Privacy-Preserving ECG Classification With Branching Programs and Neural Networks

Mauro Barni, *Senior Member, IEEE*, Pierluigi Failla, Riccardo Lazzeretti, Ahmad-Reza Sadeghi, and Thomas Schneider

***Abstract*—Privacy protection is a crucial problem in many biomedical signal processing applications. For this reason, particular attention has been given to the use of secure multiparty computation techniques for processing biomedical signals, whereby nontrusted parties are able to manipulate the signals although they are encrypted. This paper focuses on the development of a privacy preserving automatic diagnosis system whereby a remote server classifies a biomedical signal provided by the client without getting any information about the signal itself and the final result of the classification. Specifically, we present and compare two methods for the secure classification of electrocardiogram (ECG) signals: the former based on linear branching programs (a particular kind of decision tree) and the latter relying on neural networks. The paper deals with all the requirements and difficulties related to working with data that must stay encrypted during all the computation steps, including the necessity of working with fixed point arithmetic with no truncation while guaranteeing the same performance of a floating point implementation in the plain domain. A highly efficient version of the underlying cryptographic primitives is used, ensuring a good efficiency of the two proposed methods, from both a communication and computational complexity perspectives. The proposed systems prove that carrying out complex tasks like ECG classification in the encrypted domain efficiently is indeed possible in the semihonest model, paving the way to interesting future applications wherein privacy of signal owners is protected by applying high security standards.**

***Index Terms*—Linear branching programs, neural networks (NNs), privacy protection, quadratic discriminant function, secure biomedical systems, secure electrocardiogram (ECG) classification.**

I. Introduction

In THE last few years, increasing attention has been given to the development of tools for processing encrypted signals [1]. The reason for such an interest is rooted in the call for security stemming from applications where two or more nontrusted parties wish to collectively process one or more signals to reach a common goal. While the parties share the same goal, they do not trust each other hence they are not willing to disclose to the other parties the pieces of data they own, thus requiring that the signals are processed in a secure way, e.g., directly in encrypted form. In the simplest case, the above scenario consists of only two parties. One party, hereafter referred to as the Client ($\mathcal{C}$) owns a signal that has to be processed in some way by the other party, hereafter referred to as the Server ($\mathcal{S}$). Since $\mathcal{C}$ and $\mathcal{S}$ do not trust each other, $\mathcal{S}$ is required to process the signal owned by $\mathcal{C}$ without getting any information about it, not even the result of the processing. At the same time, $\mathcal{S}$ wants to protect the information it uses to process the signal provided by $\mathcal{C}$. While the above may seem a formidable, if not impossible, task, a bunch of cryptographic primitives exists, that once coupled with a suitable design of the underlying signal processing algorithms, allow us to process signals that have been *secured* in some way, e.g., (but not only) by encrypting them. In the recent scientific literature, such techniques are usually referred to as signal processing in the encrypted domain (s.p.e.d.), or secure signal processing (SSP) techniques, the latter term being preferable given that encryption is not the only way whereby signals can be *secured*.

The number of possible applications of SSP techniques is virtually endless. Among the most interesting scenarios investigated so far we mention: private database access [2], in which the client accesses a server by means of an encrypted query; private data mining [3], in which two or more parties wish to extract aggregate information from a dataset formed by the union of their private data; secure processing of biometric data [4], in which biometric signals are processed in the encrypted domain to protect the privacy of the owners; watermarking of encrypted signals [5], for digital rights management within buyer–seller protocols; recommender systems [6], in which user's data is analyzed without disclosing it; and privacy-preserving processing of medical data [7], in which sensitive medical data is processed by a nontrusted party, for remote medical diagnosis or any other form of home-care system whereby health conditions are monitored remotely.

