SoK: Cryptography for Neural Networks

Monir Azraoui[1], Muhammad Bahram[2], Beyza Bozdemir[3], Sébastien Canard[1], Eleonora Ciceri[4], Orhan Ermis[3], Ramy Masalha[2], Marco Mosconi[4], Melek Önen[3], Marie Paindavoine[5], Boris Rozenberg[2], Bastien Vialla[1], and Sauro Vicini[4]

[1] Applied Crypto Group, Orange Labs, Caen, France, {monir.azraoui, sebastien.canard, bastien.vialla}@orange.com
[2] IBM Haifa, Israel, {muhammad, borisr, Ramy.Masalha}@il.ibm.com
[3] EURECOM, France {beyza.bozdemir, orhan.ermis, melek.onen}@eurecom.fr
[4] MediaClinics, Italia, {e.ciceri, m.mosconi, s.vicini}@mediaclinics.it
[5] Cybersecurity Research, Renault, Paris, France, marie.paindavoine@renault.com

Abstract. With the advent of big data technologies which bring better scalability and performance results, machine learning (ML) algorithms become affordable in several different applications and areas. The use of large volumes of data to obtain accurate predictions unfortunately come with a high cost in terms of privacy exposures. The underlying data are often personal or confidential and, therefore, need to be appropriately safeguarded. Given the cost of machine learning algorithms, these would need to be outsourced to third-party servers, and hence protection of the data becomes mandatory. While traditional data encryption solutions would not allow accessing the content of the data, these would, nevertheless, prevent third-party servers from executing the ML algorithms properly. The goal is, therefore, to come up with customized ML algorithms that would, by design, preserve the privacy of the processed data. Advanced cryptographic techniques such as fully homomorphic encryption or secure multi-party computation enable the execution of some operations over protected data and, therefore, can be considered as potential candidates for these algorithms. However, these techniques incur high computational and/or communication costs for some operations. In this paper, we propose a Systematization of Knowledge (SoK) whereby we analyze the tension between a particular ML technique, namely, neural networks (NN), and the characteristics of relevant cryptographic techniques.

Keywords: privacy, neural networks, homomorphic encryption, secure multi-party computation

1 Introduction

Artificial Intelligence (AI) is a generic term used to designate any system that is capable of learning and solving problems based on the perception of its environment. AI is today divided into several sub-fields, depending on the technical considerations, and, several tools related to AI are now capable of solving a lot of difficult problems related to computer science. These tools such as neural networks (including deep learning), Bayesian networks, or classifiers are well-known today. In this paper, we focus on neural networks (NN) that are inspired by the architecture of neurons in the human brain. NN have two modes of operations: a *training phase* (also called learning phase) in which the network learns a new capability from a training dataset and a *querying phase* (also known as prediction or classification phase), where this capability is tested over new data. The first phase takes as input the training dataset that permits the "neurons" to create their data model used in the second phase. While goal of NN is to learn from data, the European General Data Protection Regulation (GDPR) [21] aims to protect the data often considered as personal and hence, privacy sensitive. NN and GDPR cannot a priori live together, and several advanced cryptographic techniques are used to reconcile them.

Among these advanced cryptographic techniques, Secure Multi-Party Computation (MPC) [27] allows several parties to put in common their own input to obtain a unique output. While the latter can be made publicly available or kept private at the end of the protocol, each input should remain confidential at any time. Nowadays, several practical constructions exist [3, 27, 38, 59, 60], based on garbled circuits, secret sharing, or oblivious transfer, and some implementations are available. Another important cryptographic technique is homomorphic encryption. When only additions or multiplications are performed in the "encrypted world" using RSA [51], ElGamal [20] or Paillier [49], which is quite limited in practice, the possibility to have a "fully" homomorphic encryption (FHE), capable to perform both additions and multiplications in the encrypted world in an arbitrary manner, is quite recent. And since the first construction of FHE in 2009, several papers [8, 17, 16, 22] have appeared on this subject or the way to use such encryption schemes with practical algorithms [6, 10, 26, 54]. As for MPC, several implementations for FHE exist. In this case, the entity encrypting the data will necessarily be the one who will be able to decrypt the result.

The possibility to use advanced cryptographic techniques for AI has first been suggested in 2000 by Lindell and Pinkas [37], in the case of data mining and decision trees. Regarding neural networks, , to the best of our knowledge, the first paper on the subject is the work by [48] in 2007. However, by the second half of 2010s, things accelerated a lot and many papers have been published, either focusing on the training phase [34], or working on the classification phase [6, 26, 39, 46], which is the most frequent case as it is the easiest, but also the most useful one.

In this work, we propose the first Systematization of Knowledge (SoK) paper that compares the different approaches and results of privacy-preserving NNs based on advanced cryptographic techniques. As state-of-the-art non-linear NN layers (pooling and activation) are too complex to be directly executed in the encrypted world, there is a strong need for approximating them. We first compare the different strategies for these approximations, giving the ones we have found in the literature and the resulting accuracy. Then, this allows us to exhibit the NN operations that need to be performed on protected data (such as polynomial evaluation or comparison). Finally, we present a performance evaluation to compare the advanced cryptographic techniques on particular NN models designed for arrhythmia classification and image classification.

The rest of the paper is organized as follows. In the next section, we overview the Neural Networks operations and identify the main privacy and security requirements in our context. In Section 3, we compare MPC and FHE in their use to provide a privacy-preserving NN. We detail the solutions in Section 4. Section 5 presents a performance evaluation for two NN classifier based on underlying advanced cryptographic techniques. Finally, our conclusive remarks and some potential future works are given in Section 6.

2 Neural Networks

In this section, we briefly define neural networks and describe their underlying operations. We further discuss how these operations can be approximated so that cryptographic tools can support them.

