039X + 0.045$ according to the interpolation polynomial in the Lagrange form for a given set of points $(\xi, 0.1)$, $(\xi^3, -0.01)$, $\left(\overline{\xi}, \overline{0.1}\right)$, $\left(\overline{\xi^3}, \overline{-0.01}\right)$, where $\overline{a + bi} = a - bi$, $a, b \in \mathbb{R}$ and $i^2 = -1$. Then the output of the encoding algorithm is $m(X) = -2X^3 + 2X + 3$. Note that $64^{-1} \cdot \left(m(\xi_8), m\left(\xi_8^3\right)\right) \approx (0.09107, 0.00268)$ is approximated to the input vector z with high precision.

The example shows that the encoding number -0.01 turned into 0.00268 when it is decoded. The number obtained during decoding differs in the value and sign, i.e., it does not carry any information about the initial number. Additionally, in this case, increasing Δ allows reducing the absolute value.

As a consequence, an incorrect result using HE with NN is highly probable when the input data are normalized, i.e., the values are compressed up to the interval [0, 1]. Moreover, *this kind of error* leads to incorrect results when using unstable algorithms [89].

Running errors. The errors of calculations can arise as a result of the polynomial approximation of activation functions of NNs. For instance, let us consider the function $\mathrm{ReLU}(x) = \max(x, 0) = (\mathrm{sgn}(x) + 1) \cdot x$.

The function $\max(a, b)$ returns the maximum value of a and b, where $\mathrm{sgn}(x)$ is a sign function. When we calculate $\mathrm{ReLU}(x)$, for $x < 0$, implemented by HE, with the $\mathrm{sgn}(x)$ algorithm presented in [88], the function will be greater than zero. Thus, the implementation of NN-HE should be considered rounding errors that may occur.

6 Applications and tools

In this section, we outline NN-HE implementations in real-world applications and show current tools for its development. First, we illustrate the advances and limitations of HE libraries and highlight NN frameworks. Later, we present a literature

analysis to describe the growing interest in the field, the breadth of domains that can benefit from it, and emerging trends.

6.1 NN-HE tools

High-quality implementations should complement theoretical research. Industrial and academic groups released several HE libraries in recent years: SEAL, HElib, TFHE, PALISADE, cuHE, HEAAN, HE-transformer, etc. Many implementations are based on RLWE and contain common choices for the underlying rings, error distributions, and other parameters.

Simple Encrypted Arithmetic Library (SEAL) [90] is the most used open-source HE tool supporting BFV and CKKS schemes. It is implemented in C++ with active developments for other languages C#, F#, Python, and JavaScript. SEAL can compress data to achieve significant memory footprint savings.

Homomorphic-Encryption Library (HElib) [91] is an open-source library based on the BGV scheme and developed in C++. It focuses on the effective use of ciphertext packing and data-movement optimizations. One disadvantage of HElib is limited bootstrapping performance.

Faster Fully Homomorphic Encryption (TFHE) [24] is an open-source library sustained in a Ring-variant of the GSW and an alternative representation over the torus. The library in C/C++ implements a very fast gate-by-gate bootstrapping procedure; this mode has no restriction on the number of gates or their composition.

PALISADE [92] is an open-source project for the implementation of a HE library with support to BGV, BFV, CKKS, FHEW, and THEW schemes. It was developed with C++ and provided an extension for multi-party. PALISADE takes advantage of RNS algorithms to achieve high performance.

CUDA Homomorphic Encryption (cuHE) [93] is a GPU-accelerated library implemented in C++ for the parallel platform CUDA. Arithmetic functions adopt the CRT, Number-Theoretic Transform (NTT), and Barrett reduction to handle with large polynomial operands.

Homomorphic Encryption for Arithmetic of Approximate Numbers (HEAAN) [34] is a library with supports for the CKKS scheme. It is implemented in C++ with support to fixed-point arithmetic. The approximate operations of rational numbers generate an error that depends on configurable parameters.

Homomorphic Encryption ***transformer for nGraph*** (HE-**transformer**) [94] is a HE project for Intel nGraph Compiler based on SEAL; the implementation in C++ provides a graph compiler for NN. The project is a proof-of-concept to measure the performance of HE schemes for deep learning.

The selection of a specific HE library is a step that requires knowledge of its advantages and disadvantages. Table 3 compares the most common general-purpose HE libraries with their pros and cons. It describes crucial key features for rapid adoption and further development. The selection of an adequate approach of the privacy-preserving tool should consider these essential features.

On the other hand, an important decision in the NN-HE implementation is choosing a framework to deal with the NN model. Table 4 presents the characteristics of popular ML frameworks. Our goal is to emphasize those high-level tools used to compute homomorphic operations. Core language is the primary library language, but most of them use more than one to build the framework. Bindings define official interfaces to support other languages and libraries for different activities: feature extraction, training, etc. GPU indicates the availability of CUDA for GPU computation, a crucial feature for training and testing modern CNNs.

According to Table 4, several works employ toolboxes such as Caffe, Dlib, Pytorch, TensorFlow, and Theano on researches related to CNN over encrypted data. Moreover, it underlines the GPU's use as an essential capability for HE computation. Also, it highlights C++ and Python as key languages in model building.

These efforts towards NN-HE are either insecure or impractical. They suffer from the limitations mentioned in previous sections. They are based on PHE cryptosystems [108, 110], compute plaintext and simulate ciphertext performance [99], introduce NN models without multi-party computation support [112], propose interactive NNs [110], or use the mathematical properties of "homomorphism" to NN training, not HE itself [96]. Moreover, all of them use high-level toolboxes solely for pre-process and pre-train NN models.

6.2 Literature analysis

The pros and cons of HE libraries do not describe their current study or research. So, an important topic is the analysis of their trends in the field. Moreover, the study of the applications mostly addressed by the research community. A proper and instrumented overview for NN-HE development is of utmost importance. In this section, we start with a method for retrieving relevant works, then we analyze the most prominent terms and, finally, touch upon several emerging trends in the HE domain.

Initial evidence of the interest in HE topics is academic publications' growth in the last fifteen years. Among the vast number of publications, we distinguish the following two types of papers,

a. research publications suggesting new HE approaches, or extending and applying an existing one,
b. dissertations and thesis addressing HE cryptosystems.

For databases querying, we consider three popular databases as publication sources: ProQuest Dissertations and

Table 3 Comparison of commonly general-purpose HE libraries across their pros and cons

Tool	Pros	Cons
SEAL	Well-documented. Easy security parameters setting.	Poor flexibility. Limited number of supported schemes.
HElib	Efficient homomorphic operations.	Low bootstrapping performance. Complicated security parameter setting.
TFHE	Fast bootstrapping.	Poor performance for simple tasks.
PALISADE	Multiple HE schemes. Cross-platform.	Poor documentation and support.
cuHE	Parallelism and high memory bandwidth of GPUs.	
HEAAN	Operations between rational numbers.	
HE-transformer	Integration with deep learning libraries.	Extension of SEAL.

Theses (PQDT) database, Web of Science (WoS) database, and Google Scholar. Like in many fields, in HE literature, the same notion can be named by multiple terms. To avoid misinterpretations, we consider most of these terms in the database querying, which generally holds in papers.

For PQDT, we performed searches of thesis from the corresponding query:

– query "(homomorphic) (encryption OR encrypt OR cipher OR encode)". We select "since 2005" and "since 2013" and take all works for each result.