The use of SSP for the processing of medical signals is surely one of the most promising applications among those listed above. As a matter of fact, the health-care industry is moving faster than ever toward technologies offering personalized online self-service, medical error reduction, customer data collection, and more. Such technologies have the potentiality of revolutionizing the way medical data is stored, processed,

Manuscript received August 10, 2010; revised December 30, 2010; accepted January 17, 2011. Date of publication January 28, 2011; date of current version May 18, 2011. The work of M. Barni, P. Failla, and R. Lazzeretti was supported by EU FP6 project SPEED and by MIUR under project Priv-Ware (Contract 2007JXH7ET). The work of A.-R. Sadeghi and T. Schneider was supported by EU FP6 project SPEED and by EU FP7 projects CACE and ECRYPT II. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Elisa Bertino.

M. Barni and R. Lazzeretti are with the Department of Information Engineering, University of Siena, Siena 53100, Italy (e-mail: barni@dii.unisi.it; riccardo.lazzeretti@gmail.com).

P. Failla is with the Research and Advanced System Design Group, Elt—Elettronica SpA, Rome 00131, Italy (e-mail: pierluigi.failla@gmail.com).

A.-R. Sadeghi is with CASED (TU Darmstadt and Fraunhofer SIT), 64293 Darmstadt, Germany (e-mail: ahmad.sadeghi@cased.de).

T. Schneider is with CASED, TU Darmstadt, 64293 Darmstadt, Germany (e-mail: thomas.schneider@cased.de).

Digital Object Identifier 10.1109/TIFS.2011.2108650

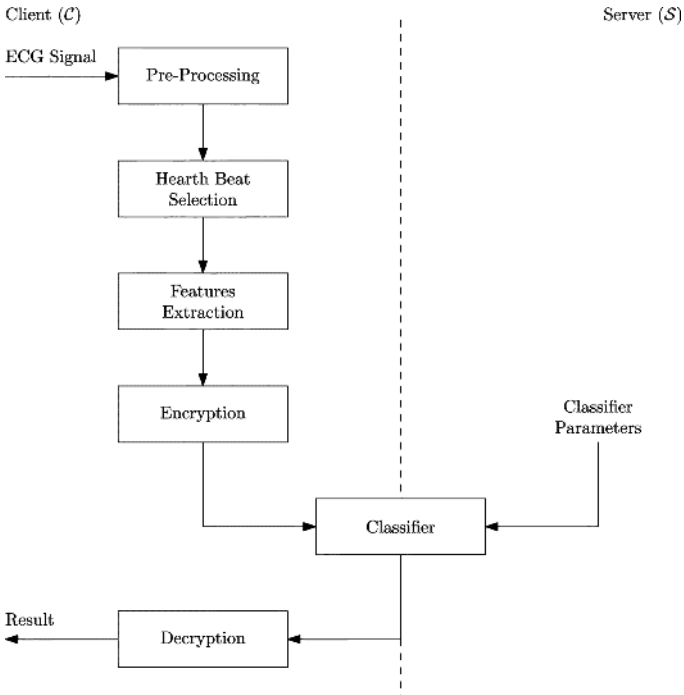

Fig. 1. SSP framework for ECG classification.

delivered, and made available in an ubiquitous and seamless way to millions of users throughout the world. In this framework, respecting the privacy of customers is a central problem, since privacy concerns may impede, or at least slow down, the diffusion of new e-health services. This is the case, for example, of on-line repositories of medical data (including signals) managed by a third party [8], [9]. Would anybody be willing to store his/her medical data in such repositories if his/her privacy rights are not adequately protected?

A. Addressed Scenario

In this work, we consider a scenario where a remote diagnosis system provided by a nontrusted party offers a service whereby biomedical signals are processed to provide a preliminary diagnosis. Such a system may either be seen as a stand-alone service or as part of a more complex e-health system where the service provider, in addition to hosting a repository of personal medical data, is also allowed to process such data. In order to preserve the privacy of the users, $\mathcal{S}$ should carry out its task without getting any knowledge about the private data provided by the users. At the same time, $\mathcal{S}$ may not be willing to disclose the algorithms it is using to process the signals, since they represent the basis for the service it is providing.