2.1 Definition

A Neural Network (NN) is a particular case of machine learning techniques. It consists of several interconnected nodes called *neurons* that are structured in layers. Each neuron performs one operation depending on which layer it belongs to. The NN layers are described are as follows:

- **Convolution layer (optional):** The basic idea behind a convolution layer is to slide a filter, or kernel, over the original input to obtain information about the similarity between the chunk of the original image covered by the filter and the filter itself. On input a (ℓ_1, ℓ_2)-matrix $\mathbf{X}$ representing the data, and a smaller (ℓ'_1, ℓ'_2)-matrix K representing the kernel, the convolution function outputs a matrix $\mathbf{Y}$ as follows:

$$\mathbf{Y}[n, m] = \sum_{i=1}^{\ell'_1} \sum_{j=1}^{\ell'_2} \Big(K[i, j] \cdot \mathbf{X}[n + i - 1, m + j - 1] \Big).$$

  $\mathbf{Y}$ is a map corresponding to the filter K, which was slid over the image with a stride of 1. This function is executed for all filters considered in the layer to obtain multiple maps.
- **Activation layer:** The goal of the activation layer is to determine whether the pattern of a filter is actually present at a given position in the data. There are different kinds of activation functions. We consider the three following ones:
  - sigmoïd σ, as $\mathbf{y} = \frac{1}{1+e^{\mathbf{x}}}$;
  - hyperbolic tangent tanh as $\mathbf{y} = \frac{e^{\mathbf{x}} - e^{-\mathbf{x}}}{e^{\mathbf{x}} + e^{-\mathbf{x}}}$;

  - the rectified linear unit (ReLU)[6] as $\mathbf{y} = \mathsf{max}(0, \mathbf{x})$. When using the ReLU activation functions, it is highly recommended to use normalization by adding a batch normalization layer prior to each activation layer to obtain a stable and normalized distribution before the execution of the activation function [35].
- **Pooling:** The max pooling operations consists in reducing the spatial size of the input in order to make it more manageable.
- **Fully Connected layer:** The fully connected layer correlates the output of the previous layer with the features of each class.

2.2 Architecture

In a scenario, NN are outsourced to a third party server, we basically consider two actors:

- a client $\mathcal{C}$ having the input i.e. some data $\mathbf{X}$;
- a server $\mathcal{S}$ having already received a trained neural network model, denoted as M.

The main goal is to delegate the *querying* phase to $\mathcal{S}$. Hence, at the end of the process, $\mathcal{C}$ wants to obtain $\mathbf{R} = \mathsf{M}(\mathbf{X})$, where M consists of a set of the predefined functions. In order to ensure data privacy, namely the privacy of $\mathbf{X}$ against $\mathcal{S}$, some NN operations unfortunately cannot easily/efficiently be supported by cryptographic techniques. These operations should therefore be approximated into polynomial operations without having a significant impact on the accuracy of the overall NN. In the next section, we discuss the different approximation methods applied for each particular layer.

2.3 Approximation of NN layers

As previously mentioned, some NN layers contain nonlinear operations so that it may be hard to directly execute them on the encrypted input. The best idea is to approximate them to simplify, and then improve the efficiency of such execution without sacrificing from the NN accuracy.

- **Approximation of pooling functions:** As the max pooling function is not linear, the literature [26] suggests to approximate it by either summing up all values or computing their average.
- **Approximation of the activation function:** There are several ways to perform the activation function but, regarding the literature, the most efficient one in terms of efficiency and prediction accuracy to use square function, which directly computes x^2 for any given input x.

[6] The ReLU is currently the activation function that is mostly used. There are also some variants, such as the parametric version PReLU and the Exponential Linear Units ELU.

Operations: Regarding the description of the different main functions (directly or using an approximation) that should be executed during an NN evaluation (activation and pooling), we have extracted the main basic functions that should be operated on encrypted data. We have obtained the following results.

- **Addition**: activation, pooling, fully connected, and convolutional layer.
- **Multiplication**: activation, fully connected, and convolutional layer.
- **Polynomial evaluation**: activation.
- **Comparison** : activation, pooling.

2.4 Security Requirements

In order to identify the main privacy requirements, we first define the overall context where neural networks are used with privacy sensitive data. We therefore consider a scenario whereby an entity, such as a hospital (or an SME), is collecting or has already collected some data from some *data subjects* such as patients (or customers). The entity wishes to infer some information about some clinical diagnosis (or customer habits), or predict the diagnosis of the next patient (or the behavior of the next customer), using this collected data and the NN tool. This entity that is usually considered as the *data controller* will outsource both the data and the relevant computations for analyzing the data and performing the prediction to a powerful cloud server, which is defined as the *data processor*. The data owner or any authorized party can further query the model. In the sequel of this paper, this party is called the *querier*.

With GDPR and given the sensitiveness of the collected and processed data, there is a strong need for the data subjects to be protected. The first and foremost requirement to satisfy is to protect their data against unauthorized access by third parties during the entire lifetime of the data, i.e., from their collection until their analysis or even their deletion. Entities who can be considered as unauthorized to access are of three types: First, the *external parties* do not play any role in the collection, storage and analysis of the data. Second, the *cloud server* acts as the data processor and is considered as a third-party server that only provides storage and computational resources to the data controller. The cloud server is considered as a honest-but-curious adversary. Finally, the *data collector* collects the data (such as the hospital or the SME in the previous example) and can also be sometimes prevented from accessing the cleartext content.

In addition to the collected data, the query of the NN prediction/classification should remain private against unauthorized parties. The query (and sometimes the corresponding result) should not reveal any information to some potentially malicious adversaries. Those can be external parties who basically should not learn any information (neither the data nor the queries and results). Even if the cloud server can be a malicious party, it should be able to process the query without discovering any information about the data being processed. In some cases, the cloud server should even not learn the classification result.

Finally, in the context of neural networks, even the model can reveal some privacy sensitive information and therefore needs some protection. Similarly to the previous two information, the model should not be revealed to external parties. The model should also be protected against the querier or the data subjects as it is obtained based on all collected data. In some cases, the model should even remain private against the cloud server.

3 Cryptographic Techniques

This paper investigates the suitability of two advanced cryptographic techniques to neural networks, namely: fully homomorphic encryption (FHE) and secure multi-party computation (MPC), and overviews the state-of-the-art solutions that succeed in obtaining privacy preserving neural networks by applying some approximations on the underlying operations.

3.0.1 Multi-party computation Secure multi-party computation is introduced in early 1980s by Yao [59, 60] who focused on the two-party computation (2PC) case by defining Yao's Millionnaire problem. Then, by Goldreich et al. in [28], the problem was generalised to multiple parties.