For WoS, we merged the results from the following queries (options "Search by topic" and "All collections" were enabled):

Table 4 Machine learning frameworks features

Ref	Tool	Core language(s)	Binding(s)	GPU	OpenMP	Mobile devices	HE implementation
[95]	Caffe	C++	Python, Matlab	•	•		[96]
[97]	Chainer	Python		•			
(Convetjs, https://github.com/karpathy/convnetjs)	Convetjs	JavaScript					
(DL4j, https://deeplearning4j.org/)	DL4j	C++, Java	Java, Scala, Clojure, Python, Kotlin	•	•		
[98]	Dlib	C++	Phyton	•	•		[99]
(DSSTNE, https://github.com/amzn/amazon-dsstne)	DSSTNE	C++		•			
[100]	Flux	Julia		•			
[101]	H20	Java	Python, R, Scala				
[102]	Keras	Python	R	•	•		[103]
[104]	Deep Learning Toolbox	C, C++, Java, MATLAB		•			
[105]	CTNK	C++	C#, Python.	•	•		
[106]	Mxnet	C++, Python	Scala, Julia, Clojure, Java, C++, R and Perl.	•	•		
(Neon, https://github.com/NervanaSystems/neon)	Neon	Python		•			
(OpenNN, https://www.opennn.net/)	OpenNN	C++		•	•		
(Paddle, https://github.com/PaddlePaddle/Paddle)	Paddle	C++	Python	•		•	
[107]	Pytorch	Python, C/C++	Java	•	•	•	[108]
[109]	TensorFlow	Phyton	C++, R, JavaScript.	•		•	[110]
[111]	Theano	Python		•	•		[112]
[113]	Torch	Lua	C/C++, LuaJIT	•	•		

- query "(homomorphic) (encryption OR encrypt OR cipher OR encode)". We select "since 2005" and "since 2013" and take all papers for each result, including their citation statistics per year;
- query "(homomorphic) (encryption OR encrypt OR cipher OR encode) (cryptosystem OR cryptography OR cybersecurity)". We select "since 2005" and "since 2013" and take all papers for each result, including their citation statistics per year.

The last part of the literature analysis consists of highlighting popular applications and tools in the HE area under the premise that a popular theme has more results in a search engine like Google Scholar than a poorly addressed.

Google Scholar indexes the majority of publications of interest. Hence, for Google Scholar, we conducted searches literature from the following independent queries:

(1). For HE libraries:

- query '(("[abbreviation]" OR "[original name]") AND "homomorphic encryption")'. We count the papers indexed for each result;
- e.g., query '(("cuHE" OR "CUDA Homomorphic Encryption") AND "homomorphic encryption")' for cuHE tool. Abbreviation set = {cuHE, PALISADE, HEAAN, HE-transformer, TFHE, SEAL, and HElib}.

(2). For HE specific areas of applications:

- query '("medical" OR "health") AND "homomorphic encryption")'. We count the papers indexed;
- query '("financial" OR "insurance") AND "homomorphic encryption")'. We count the papers indexed;
- query '("genomics" OR "DNA") AND "homomorphic encryption")'. We count the papers indexed;
- query '("government" OR "smart government" OR "smart cities") AND "homomorphic encryption")'. We count the papers indexed;
- query '("industrial" OR "cyber-physical") AND "homomorphic encryption")'. We count the papers indexed;
- query '(biometrics AND "homomorphic encryption")'. We count the papers indexed.

Figure 4 presents the number of published articles and thesis (Fig. 4a), and citations (Fig. 4b) related to HE.

The analysis is conducted based on the PQDT as well as the WoS database. PQDT is a comprehensive collection of dissertations and thesis, which offers millions of works from worldwide universities.

Both graphs show a significant increase in publications and citations since 2013 due to the practicability demonstration of the first FHE scheme in 2009 [3] and its implementation in 2011 [4].

The following analysis considers only works written within the last five years due to two main reasons: the publications and dissertations of the last five years include 74% of the total elements. This number provides a reasonable definition of recent works.

A keyword analysis of the thesis and dissertations can guide to identify trending topics in the HE area. We collected the set of keywords from HE-related literature published in the last five years.

Figure 5 presents the prevalent terms and the number of times they are included in the research. Two significant elements are identified. First, words closely related to HE concepts such as data, security, privacy, cryptography, privacy-preserving, cybersecurity, and lattices are prevalent (see Fig. 5a). The second popular field is associated with: cloud computing, multi-party computing, and secure computing. The characteristics of these environments make them suitable to receive the benefits of HE.

Similarly, but with a different direction, a keyword analysis highlights terms related to HE specific areas of applications. It allows an overview of the emerging research area of applicability. Figure 5b presents the keywords related to particular applications in the HE domain. It can measure popularity by the number of times a term is included in the research. The

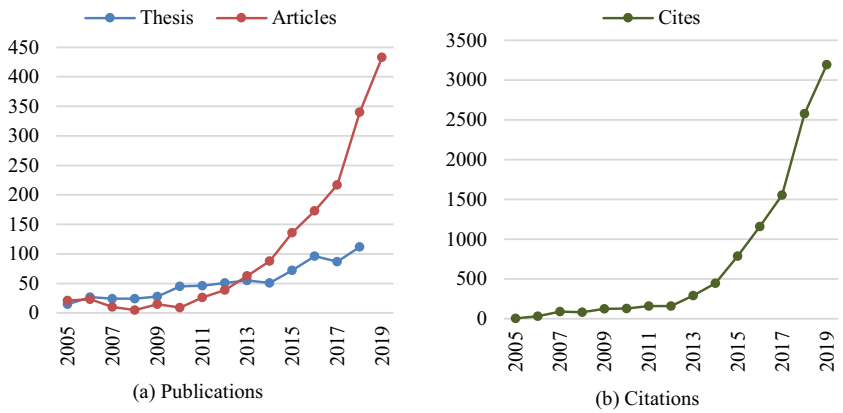

Fig. 4 Number of publications (**a**) and citations (**b**) related to HE

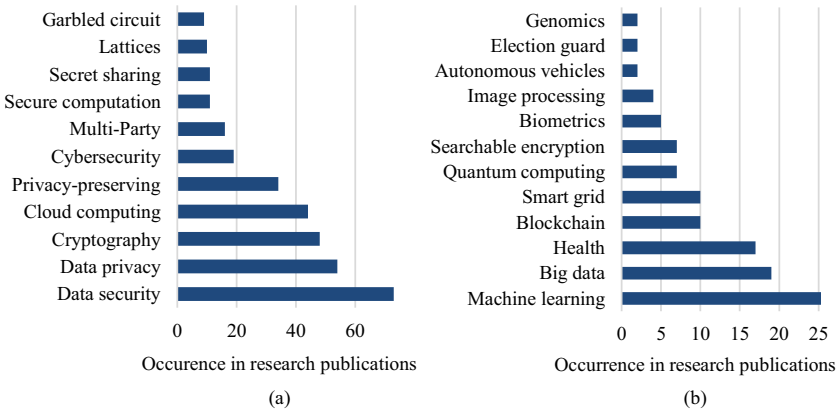

Fig. 5 Keywords related to (**a**) HE concepts and (**b**) specific applications published in the last five years

domains are not mutually exclusive. One study can address a topic in two or more fields.

According to Fig. 5b, machine learning and big data are the two most significant fields in the HE domain, mainly due to datasets' sensibility. It also shows health, robotics, biometrics, and genomics as potential areas to adopt these approaches. Smart grid and blockchain propose decentralized security systems of data exchange and processing.

Quantum computing evidences the concern in the cryptographic community to generate post-quantum cryptosystems. Searchable encryption illustrates the need for a system with search capability over encrypted information, i.e., searching without disclosure. The engine does not know what it is looking for, but it does [114]. Also, applications such as image processing, autonomous vehicles, and election guard are prevalent.

Finally, in Fig. 6, we present relevant applications and tools based on the number of related documents provided by the Google Scholar search engine. Our goal is to highlight popular topics under the premise that a popular theme has more results in a search engine than a poorly addressed. Several literature approaches have applied HE schemes to tangible real-world applications in health, genomics, smart government, cyber-physical industrial systems, and education (see more details in [37–39]).

Our search includes applications highlighted by the keyword analysis conducted above. It considers keywords such as robotics, biometrics, and financial.

Figure 6a presents the popularity of the HE-specific applications. The x-axis value represents the number of publications.