More specifically, we consider the privacy-preserving classification of electrocardiogram (ECG) signals in the semihonest model. Classification of ECG signals has long been studied by the signal processing community [10] and many good algorithms exist for this purpose; however, their efficient implementation in an SSP framework is not an easy task since many of the involved operations, while trivial in the plain domain, are very difficult to implement in a secure way. The functional description of the system addressed in our research is depicted in Fig. 1. Given an ECG signal, $\mathcal{C}$ performs some preprocessing to remove noise and clean up the samples. Subsequently, the system classifies ECG portions corresponding to single heart beats into six possible classes: five possible diseases and one healthy state (see Section II). To do so, each heart beat needs to be identified and then processed to extract a set of features allowing the subsequent classification. $\mathcal{C}$ encrypts the features and starts a classification protocol, interacting with $\mathcal{S}$. The classifier uses two inputs: $\mathcal{C}$'s features and a set of classification parameters provided by $\mathcal{S}$. On one side $\mathcal{C}$ does not want to reveal the ECG features since they are sensible information that must be kept secret. On the other side, $\mathcal{S}$ does not want to reveal the classification parameters. At the end of the protocol, $\mathcal{C}$ only learns the classification result but nothing about $\mathcal{S}$'s classification parameters which are valuable intellectual property of $\mathcal{S}$, while $\mathcal{S}$ learns nothing.

B. Contribution

A protocol implementing the functionality described in Fig. 1 could be developed by resorting to generic secure two-party computation (STPC) techniques [11], [12] allowing two parties to compute the output of a public function $f(\cdot)$ on their respective private inputs. At the end of the protocol, the only information obtained by the parties is the output of the function $f(\cdot)$ evaluated on the inputs, but no additional information about the other party's input. Specifically, we should consider a variant of the above case, where the function $f(\cdot)$ itself, or at least its parameters, has to be kept secret as well. While this can be reduced to secure evaluation of a public function using universal circuits [13], [14], this generic approach poses an enormous overhead on the protocols, hence calling for the design of efficient dedicated solutions. In this framework, the contribution of this paper is threefold. First, we present two SSP protocols for the classification of ECG signals, built upon the classification algorithm described in [15], [16]. In particular, the SSP classifier relies on a subset of the features proposed in [15] and [16] followed by either a quadratic discriminant function (QDF) classifier as in the original paper, or by a neural network (NN). The former protocol is very similar to the one presented in [7] and [17], the only differences being in the use of some of the efficient building blocks described in [18] and the garbled circuit construction of [19] to improve the overall efficiency of the protocol. The second protocol is completely new and relies on the protocol for secure NN computation described in [14]. Both protocols are designed within the framework of [20] which combines the advantages offered by homomorphic encryption (HE) [21] and garbled circuits (GC) [11]. We give a summary of this framework and its building blocks later in Section IV.

As a second contribution, we investigate the relationship between the representation accuracy of the to-be-processed signals (i.e., the number of bits representing the ECG features), the complexity of the proposed protocols, and the classification accuracy. This is a crucial step where signal processing domain knowledge must be used to ease the SSP implementation of the classifier in terms of efficiency and reduced complexity.

Finally, as a third contribution, we compare the two proposed protocols from a complexity point of view, getting interesting insights about the suitability of QDF-based and NN-based classification for efficient implementation in an SSP framework.

The rest of this paper is organized as follows. In Section II, we describe the plain version of the ECG classification algorithm, by distinguishing between the QDF and NN implementations and by paying attention to define the corresponding security requirements. In Section III, we describe how the classifiers must be quantized in order to enable the implementation in an SSP framework and analyze the trade-off between representation length and classification accuracy. The analysis regarding the NN implementation is one of the main contributions of the paper since the impact of quantization on protocols for SFE evaluation of NNs has never been carried out. In Section IV, we introduce the framework and the cryptographic primitives the proposed protocols rely on. In Section V, we describe the two protocols we have developed, analyze their efficiency, and compare them from a complexity perspective Section V-B contains the main innovation of the paper, while Section V-A presents an improved version of the protocols already described in [17]. In Section V-C, the LBP and NN approaches are compared from a complexity point of view, providing further insights into the merits and drawbacks of the two different protocols. Finally, in Section VI, we draw some conclusions by paying attention to derive some general hints on the suitability of the two classifier structures for SSP applications.