Definition. Secure multi-party computation (MPC) is defined as a system in which a group of *data owners* can jointly compute a function of their private inputs without disclosing the underlying inputs, but the output of the function. Formally, let $P_1, \ldots, P_n$ be n parties and each of them having input $x_1, \ldots, x_n$, respectively. The parties $P_1, \ldots, P_n$ want to jointly compute the function f over all inputs $\{x_1, \ldots, x_n\}$ and learn the output without revealing their input.

MPC should ensure the following two properties, at least: (i) *input privacy*, i.e., parties' inputs should remain private and only the output of the function is learned; (ii) *correctness*, i.e., even if some parties misbehave, the correct output should be obtained.

Building Blocks Existing MPCs leverage Yao's Garbled circuits [59, 60] and secret sharing (additive or Boolean) [3]. We briefly explain each method in the following paragraphs. Before going into details, we first introduce Oblivious Transfer (OT) method.

Oblivious Transfer Oblivious transfer (OT) [50] is a fundamental cryptographic primitive that is used as a building block in MPC. OT allows a party to choose k out of n secrets from another party without disclosing which secrets have been chosen. Usually, the 1-out-of-2 OT is used, ensuring that one secret out of two of them is retrieved: Let Alice have two inputs x_0 and x_1, and Bob selects a bit b and wants to obtain x_b. OT ensures that Bob does not learn x_{1-b} and does not reveal b to Alice.

Yao's Protocol Yao's protocol (*a.k.a.* Garbled Circuits (GC)) is a secure two-party computation that allows the two parties to evaluate a function $f(x_1, x_2)$ in the presence of semi-honest adversary (i.e., this adversary has to truly follow the protocol yet s/he can try to extract information during the execution of protocol), where inputs x_1 and x_2 are provided by two parties, namely Alice and Bob. Let Alice be the *garbler* and Bob be the *evaluator*. Alice builds a garbled version of a circuit for the function f by obfuscating all possible outputs in the truth table. The garbled circuit and Alice's garbled input $GI(x_1)$ are sent to Bob. Alice also provides a map from the garbled-circuit outputs to the actual bit values. After receiving the circuit, Bob uses 1-out-of-2 OT [50] with Alice to obliviously obtain his garbled circuit values $GI(x_2)$ without revealing it to Alice. Bob further evaluates the function $f(x_1, x_2)$ using $GI(x_1)$ and $GI(x_2)$.

The function f is evaluated through a Boolean circuit. The *garbler* assigns two keys that correspond to bit values 0 or 1 for each wire of the circuit. Then, Alice, the garbler, computes four ciphertexts for each binary gate with the input wires and the output wire. After obtaining ciphertexts, Alice randomly orders these four outcoming values. The evaluator, Bob, can decrypt the correct row from the table if he successfully obtains the pair of keys from Alice via OT.

Secret Sharing Alternatively to Yao's Garbled Circuits, MPC solutions based on secret sharing consist of distributing secrets among parties involved in the system and further evaluate the function defined as a circuit accordingly. The GMW protocol [28] relies on Boolean shares and mainly support XOR operations over single bits. The function to be evaluated is encoded as a Boolean circuit and OT is used during the circuit evaluation. The Boolean circuit takes as inputs bit u from Alice and bit v from Bob. These bits are first secret-shared between the parties as $u = u_1 \oplus u_2$ and $v = v_1 \oplus v_2$, where share 1 belongs to Alice and share 2 to Bob. Then both parties evaluate the circuit gate by gate. For example, given shared values, an XOR gate with input bits u and v and output bit w is evaluated locally (i.e. without communication) by each party by computing $w_i = u_i \oplus v_i$. Value w can be retrieved by exchanging and XORing the shares.

Some other solutions use arithmetic circuit whereby inputs are additively shared and addition gates (respectively multiplication gates) correspond to XOR gates (resp. AND gates). It is worth mentioning that Liu et al. [40] extend the additive secret sharing by introducing the dot-product triplets used for computing the dot product of two secret vectors in a privacy-preserving way.

Some available implementations of MPC. Several practical open-source implementations for 2PC/MPC systems have been proposed in recent years. Some consist of high-level description languages and corresponding compilers used to specify the function to be securely evaluated and to translate it into a Boolean or arithmetic circuit; for example, Fairplay [42] and its extension to multiple parties, FairplayMP [4]. Other implementations offer libraries for MPC such as SCAPI [19]. Finally, some other solutions propose more comprehensive frameworks consisting of libraries, languages and their compilers, runtime environments and OT tools such as TASTY [33], ABY [18] or EMPtoolkit [58].

3.0.2 Fully homomorphic encryption Homomorphic encryption allows to process encrypted data without learning neither the input data nor the computation result. The data owner, Alice, can thus delegate some of her computation over sensitive data to a non-trusted party, Bob. Alice encrypts her data under her own public key, and sends the encryption to Bob. Once received, Bob can evaluate a circuit over Alice's inputs, obtaining a result still encrypted under Alice's public key. Bob sends the result back to Alice, who is the only party able to decrypt it. Formally, a homomorphic encryption scheme is composed of four procedures, defined as follows:

– KeyGen: the key generation procedure takes as input the security parameter λ and outputs the public-secret key pair $(\mathsf{pk}, \mathsf{sk})$.
– Enc: the encryption procedure uses the public key pk to transform a message M into a ciphertext c.
– Eval: the evaluation procedure takes as inputs a circuit $\mathcal{C}$, and ciphertexts $c_1, \ldots c_\ell$ such that $c_i = \mathsf{Enc}(M_i, \mathsf{pk})$. It outputs another ciphertext $c_{\text{res}} = \mathsf{Eval}(\mathcal{C}, (c_1, \ldots c_\ell))$.
– Dec: the decryption procedure uses the secret key sk to transform back a ciphertext c to the message M.