On the other hand, several research groups worldwide have developed public libraries of HE for specific applications and general-purpose use. Some examples are mentioned in the previous sections. A trend analysis of HE tools can give a broader idea of their current use.

Figure 6b presents the popularity of the HE libraries. The query is performed considering both the library's original name, its abbreviation.

The systematic analysis confirms the research community's emerging interest in the construction of HE machine learning models for handling highly sensitive data and processing encrypted data.

The use of well-known general-purpose tools, such as HElib and SEAL, characterized by the effective execution of homomorphic operations, is widespread. It also underlined tools such as PALISADE or application areas such as biometrics, finance, and robotics.

Previous surveys, to the best of our knowledge, did not distinguish topics of the HE development.

Modern data security practices successfully protect stored data and data in transit from non-trustworthy third-parties but do not protect the data while decrypted to be processed. This vulnerability enables a wide range of potential application lines to develop HE schemes in almost any domain. In a nutshell, we are dealing with a nascent research area in constant

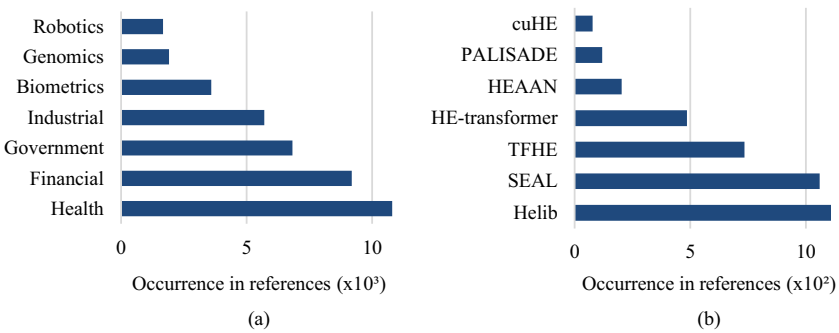

Fig. 6 Occurrence in (**a**) application areas and (**b**) commonly general-purpose HE libraries

growth, with innumerable potentials and commensurable benefits.

7 NN-HE implementations

This section presents a comparison of three NN-HE implementations. We use high-level and low-level tools and compare their performance. We show compromise between technologies and highlight combining their potentialities.

On one side, several tools are well known and highly validated. But they do not provide the facility to incorporate new data representation and their processing. On the other side, basic development offers more flexibility, but its implementation, validation, and testing are more time-consuming.

Most HE tools are developed in low-level languages, such as C or C++, with the idea of increasing application speed. The main tools for developing NNs are implemented in high-level languages, such as Python, for quick understanding. NN frameworks have greatly simplified novel methods of development, but their structures are not designed to support alternative information representations.

The objective of NN-HE systems is threefold: accuracy, data security, and computational complexity. Extremely secure models are not useful if they do not offer acceptable accuracy and computational complexity. So, the three objectives should always be presented in the system design. Here, the trade-off between technologies comes into place.

For studying purposes, we use PyTorch [107] as a representative ML tool. It is a deep learning framework used in many papers of top research conferences in the past couple of years. PyTorch has an important prevalent among the research community and industry as well. A friendly and flexible environment is its primary goal. Users can quickly and easily perform experiments on the platform.

An efficient implementation of NN-HE requires flexibility on both NN and HE technologies, combining HE tool potentialities with the PyTorch facilities.

PySyft [115], CrypTen [116], and TenSEAL libraries offer a bridge between the PyTorch platform and many privacy-preserving techniques described in a long history of academic research.

We compare the performance of three NNs implemented in PyTorch, SEAL, and a combination of both (SEAL + PyTorch). SEAL is a low-level tool, and PyTorch is a high-level tool, both with high adoption in their respective domains. A combination of both approaches provides a middle point between efficiency and facility of implementation.

The evaluation considers three types of instances: native, small, and large. Native (N) instance corresponds to the original implementations using specifications, parameters, and characteristics proposed in the corresponding papers [52, 57]. Small (S) instance has only one convolutional layer. Large instance (L) uses a convolutional network with six-layer architecture: one convolutional input layer with 28×28 input nodes, four hidden layers, and one output fully connected layer with ten neurons (one per digit or class). The difference between a small and large instance lies in the number of convolutional layers. The four hidden layers include one activation layer with pooling, one convolutional, one pooling, and a fully connected activation layer. All neurons use a square activation function.

We use the two datasets that consist of the well-known handwritten digits MNIST (Modified National Institute of Standards and Technology) database and Fashion MNIST (F-MNIST) - MNIST-like dataset of labeled fashion images. They are commonly used for training various image processing systems.

The test set includes 10,000 examples. The analysis is performed on a computer with Windows 10 of 64-bit OS and an Intel(R) Core (TM) i7-8565U CPU 1.8 GHz with 16 GB of memory and 256 GB of SSD.

Table 5 presents the accuracy, latency, and memory usage of three CNNs with MNIST and F-MNIST. Latency corresponds to the time required to process a single prediction

Table 5 Performance comparison of three implementations of Convolutional Neural Networks

Tool	HE	NN-HE	Instance	MNIST			F-MNIST		
				Lat. (s)	Acc. (%)	Memory (Mb)	Lat. (s)	Acc. (%)	Memory (Mb)
PyTorch		–	L	0.011	99.51	465.20	0.010	89.20	448.42
SEAL	•	CryptoNets	N	283.56	98.95	4750.21	290.01	83.74	5781.14
	•	LoLa	S	0.34	96.92	593.70	0.34	82.02	576.32
	•	LoLa	N	3.50	98.95	2076.40	3.45	83.74	2010.90
	•	LoLa	L	14.43	99.20	2781.60	14.21	84.13	2430.10
SEAL + PyTorch	•	–	S	1.2	96.90	1543.34	1.16	82.01	1453.21
	•	–	N	12.67	98.88	7301.92	12.52	83.71	7135.67
	•	–	L	46.92	99.16	9871.81	46.11	84.08	8869.10

request. CryptoNets takes the same processing time for one or 4096 predictions.

Results show the trade-off between the implemented approaches in memory and latency. The computational resources of the framework implemented in Phyton is compromised concerning other approaches in a language such as C++.

We are at the beginning of a long road ahead exploring the NN-HE field. Several approaches are restricted due to higher computational requirements and complex space of functions. However, the analysis elucidates a competitive accuracy performance of PyTorch + SEAL implementation.

NN-HE is a nascent area with a small number of primitives. However, these primitives can achieve an accuracy similar to the non-secure versions for some cases and under certain circumstances. While MNIST and F-MNIST are datasets smaller than medical or financial ones, like many other complex tasks in modern computing, where NN-HE can take advantage of hardware support such as GPU, FPGA, or ASIC.

8 Challenges

Although the HE standards, platforms, and implementations presented in this work help advancing NN-HE, there are still some open challenges to be solved: overhead, performance, interoperability, bootstrapping bottlenecks, sign determination, common frameworks, etc.

Overhead. NN-HE has a significant overhead compared with its unencrypted analogous, making it impractical for many applications. The training phase of NN consists of a computationally intensive task for non-HE models. With HE, it becomes more challenging even with advanced technologies. A new tendency is to avoid the training phase by using pre-trained models to achieve a balance between complexity and accuracy.

Parallelization. One way to deal with the computational overhead is to incorporate well-known and new parallelizing techniques. NN-HE models can be adapted to use high-performance computing, distributed systems, and specialized resources. Multi-core processing units (GPU, FPGA, etc.) or customized chips (ASIC) technologies give the possibility of friendlier and efficient NN-HE environments. Another way to improve the overall efficiency is related to the possibility of batching and parallelizing several bootstrapping operations together.

Polynomial Approximation. A crucial challenge in developing NN-HE consists of the computational design for the homomorphic processing of the neuron's inner functions. NN-HE requires operations not supported by HE, so it is necessary to find cryptographically compatible replacement functions to operate over encrypted data.