II. Two Approaches to Classify ECG

Classification of ECG signals has long been studied by the signal processing community; however, the interest in privacy-preserving classification has been raised only very recently [7]. In this section, we describe the plain version of the classification algorithm. First of all, we describe the features used by the classifier, then we examine two different approaches to classification. We are interested in classifying each heart beat according to the following six possible classes:

NSR	normal sinus rhythm (healthy state);
APC	atrial premature contraction;
PVC	premature ventricular contraction;
VF	ventricular fibrillation;
VT	ventricular tachycardia;
SVT	supraventricular tachycardia.

The classification algorithm we use relies on autoregressive (AR) modeling of the ECG signal, and is inspired by the work of Ge *et al.* [15]. The choice of this algorithm is justified first of all by the good classification accuracy it ensures, second because it fits well the requirements of a privacy preserving implementation, and finally because of its generality. Indeed, AR models are often used in automatic medical diagnosis as feature extractors, due to the ability of the coefficients of the AR models to represent the signal they approximate [22].

The overall architecture of the classifier is summarized by the block diagram in Fig. 2. The input of the system is an ECG chunk corresponding to a single heart beat, that, consequently, is classified as an independent entity. For the extraction of heart beats, we used the algorithm proposed in [15]. We assume that the ECG signal is sampled at 250 samples per second and that 300 samples surrounding each peak are fed to the system: 100 samples preceding the beat peak and 200 following it. We also assume that the ECG signal has been prefiltered by a notch filter to remove the noise due to power line interference and base line wander [15].

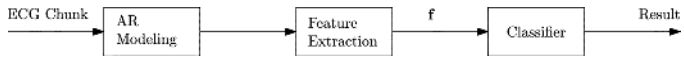

Fig. 2. General view of the ECG classifier.

Fig. 3. LBP classifier based on QDF and decision tree.

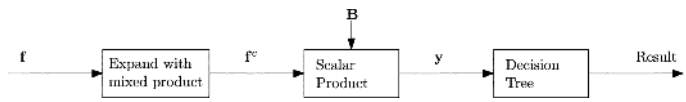

Fig. 4. Expanded LBP classifier.

Each ECG chunk is modeled by means of a fourth-order AR model. The AR model coefficients can be estimated in several ways; in our system, we used a method based upon the Yule–Walker equations [23]. The four coefficients of the AR model form the feature vector $\mathbf{f} = (f_1, f_2, f_3, f_4)^T$ used to classify each heart beat (where $(\cdot)^T$ denotes the vector transpose operator). In [15], six features were used; however, our experiments have shown that by using four features we obtain almost the same classification accuracy at a considerably lower complexity. Specifically, as described in [7], by using four features, the accuracy passes from 88.57% to 86.30%. For the above reason, and in order to keep our discussion simple, in the rest of the paper we will assume that only four features are used by the classifier. It goes without saying that whenever a loss of accuracy of 2% cannot be afforded, the complete set of features should be used: in this case, the classifier will have to rely on six features; however, most of the analysis we will present in the rest of the paper would remain the same, the only difference being a quantitative one in terms of computational complexity (see the discussion at the end of Section V-C).

A. Linear Branching Program Classifier

Once the four AR-features have been extracted, we must use them for the actual classification of the heart beat they refer to. The first approach we used to classify the feature vector $\mathbf{f}$ is by means of a quadratic discriminant function (QDF) followed by a decision tree as shown in Fig. 3. To cast the above approach into a linear framework (see Fig. 4), we introduce a composite feature vector $\mathbf{f}^c$ containing the features in $\mathbf{f}$, their square values, and their cross products, namely

$$\begin{aligned}\mathbf{f}^c &= (f_1^c, \ldots, f_{15}^c)^T \\ &= \Big(1, f_1, f_2, f_3, f_4, f_1^2, f_2^2, f_3^2, f_4^2, f_1 f_2, \\ &\qquad f_1 f_3, f_1 f_4, f_2 f_3, f_2 f_4, f_3 f_4\Big)^T.\end{aligned}$$