Homomorphic encryption has been known since many decades. The concept, initially called privacy homomorphism, has been introduced in 1978 by Rivest, Adleman and Dertouzos [51] and several "basic" schemes verifying this property followed: such as the well-known RSA [51] (multiplicatively), ElGamal [20] (multiplicatively, or additively, depending on the variant) and Paillier [49] (additively). In 2005, the Boneh-Goh-Nissim encryption [5] scheme was able to perform an arbitrary number of additions, and one single multiplication (hence becoming one of the first somewhat homomorphic encryption schemes). The first FHE scheme was proposed in 2009 by Craig Gentry [24] with an ingenious idea called bootstrapping. Further, since research on FHE has been prolific and new schemes have been proposed based on Gentry's first idea to improve the efficiency of the original but impractical schemes. Both somewhat and leveled homomorphic schemes include some noise in the ciphertexts. This noise grows throughout computations. A refreshing procedure, *bootstrapping*, can be added to manage the noise growth. The bootstrapping operation remains a bottleneck when evaluating circuits in the encrypted domain.

Three generations of lattice-based FHE While Gentry's work is today considered as the first FHE generation, the second generation has been marked by Brakerski and two important schemes were published in 2012: Brakerski-Gentry-Vaikuntanathan (BGV) [8] and Fan-Vercauteren (FV) [22], for which many optimizations have later been proposed. This second generation of FHE enjoys a huge efficiency increase in comparison to the first one [29, 30]. The third generation has started in 2013 with the Gentry-Sahai-Waters (GSW) scheme [25] and a different way to represent keys. However, this generation is less used, because existing optimizations are not compatible with such a kind of representation. Recently, the fourth generation has been introduced by the scheme named TFHE [17, 16], which is based on a mathematical object called a torus, allowing to have the advantage of both second and third generations. Today, existing implementations are mostly based on the second and fourth generations.

Alternative designs An entire line of work is dedicated to the FHE over the integers. These solutions rely on the Approximate Greatest Common Divisor

(AGCD) problem. They achieve mostly the same properties as LWE-based FHE. In [15], the authors proved that both AGCD and LWE problems are equivalent. Another line of work has designed FHE schemes based on the NTRU problem, such as LTV [41]. Finally, a scheme allowing to do computation directly on floating point numbers was introduced [14]. This scheme had major implications in applied homomorphic encryption by facilitating the implementation of algorithms having a lot of numerical computation such as neural networks.

Available implementations There are multiple implementations of (fully) homomorphic encryption. Most of them provide a leveled homomorphic scheme (without bootstrapping). The Simple Encrypted Arithmetic Library (SEAL)[7] is edited by Microsoft and provides an implementation of the FV scheme in C++. A Python version also exists, Pyfhel[8]. Unlike FV-NFLlib[9], which is based on the NFLlib library dedicated to lattice cryptography, SEAL does not require any dependencies. PALISADE[10] is a standalone library written in C++ that lets the user choose between four schemes: FV, BGV, LTV and Stehlé Steinfield. Finally $\Lambda \circ \lambda$[11] is a Haskell library that offers a refinement of BGV scheme. HElib[12] implements BGV in C++ and provides bootstrapping. FHEW[13] implements a fast bootstrapping procedure. Finally, TFHE[14] implements one of third generation of FHE schemes that features boostrapping under 0.1 seconds. A performance comparison between the most used libraries published in [44]

4 Existing Solutions

4.1 MPC-based privacy preserving NN solutions

We analyze the method of secure MPC-based privacy preserving neural network solutions. Most of the early solutions use the 2PC-based approach between two entities, namely the client and the server. One such example for 2PC-based solutions is SecureML [46] which aims at building a privacy preserving training and classification solution for neural networks using secure multiparty computation. SecureML uses the stochastic gradient descent method to build the model and supports secure arithmetic operations on shared decimal numbers. In SecureML, evaluation of ReLU using Garbled circuits and they further a secure computation of polynomial approximation for activation functions (i.e., *sigmoïd* and *softmax* functions) are provided. Another secure 2PC-based study is MiniONN proposed by Liu et al. [38] for providing privacy preserving convolutional neural networks (CNN). To ensure data privacy, MiniONN defines oblivious transformations for

[7] SEAL: https://www.microsoft.com/en-us/research/project/simple-encrypted-arithmetic-library/ [Last update: July 2019]

[8] Pyfhel: https://github.com/ibarrond/Pyfhel [Last update: Mars 2019]

[9] FVNFLlib: https://github.com/CryptoExperts/FV-NFLlib [Last update: July 2016]

[10] PALISADE: https://git.njit.edu/palisade/PALISADE [Last update: April 2019]

[11] https://hackage.haskell.org/package/lol

[12] HElib: https://github.com/shaih/HElib [Last update: May 2019]

[13] FHEW: https://github.com/lducas/FHEW [Last update: May 2017]

[14] TFHE: https://github.com/tfhe/tfhe [Last update: November 2017]

each CNN operation. Moreover, DeepSecure [52] relies on Yao's Garbled circuits to securely compute deep learning models. Rouhani et al. [52] use sigmoid and *tanh* as activation functions due to the optimization of Garbled circuits. Ball et al. [2] propose an extension to DeepSecure that is a secure evaluation of the NN classifier based on garbled circuits. Unlike DeepSecure, authors in [2] make use of arithmetic circuits instead of Boolean circuits and further use some improved techniques for the computation of matrix multiplication, activation function, and max-pooling. Chameleon proposed by Riazi et al. [53] is a hybrid protocol to securely compute function evaluation where two parties jointly perform a function without disclosing their inputs. Chameleon is called a hybrid framework since two parties can use Garbled circuits, the Goldreich-Micali-Wigderson (GMW) protocol, and arithmetic sharing. Similar to Chameleon, another solution EzPC [12] is a cryptographic cost-aware secure 2PC protocol generator and makes use of arithmetic and Boolean circuits for a secure NN classification. This scheme is useful for the one who does not have sufficient knowledge on the cryptographic techniques to compute a secure NN classification since EzPC takes source code as an input and outputs 2PC protocols.

Recently, MPC-based solutions also proposed for NN classification and training. ABY^3 proposed by Mohassel et al. [45] is an extension to SecureML. In SecureML, there are two non-colluding servers and clients share their private inputs among them whereas in ABY^3 there are three non-colluding servers; hence, arithmetic, Boolean and Yao's sharing are redefined across these three servers. SecureNN [56] presents secure a 3-party computation protocols for NN operations. This scheme supports both NN training and NN classification on convolutional neural networks based on MNIST dataset. SecureNN does not use garbled circuits and oblivious transfer to obtain performance gain in terms of the communication cost. The proposed protocols are secure against not only semi-honest adversary but also malicious adversary. Furthermore, a protocol proposed by Ohrimenko et al. [47] use trusted SGX processors for NN training.