The activation function is an essential element in the construction of an effective NN-HE. It determines NN-HE accuracy and computational efficiency. Moreover, activation functions have a significant effect on the converging speed of the network. Also, its derivative, also known as gradient, is fundamental in the training phase.

Multiple approaches address the limitation by polynomially approximating non-compatible functions with a cryptographically consistent polynomial form. These functions should exhibit a trade-off between complexity and accuracy, limiting the efficiency of conventional approximation techniques [117].

In practice, an inadequate approximation function can result in poor performance and long processing time of NN-HE. Moreover, it produces larger encrypted messages that increase memory use.

The challenge of designing a cryptographically computable approximation of the activation function is in identifying low-degree polynomial with a minimal error and good accuracy.

Leveled HE schemes. Another vital direction focuses on designing schemes without bootstrapping that supports the NN evaluation of bounded (pre-determined) depth. Such leveled HE schemes dramatically improves performance by removing the bottleneck and complexity generated by the bootstrapping recrypt function. However, this approach limits the deep learning implementation. While it is efficient for bounded NN-HE, the complexity may become undesirably high for deep learning models.

Binary Neural Networks (BNN) are emerged as an area of opportunity to achieve blind non-interactive NN-HE models. Since the space of functions is restricted, the solution should be limited by the number of possible inputs and outputs. In BNN, every layer maps a binary input to a binary output, using a set of binary weights and a binary activation function. For the bias, it applies a batch normalization before each activation function. The input data is binarized using a predefined threshold.

In general, data and weights in non-standard binary representations $\{-1, 1\}$ can be mapped to binary space $\{0, 1\}$ by replacing -1 with 0. The weighted-sum can be performed by an element-wise product. It uses the logical operator XNOR and subsequently sums the result of the previous step by counting the number of ones. The binary activation function $f(y)$ returns 1 if $y > 0$ and -1 otherwise.

Interoperability of existing ML tools is another challenging problem to achieve friendly NN-HE models. Popular NN frameworks have simplified the development of novel NN methods, but they do not provide HE support. The development of NN-HE depends on the current tools and their flexibility to supply or incorporate new approaches. The low flexibility of several HE libraries restricts their interaction with other frameworks. It makes more complicated the design, testing, and implementing new models, hence, increases the development time.

Automatization. The development of HE applications implies manual configuration and high expertise in different domains: scheme-specific optimizations, complicated security parameter setting, low-level programming, among others. Improper setup can generate low performance, encryption insecurity, and corrupted or unrecoverable information. The automatization and simplification of the development lifecycle are required. The implementation should be easily employed by beginners and highly configurable for expert users.

Common framework. Most of the related works focus on specific environments with different characteristics. It limits the possibility of comparing new approaches to state-of-art algorithms. A standard framework can simplify the comparative analysis and show the advantages of new models. It should simplify the adoption of libraries, algorithms, measures, and statistical analysis.

9 Conclusion

Easy deploying of machine learning in clouds makes homomorphic encryption an important mechanism to solve security and privacy concerns. Many solutions have been proposed in the continuum between theoretical and applied aspects. Theoretical research increases understanding of the problem by developing new theories and algorithms, while applied research is to solve real-world problems in business, medicine, industry, etc.

Although different aspects have been discussed in the literature, a systematic comparison of the state-of-the-art NN-HE solutions has not been conducted.

In this paper, we:

- review the latest advances of HE cryptosystems, focusing mainly on the intersection of cryptography and neural networks, discussing the state-of-the-art and the state-of-the-practice;
- describe fundamental concepts, such as bootstrapping, key-switching, noise, running errors, algorithmic errors, etc., that are easy to follow for readers who are not familiar with NN-HE. We cover the main theoretical results, capabilities, opportunities, potential applications, and trends of design and application;
- Highlight the compromises between NN technologies and HE feasibility, combining their potentialities and important limitations
- discuss current development tools, frameworks, emerging trends in the research, and relevant application domains;
- compare three NN-HE implementations with CryptoNets and LoLa, both using high-level and low-level tools to demonstrate the advantages and disadvantages of each approach.
- Finally, we sketch open problems, challenges, and solutions associated with homomorphic cryptosystems and machine learning.

The key goal is to show how to process encrypted information by non-trustworthy third-parties without disclosing confidential data. Specifically, we concentrate on the privacy-preserving neural networks and their implementations.

Acknowledgments This work was supported by the Ministry of Education and Science of the Russian Federation (Project 075-15-2020-788).

Appendix

To facilitate understanding the described ideas, we summarize the main HE and NN terminology. Table 6 shows the general acronyms. Table 7 presents main notations. Table 8 describes the terms used in the paper.

Acronyms

Notations

Table 6 Acronyms

Acronym	Description
HE	Homomorphic Encryption
PHE	Partially Homomorphic Encryption
SHE	Somewhat Homomorphic Encryption
FHE	Fully Homomorphic Encryption
CSP	Cloud Service Provider
NN	Neural Network
DNN	Deep Neural Network
CNN	Convolutional Neural Network
BNN	Binary Neural Network
NN-HE	Neural Network with Homomorphic Encryption
ICP	Ideal Coset Problem
NTRU	Nth-Degree Truncated Polynomial Ring Unit
A-GCD	Approximate of Greatest Common Divisor
LWE	Learning with Error
RLWE	Ring Learning with Error
AI	Artificial Intelligence
ML	Machine Learning
MLaaS	Machine Learning as a Service
MLaaS-HE	MLaaS with Homomorphic Encryption
SMC	Secure Multi-party Computing

Table 6 (continued)

Acronym	Description
LoLa	Low-Latency CryptoNets
LR	Logistic Regression
DT	Decision Tree
ReLU	Rectified Linear Unit
GPU	Graphics Processing Unit
SGD	Stochastic Gradient Descent
RNS	Residue Number System
SL	Supervised Learning
UL	Unsupervised Learning
CRT	Chinese Remainder Theorem
SIMD	Single Instruction Multiple Data
FPGA	Field-Programmable Gate Array
ASIC	Application-Specific Integrated Circuit
NTT	Number-Theoretic Transform
PQDT	ProQuest Dissertations and Theses
WoS	Web of Science
MNIST	Modified National Institute of Standards and Technology
F-MNIST	Fashion MNIST

Terminology

Table 7 Notations

Notation	Description
$\mathbb{P}$	Plaintext space
$\mathbb{C}$	Ciphertext space
m_i	Message i in the plaintext space
q_i	Prime i
c_i	Message m_i in the ciphertext space (encryption of m_i)
$\ddot{\times}$	Homomorphic multiplication
$\ddot{+}$	Homomorphic addition
c_+, $c_\times$	Addition and multiplication of ciphertexts, respectively
$a \leftarrow b$	Set a as b
N	Polynomial of degree N
$(a)\ mod\ n$	Modulo n of a
b_i^2	i-th quadratic nonresidue value
$gcd(a,b)$	Greatest common division of a and b
$I_1, I_2, \ldots, I_l$	Sequence of l instructions
P	Program P
$\widehat{P}$	Obfuscated version of program P
O	Obfuscating transformation
ε	Homomorphic encryption scheme
e	Noise
pk	Public key
λ	Security parameter

Table 7 (continued)

Notation	Description
bk	Bootstrapping key
sk_i	Secret key i
$\langle \overline{sk}_i \rangle$	Secret key i on bits
δ_ε	Circuit on the homomorphic encryption scheme
D_ε	$Decrypt_\varepsilon$ function expressed as a circuit
α	Machine learning model
ϑ	Neural network output
$\widehat{\vartheta}$	Neural network expected output
$L\left(\widehat{\vartheta}, \vartheta\right)$	Loss function
n_I	Number of inputs in neurons
$x = (x_1, \ldots, x_{n_I})$	Neuron inputs
$w = (w_1, \ldots, w_{n_I})$	Neuron weights
β	Neuron bias
$y = \sum_{i=1}^{n_I} w_i\, x_i + \beta$	Neuron weighted sum function
l	Number of neural network layers
f	Activation function
d	Activation function degree
Δ	Scaling factor
sgn(x)	Sign function
R_t^N	Ring
v_i	Vector of i dimensions
Φ_M	Cyclotomic polynomial of degree M

Table 8 Terms

Term	Description
Bootstrapping	Process to reduce the noise in ciphertexts by the homomorphic evaluation of the decryption circuit, where the fresh ciphertext contains less noise than the original. This notion maintains the noise under a threshold, allowing unbounded homomorphic computations.
Bootstrapping key	Encryption of the secret key. When the bootstrapping process encrypts the ciphertext again, the bootstrapping key removes the inner encryption by homomorphically evaluating the doubly encrypted plaintext.
Ciphertext	Encrypted message.
Algorithmic error	Error in data processing that compromises the correctness of the results generated by representing real numbers with integer polynomial. An error of this nature can occur in numbers in the vicinity of zero, even without adding noise.
Functional error	Rounding error that results from the polynomial approximation of activation functions of neural networks.