4.2 FHE-based privacy preserving NN solutions

The first work leveraging homomorphic encryption for NN is CryptoNets [26]. The selected homomorphic encryption scheme is FV [22], without its bootstrapping procedure. In order to minimize the computation depth, they approximate the activation function by the square function, which only consumes one level of homomorphic evaluation. While effectively transforming a neural network into a FHE-friendly circuit, the square function is only a good approximation of the RELU on restricted distributions.

In [11], the authors introduce a batch normalization layer before each activation layer in order to stabilize the distribution and achieve better accuracy. Where CryptoNets gets an accuracy of 98.95 %, Chabanne et al.'s work [11] permits to obtain 99.30 %. Moreover, in both protocols, the number of neurons that can be handled is bound by the initially chosen parameters.

Encoding real numbers in a way that preserves the operations is a core problem that might affect performances. CryptoNets first convert the real number to a fixed precision number, and then embed it into a polynomial whose coefficients are its binary decomposition. The inverse mapping consists in evaluating the polynomial at 2. While neural networks operate on real numbers, with natural arithmetic, plaintext spaces for homomorphic encryption are finite field of polynomials (modular arithmetics). Therefore, different encoding methods have been proposed to operate with integers, as in the case of BGV [7] or FV [22].

In order to avoid this encoding step, the authors in [6] only consider discretized neural networks that directly operate on integers. They use a particularly restrictive form of neural network, Binarized Neural Networks, where weights are set in $\{1, -1\}$. In order to preserve this property, they select the `sign` function as approximation function, where negative integers are mapped to -1 and positive ones to 1. The underlying homomorphic scheme is TFHE [17, 16], with bootstrapping. They modified this scheme to compute both bootstrapping and the `sign` function at the same time. Due to the use of bootstrapping, their scheme can evaluate networks with arbitrary numbers of neurons. Although the use of bootstrapping allows a huge increase in the number of layers in the network, the discretization of the network incurs a non negligible loss of accuracy (obtaining “only” 96 %).

A recent work of [34] addresses the problem of both training and classification over encrypted data. Their solution follows the approach of approximating activation functions with low-degree polynomials by introducing a new method based on Chebyshev-like orthogonal polynomials. Besides the authors resort to homomorphic encryption (HElib) to encrypt both the inputs and (unlike previous work [26, 6]) the models. To handle the noise in the ciphertexts during the computations, they do not consider bootstrapping since it imposes a high computational overhead, but instead propose to have the server (i) check the noise level and (ii) ask the client to recrypt the ciphertext when the noise reaches a predefined threshold.

4.3 Hybrid Solution

The classification protocol of GAZELLE [36] combines FHE and MPC (via Garbled Circuits) to compute neural network classifications privately. Fully connected and convolutional layers are computed via FHE. Activation functions and max pooling layers are computed via MPC. Transitions between FHE and MPC are performed by each participant having an additive secret sharing of the intermediate result. This allows to take FHE with very low noise capacity (which results in efficient computation). Transitions between FHE and MPC act effectively as bootstrapping as they reset the noise. Another feature of these transitions is that computational and communication costs grow only linearly with network depth.

The transitions between FHE and MPC and the specialized algorithms for linear layers in FHE could be applied regardless of the given cryptographic primitives which realize FHE and MPC. However, another avenue of innovation for GAZELLE lies in the FHE implementation and parameter choice. In GAZELLE, the Brakerski-Fan-Vercauteren (BFV) scheme [23] is used.

GAZELLE is secure for semi-honest adversaries, that is, neither the server nor the client recovers any information if they follow the protocol. The protocol does not reveal the weights of each layer or their exact size. The performance of GAZELLE was evaluated by using the MNIST dataset. The neural network used in this evaluations consists of one convolution layer and two consecutive fully connected layers. The offline runtime is 0.15 ms, and online runtime is 0.05 ms (overall 0.2 ms), which is better than other approaches [38, 46].

5 Performance Study

In this section, we present a performance study that motivates the usage of neural networks in the context of health data and image processing. For this respect, we compare the performance of FHE-based, 2PC-based and Hybrid (GAZELLE-based) solutions for arrhythmia and image classification.

5.1 Arrhythmia Case Study

Heart arrhythmia is a set of conditions in which the heartbeat is not regular. Most types of arrhythmia a patient can be subjected to, are not causes for concern, as they neither cause damages to the heart nor make the patient experience symptoms. Unfortunately, several arrhythmia types cause symptoms that range from tolerable ones (e.g., lightheadedness) to more serious ones (e.g., short breath), and some others predispose patients to heart failure and stroke, resulting in grave consequences such as cardiac arrests. For this reason, it becomes vital to monitor chronic patients' ECG signals to identify arrhythmias at their onset, and prevent the aggravation of patients' conditions.

Nowadays, several commercial services that perform arrhythmia detection on ECG signals can be found in the market. These services collect patients' ECG data via dedicated wearable devices, analyse them to detect arrhythmias and report the results to a healthcare professional, who creates a report. As the identification of arrhythmia in this case is done by a machine, there is a need for building reliable and accurate algorithms for the identification of critical ECG sections. In this context, the algorithmic paradigm of deep learning represents a valid tool for improving the performance of automated ECG analysis [31].

Unfortunately, there are limitations to this approach. Indeed, the burden of analyzing long streams of ECG data for a large number of patients may be difficult to be handled on premises, where the potential lack of computational resources would limit the performance. To overcome this issue, one could acquire ECG data on premises and outsource them to an external environment (with more resources), where the arrhythmia detection would be performed. Nevertheless, moving from a trusted environment to an untrusted one would endanger the protection of personal data. This aspect becomes particularly critical when analyzing health-related data, as the most recent regulations on data protection (such as the GDPR) impose strict analysis constraints for the so-called *special categories of data* (as per Article 9). Hence, it becomes essential, in this case, to protect data before outsourcing them to the untrusted environment, e.g., via the use of advanced cryptographic techniques.