Table 8 (continued)

Term	Description
Homomorphism	Structure-preserving transformation. It describes a correspondence between functions on the space of texts and ciphertexts.
Key-switching	Multi-key handling technique to perform bootstrapping operations, each recrypt process requires a new secret key.
Circular security	Self-encryption approach where the public key is encrypted under itself. In contrast to the key-switching technique, it avoids the use of several keys.
Noise	Moderate quantity of error injected in the encrypted message.
Plaintext	Original unencrypted message.
Semantically secure	If an attacker cannot distinguish two encryptions from each other even if the attacker knows (or has chosen) the corresponding plaintexts.
Padding	Practice of adding data to the beginning, middle, or end of a message prior to encryption.

References

1. Cloud Security Aliance (2019) Top threats to cloud computing: Egregious eleven. Accessed 20.07.20
2. Vaikuntanathan V (2011) Computing blindfolded: new developments in fully Homomorphic Encryption. In: 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science. Palm Springs, CA, pp 5–16. https://doi.org/10.1109/FOCS.2011.98
3. Gentry C (2009) A fully Homomorphic encryption scheme. In: Stanford University. Stanford, PhD Thesis
4. Gentry C, Halevi S (2011) Implementing gentry's fully homomorphic encryption scheme. In: Paterson KG (ed) Advances in Cryptology – EUROCRYPT 2011. Lecture notes in computer science, vol 6632. Springer, Berlin, Heidelberg, pp 129–148. https://doi.org/10.1007/978-3-642-20465-4_9
5. Player R (2017) Parameter selection in lattice-based cryptography. In: University of London. PhD Thesis, Royal Holloway
6. Rivest R, Shamir A, Adleman L (1978) A method for obtaining digital signatures and public-key cryptosystems. Commun ACM 21:120–126. https://doi.org/10.1145/359340.359342
7. ElGamal T (1985) A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Trans Inf Theory 31: 469–472. https://doi.org/10.1109/TIT.1985.1057074
8. Goldwasser S, Micali S (1982) Probabilistic encryption and how to play mental poker keeping secret all partial information. In: Proceedings of the fourteenth annual ACM symposium on Theory of computing (STOC '82). ACM, New York, USA, pp 365–377. https://doi.org/10.1145/800070.802212
9. Benaloh J (1994) Dense probabilistic encryption. Proceedings of the workshop on selected areas of cryptography, In, pp 120–128
10. Naccache D, Stern J (1998) A new public key cryptosystem based on higher residues. In: Proceedings of the 5th ACM conference on Computer and communications security (CCS '98). ACM, New York, USA, pp 59–66. https://doi.org/10.1145/288090.288106
11. Okamoto T, Uchiyama S (1998) A new public-key cryptosystem as secure as factoring. In: Nyberg K (ed) Advances in Cryptology — EUROCRYPT'98. Lecture notes in computer science, vol 1403. Springer, Berlin, Heidelberg, pp 308–318. https://doi.org/10.1007/BFb0054135
12. Paillier P (1999) Public-key cryptosystems based on composite degree residuosity classes. In: Stern J (ed) Advances in Cryptology - EUROCRYPT '99. Lecture notes in computer science, vol 1592. Springer, Berlin, Heidelberg, pp 223–238. https://doi.org/10.1007/3-540-48910-X_16
13. Damgård I, Jurik M (2001) A Generalisation, a simplification and some applications of paillier's probabilistic public-key system. In: Kim K (ed) Public Key Cryptography. PKC 2001. Lecture Notes in Computer Science, vol 1992. Springer, Berlin, Heidelberg, pp 119–136. https://doi.org/10.1007/3-540-44586-2_9
14. Galbraith SD (2002) Elliptic curve paillier schemes. J. Cryptology 15:129–138. https://doi.org/10.1007/s00145-001-0015-6
15. Kawachi A, Tanaka K, Xagawa K (2007) Multi-bit cryptosystems based on lattice problems. In: Okamoto T, Wang X (eds) Public Key Cryptography – PKC 2007. Lecture Notes in Computer Science, vol 4450. Springer, Berlin, Heidelberg, pp 315–329. https://doi.org/10.1007/978-3-540-71677-8_21
16. Minelli M (2018) Fully homomorphic encryption for machine learning. In: PSL Research University. PhD Thesis, Paris
17. Boneh D, Goh EJ, Nissim K (2005) Evaluating 2-DNF formulas on ciphertexts. In: Kilian J (ed) Theory of cryptography. TCC 2005, Lecture Notes in Computer Science, vol 3378. Springer, Berlin, Heidelberg, pp 325–341. https://doi.org/10.1007/978-3-540-30576-7_18
18. Gjøsteen K (2004) Subgroup membership problems and public key cryptosystem. In: Norwegian University of Science and Technology. PhD Thesis, Trondheim
19. Yao A (1982) Protocols for secure computations. In: 23rd Annual Symposium on Foundations of Computer Science. USA, Chicago, pp 160–164. https://doi.org/10.1109/SFCS.1982.38
20. Sander T, Young A, Yung M (1999) Non-interactive cryptocomputing for NC1. In: Proceedings of the 40th Annual Symposium on Foundations of Computer Science (FOCS '99). IEEE, USA, pp 554–566. https://doi.org/10.1109/SFFCS.1999.814630
21. Ishai Y, Paskin A Evaluating branching programs on encrypted data. Theory Cryptogr:575–594
22. Didie W, Hellma M (1976) New directions in cryptography. IEEE Trans Inf Theory 22:472–492
23. Van Dijk M, Gentry C, Halevi S, Vaikuntanathan V (2010) Fully Homomorphic Encryption over the Integers. In: Gilbert H (ed) Advances in Cryptology – EUROCRYPT 2010, Lecture Notes in Computer Science, vol 6110. Springer, Berlin, Heidelberg, pp 24–43. https://doi.org/10.1007/978-3-642-13190-5_2
24. Chillotti I, Gama N, Georgieva M, Izabachène M (2016) Faster Fully Homomorphic Encryption: Bootstrapping in Less Than 0.1 Seconds. In: Cheon J, Takagi T (eds) Advances in Cryptology – ASIACRYPT 2016. Lecture Notes in Computer Science, vol 10031. Springer, Berlin, Heidelberg, pp 3–33. https://doi.org/10.1007/978-3-662-53887-6_1
25. Brakerski Z, Vaikuntanathan V (2011) Efficient fully homomorphic encryption from (Standard) LWE. In: 2011 IEEE 52nd