5.2 Image Classification Case Study

Image classification [32] is the study of processing an image and extract valuable information from its content. Image classification has various application areas that spans from face [13] or finger print [57] recognition for biometrics to video surveillance systems [55], hand gestures recognition for sign language [9], etc.

Classifying an image involves computationally intensive operations. With the recent developments in information systems, particularly with the rise of Graphical Processing Units (GPUs), the popularity of image classification has increased again for researchers in machine learning. Thus, many small and medium organizations started developing new applications based on the purpose of image classification for either providing better services for their customers such as face recognition for access control rather than using password based access control, or surveilling an area, building, room, etc. Although GPUs provide extra computation power for the processing of an image, such companies may require the help of computationally more powerful environments such as cloud servers. Companies again face with the dilemma mentioned in Section 5.1: outsourcing the images and the underlying image classification operations to an untrusted environment raise privacy issues since these images may contain sensitive information about individuals and more critically some malicious parties can gain access to various online systems using these individuals' images. Therefore, an extra layer of protection should be provided before outsourcing these images to the untrusted environment such as the use of advanced cryptographic techniques mentioned throughout the paper.

5.3 Performance Evaluation of Cryptographic Techniques on Arrhythmia Classification

5.3.1 The Neural Network Models: In order to evaluate the suitability and efficiency of the advanced cryptographic techniques mentioned in this paper, we propose a comparative study for neural network classification with the two previously described use cases, namely arrhythmia and image classification. To this aim, we build a small NN model for the arrhythmia classification and a deeper NN model for the image classification. These models are newly built in order to be compatible with the use of FHE and 2PC.

For the arrhythmia classification use case, the PhysioBank database[15] is employed for training and classification of the newly built NN model. The network consists of 2 fully-connected layers and 1 activation layer that uses x^2 with the input vector size 180, 40 hidden neurons and the output vector size 16 as defined in [43]. The model achieves 96.51% accuracy.

[15] https://www.physionet.org/physiobank/database/mitdb

For the image classification use case, the MNIST database[16] that consists of handwritten digits is used to construct the NN model. The organization of the layers in the model are as follows: one convolution layer with 5 different 5×5 filters that have $(2, 2)$ strides, one x^2 activation layer, a flatten layer, a fully connected layer with 100 neurons, another x^2 activation layer and finally a fully connected layer with 10 neurons. This model achieves 97.39% accuracy.

5.3.2 Privacy preserving classifiers: Once these NN models are designed, the goal is to execute them over protected inputs. We have developed three different solutions that are based on the use of FHE, 2PC or Hybrid (see Section 4) for both NN models.

For the FHE-based solution, we use the CKKS scheme implemented in Microsoft SEAL 3.1. The CKKS scheme allows making computations on floating-point numbers, directly. We choose a precision of 15 bits after the point to ensure that the accuracy of the encrypted evaluation is the same as its evaluation in the non-encrypted version for both NN models. The network weights are not encrypted. Therefore, we mostly use operations between plaintexts and ciphertexts, instead of ciphertext and ciphertext, which improve performances the solution. We choose $m = 4096$ and $q = 2^{116}$ as parameters for the scheme in order to ensure 128-bit security.

In the 2PC-based solution, we propose to use the ABY framework [18] to realize the operations of the proposed NN models such as additions and multiplications. In particular, we propose to use arithmetic circuits in the ABY framework since the majority of the underlying operations are linear (matrix multiplications) and there are no comparisons. Moreover, we approximate all real numbers into integers by using a simple truncation method that consists of keeping only some digits of the fractional part (hence by multiplying them with 10^n). The resulting circuit for privacy preserving arrhythmia classifier has depth 5 and 127 arithmetic gates and for privacy preserving image classifier, the circuit has depth 7 and 37685 arithmetic gates. Both classifiers also allow for prediction in batches thanks to the use of the SIMD packing method.

For the Hybrid solution, we follow the approach in [36] and use the so-called Gazelle technique. In other words, we have implemented the linear operations such as vector/matrix multiplication using FHE and the non-linear ones such as operations in activation layer by using MPC. On the other hand, we use HElib [1] as the homomorphic encryption library and BGV as the FHE tool. We also employ a truncation method to deal with the real numbers. This method is applied on the plaintext value before and after the classification such that floating point numbers are converted into integers before the classification and the resulting integer value is converted into floating point number after the classification.

5.3.3 Experimental Results: In this section, we present the experimental results to compare the performance of the FHE-based, 2PC-based, and Hybrid

[16] http://yann.lecun.com/exdb/mnist/

solutions on the arrhythmia and image classifications. All the simulations were carried out using a computer which has six 4.0 GHz Intel Core i7-7800X processors, 128 GB RAM and 1TB SSD disk. The experimental results are given in Table 1. We have performed two different tests for NN models: one classifying a single heartbeat/image and the other one classifying heartbeats in batches of 2048 heartbeats/images.

For the arrhythmia classification, the 2PC-based solution seems to provide the lowest computational cost when compared with the other solutions for the classification of a single heartbeat. However, the FHE-based solution outperforms when classification is performed in batches. Additionally, the FHE-based solution has better advantage in terms of the communication cost since all the computations in this solution are realized at the server and there is no need for interaction except for the transfer of the input. Moreover, the NN model of arrhythmia classification, which only involves linear operations and one square operation, the FHE-based solution seems to be the most suitable one. Nevertheless, this may not be the case for deeper neural networks such as for the case of image classification. For the image classification, the 2PC-based solution and the hybrid solution outperforms the FHE based solution for the classification of a single image. On the other hand, the FHE-based solution again provides better results for the classification in batches. However, these two NN models are designed to be implemented in all proposed solutions and therefore they do not consist of any non-linear operations such as the max pooling layer and ReLU activation function which are not supported by FHE. The hybrid solution may be the most appropriate one in such a case as it combines the use of 2PC and FHE. Indeed, this particular solution is specifically designed to operate on large NN models and it can be seen that the increase rate on the computational cost is lower than in the case of other solutions. Furthermore, the current version of the hybrid solution does not support packing yet.

Table 1: Performance Evaluation on ECG classification and image classification for FHE-based, 2PC-based and Hybrid solutions.