Annual Symposium on Foundations of Computer Science. Palm Springs, CA, pp 97–106. https://doi.org/10.1109/FOCS.2011.12
26. Brakerski Z, Gentry C, Vaikuntanathan V (2012) (Leveled) fully homomorphic encryption without bootstrapping. In: Proceedings of the 3rd Innovations in Theoretical Computer Science Conference - ITCS '12. ACM, New York, USA, pp 309–325. https://doi.org/10.1145/2090236.2090262
27. Gentry C, Sahai A, Waters B (2013) Homomorphic encryption from learning with errors: conceptually-simpler, asymptotically-faster, attribute-based. In: Canetti R, Garay JA (eds) Advances in Cryptology – CRYPTO 2013. Lecture Notes in Computer Science, vol 8042. Springer, Berlin, Heidelberg, pp 75–92. https://doi.org/10.1007/978-3-642-40041-4_5
28. Gentry C (2010) Computing arbitrary functions of encrypted data. Commun ACM 53:97–105. https://doi.org/10.1145/1666420.1666444
29. Fan J, Vercauteren F (2012) Somewhat practical fully Homomorphic encryption. IACR Cryptol. ePrint Arch:2012/144
30. Brakerski Z (2012) Fully homomorphic encryption without modulus switching from classical GapSVP. In: Safavi-Naini R, Canetti R (eds) Advances in Cryptology – CRYPTO 2012. Lecture Notes in Computer Science, vol 7417. Springer, Berlin, Heidelberg, pp 868–886. https://doi.org/10.1007/978-3-642-32009-5_50
31. Rohloff K, Cousins DB (2014) A scalable implementation of fully Homomorphic encryption built on NTRU. In: Böhme R, Brenner M, Moore T, Smith M (eds) Financial Cryptography and Data Security. FC 2014. Lecture Notes in Computer Science, vol 8438. Springer, Berlin, Heidelberg, pp 221–234. https://doi.org/10.1007/978-3-662-44774-1_18
32. Hiromasa R, Abe M, Okamoto T (2015) Packing messages and optimizing bootstrapping in GSW-FHE. In: Katz J (ed) Public-Key Cryptography - PKC 2015. Lecture Notes in Computer Science, vol 9020. Springer, Berlin, Heidelberg, pp 699–715. https://doi.org/10.1007/978-3-662-46447-2_31
33. Alperin-Sheriff J, Peikert C (2014) Faster bootstrapping with polynomial error. In: Garay JA, Gennaro R (eds) Advances in Cryptology – CRYPTO 2014. Lecture Notes in Computer Science, vol 8616. Springer, Berlin, Heidelberg, pp 297–314. https://doi.org/10.1007/978-3-662-44371-2_17
34. Cheon JH, Kim A, Kim M, Song Y (2017) Homomorphic Encryption for Arithmetic of Approximate Numbers. In: Takagi T, Peyrin T (eds) Advances in Cryptology – ASIACRYPT 2017. Lecture Notes in Computer Science, vol 10624. Springer, Cham, pp 409–437. https://doi.org/10.1007/978-3-319-70694-8_15
35. Rivest RL, Dertouzos ML, Adleman L (1978) On data banks and privacy homomorphisms. Found Secur Comput 4:160–179
36. Collberg C, Thomborson C, Low D (1997) A taxonomy of obfuscating transformations. Technical Report 148. University of Auckland, New Zealand
37. Armknecht F, Boyd C, Carr C, Gjøsteen K, Jäschke A, Reuter CA, Strand M (2015) A guide to fully homomorphic encryption. IACR Cryptology ePrint Archive 1192
38. Naehrig M, Lauter K, Vaikuntanathan V (2011) Can homomorphic encryption be practical? In: Proceedings of the 3rd ACM workshop on Cloud computing security workshop - CCSW '11. ACM, New York, USA, pp 113–124. https://doi.org/10.1145/2046660.2046682
39. Archer D, Chen L, Cheon JH, Gilad-Bachrach R, Hallman RA, Huang Z, Jiang X, Kumaresan R, Malin BA, Sofia H, Song Y, Wang S (2017) Applications of Homomorphic encryption. Technical report, HomomorphicEncryption.org, Redmond WA
40. Acar A, Aksu H, Selcuk Uluagac A, Aksu H, Uluagac AS (2018) A survey on Homomorphic encryption schemes: theory and implementation. ACM Comput Surv 51:1–35. https://doi.org/10.1145/3214303
41. Martins P, Sousa L, Mariano A (2017) A survey on fully Homomorphic encryption: an engineering perspective. ACM Comput Surv 50:33–33. https://doi.org/10.1145/3124441
42. Parmar PV, Padhar SB, Patel SN, Bhatt NI, Jhaveri RH, S'ad Vidya S, Shri S'ad M, Mandal V (2014) Survey of various Homomorphic encryption algorithms and schemes. Int J Comput Appl 91:26–32. https://doi.org/10.5120/15902-5081
43. Sobitha Ahila S, Shunmuganathan KL (2014) State of art in Homomorphic encryption schemes. Int J Eng Res Appl 4:37–43
44. Gentry C (2014) Computing on the edge of Chaos: structure and randomness in encrypted computation. In: Proceedings of the 2014 International Congress of Mathematicians (ICM), pp 609–632. http://eprint.iacr.org/2014/610
45. Aguilar-Melchor C, Fau S, Fontaine C, Gogniat G, Sirdey R (2013) Recent advances in Homomorphic encryption: a possible future for signal processing in the encrypted domain. IEEE Signal Process Mag 30:108–117. https://doi.org/10.1109/MSP.2012.2230219
46. Hrestak D, Picek S (2014) Homomorphic encryption in the cloud. In: 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO'14). IEEE, Opatija, pp 1400–1404. https://doi.org/10.1109/MIPRO.2014.6859786
47. Moore C, O'Neill M, Hanley N, O'Sullivan E (2014) Accelerating integer-based fully homomorphic encryption using Comba multiplication. In: IEEE Workshop on Signal Processing Systems (SiPS). IEEE, Belfast, pp 1–6. https://doi.org/10.1109/SiPS.2014.6986063
48. Hunt T, Song C, Shokri R, Shmatikov V, Witchel E (2018) Chiron: privacy-preserving machine learning as a service. arXiv:1803.05961
49. Zheng Q, Wang X, Khurram Khan M, Zhang W, Gupta BB, Guo W (2018) A lightweight authenticated encryption scheme based on chaotic SCML for railway cloud service. IEEE Access 6:711–722. https://doi.org/10.1109/ACCESS.2017.2775038
50. Bost R, Popa RA, Tu S, Goldwasser S (2015) Machine learning classification over encrypted data. IACR Cryptology ePrint Archive 2014:331
51. Khedr A, Gulak G, Member S, Vaikuntanathan V (2015) SHIELD: Scalable Homomorphic implementation of encrypted data-classifiers. IEEE Trans Comput 65:2848–2858. https://doi.org/10.1109/TC.2015.2500576
52. Dowlin N, Gilad-Bachrach R, Laine K, Lauter K, Naehrig M, Wernsing J (2016) CryptoNets: Applying neural networks to encrypted data with high throughput and accuracy. In: Balcan M-F, Weinberger KQ (eds) Proceedings of the 33rd International Conference on Machine Learning (ICML'16), pp 201–210. JMLR.org
53. Chabanne H, De Wargny A, Milgram J, Morel C, Prouff E (2017) Privacy-preserving classification on deep neural network. IACR Cryptology ePrint Archive 2017/35
54. Hesamifard E, Takabi H, Ghasemi M (2017) CryptoDL: deep neural networks over encrypted data. arXiv:1711.05189
55. Badawi A Al, Chao J, Lin J, Mun CF, Sim JJ, Tan BHM, Nan X, Aung KMM, Chandrasekhar VR (2018) Towards the AlexNet moment for homomorphic encryption: HCNN, the First Homomorphic CNN on Encrypted Data with GPUs. IEEE Transactions on Emerging Topics in Computing. https://doi.org/10.1109/TETC.2020.3014636
56. Zhang Q, Yang LT, Chen Z (2016) Privacy preserving deep computation model on cloud for big data feature learning. IEEE Trans Comput 65:1351–1362. https://doi.org/10.1109/TC.2015.2470255
57. Brutzkus A, Elisha O, Gilad-Bachrach R (2019) Low latency privacy preserving inference. In: Proceedings of the 36th

International Conference on Machine Learning (ICML'19), pp 1295–1304 JMLR.org

58. Takabi H, Hesamifard E, Ghasemi M (2016) Privacy preserving multi-party machine learning with Homomorphic encryption. In: 29th Annual Conference on Neural Information Processing Systems (NIPS). Barcelona, Spain
59. Phong LT, Aono Y, Hayashi T, Wang L, Moriai S (2018) Privacy-preserving deep learning via additively Homomorphic encryption. IEEE Trans Inf Forensics Secur 13:1333–1345. https://doi.org/10.1109/TIFS.2017.2787987
60. Wagh S, Gupta D, Chandran N (2019) SecureNN: 3-party secure computation for neural network training. Proc Priv Enhancing Technol 2019:26–49. https://doi.org/10.2478/popets-2019-0035
61. Babenko M, Tchernykh A, Chervyakov N, Kuchukov V, Miranda-López V, Rivera-Rodriguez R, Du Z, Talbi E-G (2019) Positional characteristics for efficient number comparison over the Homomorphic encryption. Program Comput Softw 45:532–543. https://doi.org/10.1134/S0361768819080115
62. Aslett LJM, Esperança PM, Holmes CC (2015) A review of homomorphic encryption and software tools for encrypted statistical machine learning. arXiv:1508.06574
63. Bos JW, Lauter K, Naehrig M (2014) Private predictive analysis on encrypted medical data. J Biomed Inform 50:234–243. https://doi.org/10.1016/j.jbi.2014.04.003
64. Xu C, Chen J, Wu W, Feng Y (2016) Homomorphically encrypted arithmetic operations over the integer ring. In: Bao F, Chen L, Deng R, Wang G (eds) Information Security Practice and Experience. ISPEC 2016. Lecture Notes in Computer Science, vol 10060. Springer, Cham, pp 167–181. https://doi.org/10.1007/978-3-319-49151-6_12
65. Aono Y, Hayashi T, Phong LT, Wang L (2016) Scalable and secure logistic regression via homomorphic encryption. In: Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy - CODASPY 2016. ACM, New York, USA, pp 142–144. https://doi.org/10.1145/2857705.2857731
66. Kim A, Song Y, Kim M, Lee K, Cheon JH (2018) Logistic regression model training based on the approximate homomorphic encryption. BMC Med Genet 11:83. https://doi.org/10.1186/s12920-018-0401-7
67. Coron JS, Lepoint T, Tibouchi M (2014) Scale-invariant fully homomorphic encryption over the integers. In: Krawczyk H. (eds) Public-Key Cryptography. PKC 2014. Lecture Notes in Computer Science, vol 8383. Springer, Berlin, Heidelberg, pp 311–328. https://doi.org/10.1007/978-3-642-54631-0_18
68. Wood A, Shpilrain V, Najarian K, Kahrobaei D (2019) Private naive Bayes classification of personal biomedical data: application in Cancer data analysis. Comput Biol Med 105:144–150. https://doi.org/10.1016/j.compbiomed.2018.11.018
69. Kaushik S, Gandhi C (2020) Capability based outsourced data access control with assured file deletion and efficient revocation with trust factor in cloud computing. Int J Cloud Appl Comput 10:64–84. https://doi.org/10.4018/IJCAC.2020010105
70. Premkamal PK, Pasupuleti SK (2020) Alphonse PJA: efficient escrow-free CP-ABE with constant size Ciphertext and secret key for big data storage in cloud. Int J Cloud Appl Comput 10:28–45. https://doi.org/10.4018/IJCAC.2020010103
71. Tchernykh A, Babenko M, Kuchukov V, Miranda-Lopez V, Avetisyan A, Rivera-Rodriguez R, Radchenko G (2019) Data reliability and redundancy optimization of a secure multi-cloud storage under uncertainty of errors and falsifications. In: 2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW). Rio de Janeiro, Brazil, pp 565–572. https://doi.org/10.1109/IPDPSW.2019.00099
72. Zhang GP (2000) Neural networks for classification: a survey. IEEE Trans Syst Man Cybern Part C Appl Rev 30:451–462. https://doi.org/10.1109/5326.897072
73. Rondeau T (2020) Data protection in virtual environments (DPRIVE). DARPA/MTO, Technical report
74. Tchernykh A, Schwiegelsohn U, Talbi EG, Babenko M (2019) Towards understanding uncertainty in cloud computing with risks of confidentiality, integrity, and availability. J Comput Sci 36:100581. https://doi.org/10.1016/j.jocs.2016.11.011
75. Miranda-Lopez V, Tchernykh A, Babenko M, Avetisyan A, Toporkov V, Drozdov AY (2020) 2Lbp-RRNS: Two-levels RRNS with backpropagation for increased reliability and privacy-preserving of secure multi-clouds data storage. IEEE Access. Multidiscip. Open Access J. 1–1. https://doi.org/10.1109/ACCESS.2020.3032655
76. Kidd JM, Cooper GM, Donahue WF, Hayden HS, Sampas N, Graves T, Hansen N, Teague B, Alkan C, Antonacci F, Haugen E, Zerr T (2008) Mapping and sequencing of structural variation from eight human genomes. Nature. 453:56–64. https://doi.org/10.1038/nature06862
77. Bos JW, Lauter K, Loftus J, Naehrig M (2013) Improved security for a ring-based fully homomorphic encryption scheme. In: Stam M (ed) Cryptography and Coding. IMACC 2013. Lecture Notes in Computer Science, vol 8308. Springer, Berlin, Heidelberg, pp 45–64. https://doi.org/10.1007/978-3-642-45239-0_4
78. Chou E, Beal J, Levy D, Yeung S, Haque A, Fei-Fei L (2018) Faster CryptoNets: leveraging sparsity for real-world encrypted inference. arXiv:1811.09953
79. Shokri R, Shmatikov V (2015) Privacy-preserving deep learning. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security. ACM, New York, USA, pp 1310–1321. https://doi.org/10.1145/2810103.2813687
80. Bakshi M, Last M (2020) CryptoRNN-privacy-preserving recurrent neural networks using Homomorphic encryption. In: Dolev S, Kolesnikov V, Lodha S, Weiss G (eds) Cyber Security Cryptography and Machine Learning. CSCML 2020. Lecture Notes in Computer Science, vol 12161. Springer, Cham, pp 245–253. https://doi.org/10.1007/978-3-030-49785-9_16
81. Bourse F, Minelli M, Minihold M, Paillier P (2018) Fast Homomorphic evaluation of deep discretized neural networks. In: Shacham H, Boldyreva A (eds) Advances in Cryptology – CRYPTO 2018. Lecture Notes in Computer Science, vol 10993. Springer, Cham, pp 483–512. https://doi.org/10.1007/978-3-319-96878-0_17
82. Stone MH (1948) The generalized Weierstrass approximation theorem. Math Mag 21(4):167–184
83. Boemer F, Cammarota R, Costache A, Wierzynski C (2019) nGraph-HE2: A high-throughput framework for neural network inference on encrypted data. In: Proceedings of the 7th ACM Workshop on Encrypted Computing & Applied Homomorphic Cryptography. ACM, New York, USA, pp 45–56. https://doi.org/10.1145/3338469.3358944
84. Qaisar Ahmad Al Badawi A, Polyakov Y, Aung KMM, Veeravalli B, Rohloff K (2019) Implementation and performance evaluation of RNS variants of the BFV Homomorphic encryption scheme. IEEE Trans Emerg Top Comput. https://doi.org/10.1109/TETC.2019.2902799
85. Halevi S, Polyakov Y, Shoup V (2019) an improved rns variant of the bfv homomorphic encryption scheme. In: Matsui M (ed) Topics in Cryptology – CT-RSA 2019. Lecture Notes in Computer Science, vol 11405. Springer, Cham, pp 83–105. https://doi.org/10.1007/978-3-030-12612-4_5
86. Bajard JC, Eynard J, Hasan MA, Zucca V (2017) A full RNS variant of FV like somewhat homomorphic encryption schemes. In: Avanzi R, Heys H (eds) Selected Areas in Cryptography – SAC 2016. Lecture Notes in Computer Science, vol 10532.