	FHE-based solution (with packing)		**2PC-based solution (with packing)**			**Hybrid solution (without packing)**		
	Comp. Cost (ms)	**Comm. Cost (MB)**	**Online Comp. Cost (ms)**	**Total Comp. Cost (ms)**	**Comm. Cost (MB)**	**Online Comp. Cost (ms)**	**Total Comp. Cost (ms)**	**Comm. Cost (MB)**
1 Heartbeat	1253	0.0018	25.7	212.947	1.85	43	5638	15.5
2048 Heartbeats	1253	3.69	3792.7	23092.4	3801	88064	11546624	31744
1 Image	13570	0.075	205.4	1083.2	3.5	1200	6500	264
2048 Image	13570	155	369878	776778.3	82015	4505600	13312000	540672

6 Summary

In this paper, we have presented the systematization of knowledge for privacy preserving Neural Network classifiers. We have first overviewed the NN operations and their approximations when needed for the cryptographic world. Later, we have introduced the definitions of advanced cryptographic techniques presented in this paper and with this aim, we have overviewed the existing solutions in the literature that utilize these techniques for NN classification. We further have developed particular NN models for the arrhythmia and image classification case studies, a small architecture for the first one and a deeper model for the second one and applied them on encrypted data using FHE-based, 2PC-based, and Hybrid solutions. We presented a performance evaluation to compare the FHE-based, 2PC-based, and Hybrid solutions. From the performance study, we can conclude that there is no single technique that outperforms. Moreover, in this study, the neural networks used are specifically selected by the compatibility with the FHE-based solution, which means these models do not consist of any non-linear operations. Therefore, for other usage scenarios, we may have to consider to use these operations, and this makes us deal with another dilemma between privacy, accuracy, and efficiency.

Acknowledgement

This work was partly supported by the PAPAYA project funded by the European Union's Horizon 2020 Research and Innovation Programme, under Grant Agreement no. 786767.

References

1. HElib An Implementation of homomorphic encryption. https://github.com/shaih/HElib
2. Ball, M., Carmer, B., Malkin, T., Rosulek, M., Schimanski, N.: Garbled neural networks are practical. Cryptology ePrint Archive, Report 2019/338 (2019), https://eprint.iacr.org/2019/338
3. Beaver, D.: Efficient multiparty protocols using circuit randomization. In: Advances in Cryptology – CRYPTO'91 (1992)
4. Ben-David, A., Nisan, N., Pinkas, B.: Fairplaymp: A system for secure multi-party computation. In: Proceedings of the 15th ACM Conference on Computer and Communications Security. pp. 257–266. CCS '08 (2008)
5. Boneh, D., Goh, E., Nissim, K.: Evaluating 2-dnf formulas on ciphertexts. In: Theory of Cryptography Conference, TCC 2005
6. Bourse, F., Minelli, M., Minihold, M., Paillier, P.: Fast homomorphic evaluation of deep discretized neural networks (2017)
7. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: Fully homomorphic encryption without bootstrapping. Cryptology ePrint Archive, Report 2011/277 (2011), http://eprint.iacr.org/2011/277
8. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (Leveled) fully homomorphic encryption without bootstrapping. In: ITCS 2012: 3rd Innovations in Theoretical Computer Science (2012)
9. Camgöz, N.C., Kındıroğlu, A.A., Akarun, L.: Sign language recognition for assisting the deaf in hospitals. In: Human Behavior Understanding. pp. 89–101 (2016)
10. Canard, S., Carpov, S., Nokam, D., Sirdey, R.: Running compression algorithms in the encrypted domain: a case-study on the homomorphic execution of RLE
11. Chabanne, H., de Wargny, A., Milgram, J., Morel, C., Prouff, E.: Privacy-preserving classification on deep neural network (2017)
12. Chandran, N., Gupta, D., Rastogi, A., Sharma, R., Tripathi, S.: EzPC: Programmable, efficient, and scalable secure two-party computation for machine learning. (IEEE EuroS&P 2019) (2019)

13. Chen, L.F., Liao, H.Y.M., Ko, M.T., Lin, J.C., Yu, G.J.: A new LDA-based face recognition system which can solve the small sample size problem. Pattern Recognition **33**(10), 1713 – 1726 (2000)
14. Cheon, J.H., Kim, A., Kim, M., Song, Y.S.: Homomorphic encryption for arithmetic of approximate numbers. In: ASIACRYPT 2017
15. Cheon, J.H., Stehlé, D.: Fully homomorphic encryption over the integers revisited. In: Advances in Cryptology – EUROCRYPT 2015, Part I (2015)
16. Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: Faster packed homomorphic operations and efficient circuit bootstrapping for TFHE. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017
17. Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In: Advances in Cryptology – ASIACRYPT 2016, Part I (2016)
18. Demmler, D., Schneider, T., Zohner, M.: ABY - A framework for efficient mixed-protocol secure two-party computation. In: ISOC Network and Distributed System Security Symposium – NDSS 2015 (2015)
19. Ejgenberg, Y., Farbstein, M., Levy, M., Lindell, Y.: Scapi: The secure computation application programming interface. Cryptology ePrint Archive, Report 2012/629 (2012), https://eprint.iacr.org/2012/629
20. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. In: Blakley, G.R., Chaum, D. (eds.) Advances in Cryptology – CRYPTO'84. Lecture Notes in Computer Science, vol. 196, pp. 10–18. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 19–23, 1984)
21. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation). Official Journal of the European Union **L119**, 1–88 (May 2016)
22. Fan, J., Vercauteren, F.: Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144 (2012), http://eprint.iacr.org/2012/144
23. Fan, J., Vercauteren, F.: Somewhat practical fully homomorphic encryption. IACR Cryptology ePrint Archive **2012**, 144 (2012), http://eprint.iacr.org/2012/144
24. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Mitzenmacher, M. (ed.) Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009
25. Gentry, C., Sahai, A., Waters, B.: Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In: Advances in Cryptology – CRYPTO 2013, Part I (2013)
26. Gilad-Bachrach, R., Dowlin, N., Laine, K., Lauter, K.E., Naehrig, M., Wernsing, J.: Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In: ICML 2016. vol. 48, pp. 201–210. JMLR.org (2016)
27. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or A completeness theorem for protocols with honest majority. In: 19th Annual ACM Symposium on Theory of Computing (1987)
28. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or A completeness theorem for protocols with honest majority. In: ACM Symposium on Theory of Computing (1987)
29. Halevi, S., Shoup, V.: Algorithms in HElib. In: Advances in Cryptology – CRYPTO 2014, Part I (2014)
30. Halevi, S., Shoup, V.: Bootstrapping for HElib. In: Advances in Cryptology – EUROCRYPT 2015, Part I (2015)

31. Hannun, A.Y., Rajpurkar, P., Haghpanahi, M., Tison, G.H., Bourn, C., Turakhia, M.P., Ng, A.Y.: Cardiologist-level arrhythmia detection and classification in ambulatory electrocardiograms using a deep neural network. Nature medicine **25**(1), 65 (2019)
32. Haralick, R.M., Shanmugam, K., Dinstein, I.: Textural features for image classification. IEEE Transactions on Systems, Man, and Cybernetics **SMC-3**(6), 610–621 (1973)
33. Henecka, W., Kögl, S., Sadeghi, A., Schneider, T., Wehrenberg, I.: TASTY: tool for automating secure two-party computations. In: ACM CCS (2010)
34. Hesamifard, E., Takabi, H., Ghasemi, M., Wright, R.N.: Privacy-preserving Machine Learning as a Service. Proceedings on Privacy Enhancing Technologies **3**, 123–142 (2018)
35. Ibarrondo, A., Önen, M.: Fhe-compatible batch normalization for privacy preserving deep learning. In: Data Privacy Management, Cryptocurrencies and Blockchain Technology (2018)
36. Juvekar, C., Vaikuntanathan, V., Chandrakasan, A.: Gazelle: A low latency framework for secure neural network inference. arXiv preprint arXiv:1801.05507 (2018)
37. Lindell, Y., Pinkas, B.: Privacy preserving data mining. In: Advances in Cryptology – CRYPTO 2000 (2000)
38. Liu, J., Juuti, M., Lu, Y., Asokan, N.: Oblivious neural network predictions via minionn transformations. In: ACM CCS 2017
39. Liu, J., Juuti, M., Lu, Y., Asokan, N.: Oblivious neural network predictions via MiniONN transformations. Cryptology ePrint Archive, Report 2017/452 (2017), http://eprint.iacr.org/2017/452
40. Liu, J., Juuti, M., Lu, Y., Asokan, N.: Oblivious neural network predictions via minionn transformations. In: 2017 ACM SIGSAC Conference on Computer and Communications Security. pp. 619–631. CCS '17 (2017)
41. López-Alt, A., Tromer, E., Vaikuntanathan, V.: On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In: 44th Annual ACM Symposium on Theory of Computing. pp. 1219–1234 (2012)
42. Malkhi, D., Nisan, N., Pinkas, B., Sella, Y.: Fairplay—a secure two-party computation system. In: USENIX Security Symposium (2004)
43. Mansouri, M., Bozdemir, B., Önen, M., Ermis, O.: PAC: Privacy-preserving arrhythmia classification with neural networks. In: FPS (2019)
44. Melchor, C.A., Killijian, M., Lefebvre, C., Ricosset, T.: A comparison of the homomorphic encryption libraries helib, SEAL and fv-nfllib. In: Innovative Security Solutions for Information Technology and Communications, SecITC 2018
45. Mohassel, P., Rindal, P.: ABY^3: A mixed protocol framework for machine learning. In: ACM CCS (2018)
46. Mohassel, P., Zhang, Y.: Secureml: A system for scalable privacy-preserving machine learning. In: IEEE Symposium on Security and Privacy. pp. 19–38. IEEE Computer Society (2017)
47. Ohrimenko, O., Schuster, F., Fournet, C., Mehta, A., Nowozin, S., Vaswani, K., Costa, M.: Oblivious multi-party machine learning on trusted processors. In: 25th USENIX Security Symposium (USENIX Security 16). pp. 619–636 (2016)
48. Orlandi, C., Piva, A., Barni, M.: Oblivious Neural Network Computing via Homomorphic Encryption. EURASIP Journal on Information Security (1) (2007)
49. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT '99 (1999)

50. Rabin, M.O.: How to exchange secrets with oblivious transfer. Cryptology ePrint Archive, Report 2005/187 (2005), https://eprint.iacr.org/2005/187
51. Rivest, R.L., Shamir, A., Adleman, L.M.: A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM **21**(2), 120–126 (1978)
52. Rouhani, B.D., Riazi, M.S., Koushanfar, F.: DeepSecure: scalable provably-secure deep learning. In: DAC 2018
53. Sadegh Riazi, M., Weinert, C., Tkachenko, O., Songhori, E.M., Schneider, T., Koushanfar, F.: Chameleon: A Hybrid Secure Computation Framework for Machine Learning Applications. ArXiv e-prints (Jan 2018)
54. Singh, K., Sirdey, R., Artiguenave, F., Cohen, D., Carpov, S.: Towards confidentiality-strengthened personalized genomic medicine embedding homomorphic cryptography. In: ICISSP 2017. pp. 325–333 (2017)
55. Srinivasan, S., Latchman, H., Shea, J., Wong, T., McNair, J.: Airborne traffic surveillance systems: Video surveillance of highway traffic. In: International Workshop on Video Surveillance & Sensor Networks (2004)
56. Wagh, S., Gupta, D., Chandran, N.: SecureNN: Efficient and private neural network training. In: (PETS 2019) (2019)
57. Wahab, A., Chin, S., Tan, E.: Novel approach to automated fingerprint recognition. IEE Proceedings - Vision, Image and Signal Processing **145**, 160–166(6) (1998)
58. Wang, X., Malozemoff, A.J., Katz, J.: Faster secure two-party computation in the single-execution setting. Cryptology ePrint Archive, Report 2016/762 (2016), https://eprint.iacr.org/2016/762
59. Yao, A.C.C.: Protocols for secure computations (extended abstract). In: 23rd Annual Symposium on Foundations of Computer Science. pp. 160–164 (1982)
60. Yao, A.C.C.: How to generate and exchange secrets (extended abstract). In: 27th Annual Symposium on Foundations of Computer Science. pp. 162–167 (1986)