Springer, Cham, pp 423–442. https://doi.org/10.1007/978-3-319-69453-5_23
87. Cheon JH, Han K, Kim D (2019) Faster Bootstrapping of FHE over the Integers. In: Seo J (ed) Information Security and Cryptology – ICISC 2019. Lecture Notes in Computer Science, vol 11975. Springer, Cham, pp 242–259. https://doi.org/10.1007/978-3-030-40921-0_15
88. Cheon JH, Kim D, Kim D (2020) Efficient Homomorphic comparison methods with optimal complexity. In: Moriai S, Wang H (eds) Advances in Cryptology - ASIACRYPT 2020. Lecture Notes in Computer Science, vol 12492. Springer, Cham, pp 221–256. https://doi.org/10.1007/978-3-030-64834-3_8
89. Gregory RT, Krishnamurthy EV (1984) Methods and applications of error-free computation. Springer-Verlag New York. https://doi.org/10.1007/978-1-4612-5242-9
90. Chen H, Laine K, Player R (2017) Simple encrypted arithmetic library - SEAL v2.1. In: Brenner M et al (eds) Financial Cryptography and Data Security. FC 2017, Lecture Notes in Computer Science, vol 10323. Springer, Cham, pp 3–18. https://doi.org/10.1007/978-3-319-70278-0_1
91. Halevi S, Shoup V (2013) Design and implementation of a Homomorphic-encryption library. IBM Res 6:12–15
92. PALISADE, https://palisade-crypto.org/community
93. Dai W, Sunar B (2016) cuHE: A Homomorphic Encryption Accelerator Library. In: Pasalic E, Knudsen L (eds) Cryptography and Information Security in the Balkans. Lecture Notes in Computer Science, vol 9540. Springer, Cham, pp 169–186. https://doi.org/10.1007/978-3-319-29172-7_11
94. Boemer F, Lao Y, Cammarota R, Wierzynski C (2019) NGraph-HE: A graph compiler for deep learning on Homomorphically encrypted data. In: Proceedings of the 16th ACM International Conference on Computing Frontiers (CF '19). ACM, New York, USA, pp 3–13. https://doi.org/10.1145/3310273.3323047
95. Jia Y, Shelhamer E, Donahue J, Karayev S, Long J, Girshick R, Guadarrama S, Darrell T (2014) Caffe: Convolutional architecture for fast feature embedding. In: Proceedings of the 22nd ACM international conference on Multimedia. ACM, New York, USA, pp 675–678. https://doi.org/10.1145/2647868.2654889
96. Ma Y, Wu L, Gu X, He J, Yang Z (2017) A secure face-verification scheme based on Homomorphic encryption and deep neural networks. IEEE Access 5:16532–16538
97. Tokui S, Oono K, Hido S, Clayton J (2015) Chainer: a next-generation open source framework for deep learning. In: Proceedings of the Workshop on Machine Learning Systems (LearningSys) at the 28th Annual Conference on Neural Information Processing Systems (NIPS), pp 1-6. http://learningsys.org/papers/LearningSys_2015_paper_33.pdf
98. King DE (2009) Dlib-ml: a machine learning toolkit. J Mach Learn Res 10:1755–1758
99. Boura C, Gama N, Georgieva M, Jetchev D (2019) Simulating Homomorphic Evaluation of Deep Learning Predictions. In: Dolev S, Hendler D, Lodha S, Yung M (eds) Cyber Security Cryptography and Machine Learning. CSCML 2019. Lecture Notes in Computer Science, vol 11527. Springer, Cham, pp 212–230. https://doi.org/10.1007/978-3-030-20951-3_20
100. Innes M (2018) Flux: elegant machine learning with Julia. Journal of Open Source Software, 3(25):602. 10.21105/joss.00602
101. Candel A, Parmar V, LeDell E, Arora A (2016) Deep Learning with H2O. 4th ed, Mountain View, CA, H2O.ai Inc
102. Chollet F (2015) Keras. https://keras.io
103. Salem M, Taheri S, Yuan JS (2019) Utilizing transfer learning and Homomorphic encryption in a privacy preserving and secure biometric recognition system. Computers. 8(1):3
104. Kim P (2017) MATLAB deep learning: with machine learning, neural networks and artificial intelligence. Berkeley, CA, Apress. https://doi.org/10.1007/978-1-4842-2845-6
105. Seide F, Agarwal A (2016) CNTK: Microsoft's Open-Source Deep-Learning Toolkit. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, New York, USA, pp 2135–2135. https://doi.org/10.1145/2939672.2945397
106. Chen T, Li M, Li Y, Lin M, Wang N, Wang M, Xiao T, Xu B, Zhang C, Zhang Z (2015) Mxnet: a flexible and efficient machine learning library for heterogeneous distributed systems. arXiv: 1512.01274
107. Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, Killeen T (2019) PyTorch: An Imperative Style, High-Performance Deep Learning Library. In: Proceedings of the Advances in Neural Information Processing Systems, Vancouver, BC, Canada, pp. 8024–8035. arXiv, vol 1912, p 01703
108. Wang X, Maturana D, Yang S, Wang W, Chen Q, Scherer S (2019) Improving learning-based ego-motion estimation with homomorphism-based losses and drift correction. In: 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Macau, China, pp 970–976. https://doi.org/10.1109/IROS40897.2019.8968515
109. Abadi M, Agarwal A, Barham P, Brevdo E (1603) Large-Scale Machine Learning on Heterogeneous Systems. arXiv, Xiaoqiang Z TensorFlow, p 04467
110. Zhu Q, Lv X (2018) 2P-DNN: Privacy-preserving deep neural networks based on Homomorphic cryptosystem. arXiv: 1807.08459
111. Bergstra J, Bastien F, Breuleux O, Lamblin P, Pascanu R, Delalleau O, Bengio Y (2011) Theano: deep learning on GPU with Python. J Mach Learn Res 1:1–48
112. Servia-Rodriguez S, Wang L, Zhao JR, Mortier R, Haddadi H (2017) Personal model training under privacy constraints. Training 40(33):24–38
113. Collobert R, Bengio S, Mariethoz J (2002) Torch: a modular machine learning software library. Technical report, IDIAP. https://infoscience.epfl.ch/record/82802/files/rr02-46.pdf
114. Li Z, Zhao M, Jiang H, Xu Q (2019) Keyword guessing on multi-user searchable encryption. Int J High Perform Comput Netw 14: 60–68. https://doi.org/10.1504/IJHPCN.2019.099744
115. Ryffel T, Trask A, Dahl M, Wagner B (1811) Mancuso J. Passerat-Palmbach J A generic framework for privacy preserving deep learning. arXiv, Rueckert D, p 04017
116. Gunning D, Hannun A, Ibrahim M, Knott B, van der Maaten L, Reis V, Sengupta S, Venkataraman S, Zhou X (2019) CrypTen: a new research tool for secure machine learning with PyTorch. https://ai.facebook.com/blog/crypten-a-new-research-tool-for-securemachine-learning-with-pytorch
117. Cortés-Mendoza JM, Tchernykh A, Babenko M, Pulido-Gaytán LB, Radchenko G, Leprevost F, Wang X, Avetisyan A (2020) Privacy-preserving logistic regression as a cloud service based on residue number system. In: Voevodin V, Sobolev S (eds) Supercomputing. RuSCDays 2020. Communications in Computer and Information Science, vol 1331. Springer, Cham, pp 598–610. https://doi.org/10.1007/978-3-030-64616-5_51