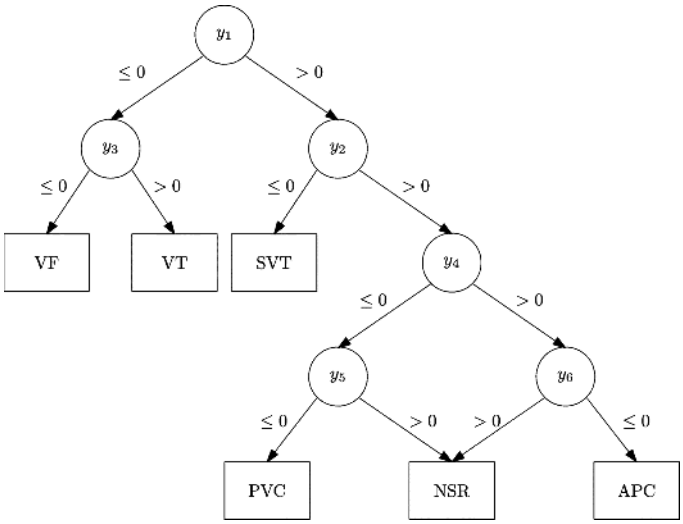

Fig. 5. Binary decision tree for ECG classification.

The vector $\mathbf{f}^c$ represents the input of the scalar product block shown in Fig. 4. The scalar product projects $\mathbf{f}^c$ onto six directions $\boldsymbol{\beta}_i$, obtaining a six-component vector $\mathbf{y}$, that represents the input of the final classification step: $\mathbf{y} = (y_1, \ldots, y_6)^T = \mathbf{B}\mathbf{f}^c$, where $\mathbf{B}$ is a matrix whose rows are the vectors $\boldsymbol{\beta}_i$. The matrix $\mathbf{B}$ contains part of the knowledge embedded within the classification system, and is computed by relying on a set of training ECGs (see [15] for details). For the final classification, the signs of the values y_i are extracted and used to actually classify the ECG, by means of the binary decision tree given in Fig. 5. A classifier having the structure described above can be represented by means of a linear branching program (LBP), i.e., a decision tree in which the decision path is decided according to the value assumed by the projection of the input features onto a given set of projection vectors (see [17] and Section V for further details on LBPs).

B. Neural Network Classifier

The second solution we developed relies on artificial neural networks (NNs). In fact, NNs are well-know machine learning structures used in many different fields ranging from approximation to classification. NNs are widely used as classifiers and, in general, they give good results if the training set used to train the network is representative of all the considered classes and the generalization grade is good enough (see [24] or [25]).

Finding the right topology for an NN is not a simple task due to the fact that NNs have several degrees of freedom including: number of hidden layers, neurons per hidden layer, and form of activation functions. In most cases, a two-layer NN is sufficient to obtain a good classification, so in the rest of the paper we focus on NNs with two layers, that is NNs in which the inputs are connected to a hidden layer, that, in turn, is connected to the output layer.

Before going on, in order to ease the description of the SSP protocol based on an NN classifier, we now review the details of the operations carried out by an NN. Generally speaking, each neuron in an NN performs only two simple operations: a scalar product and a function evaluation. As shown in Fig. 6, a single

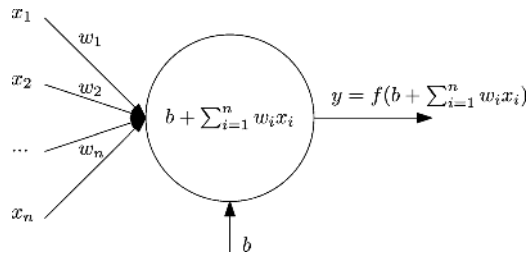

Fig. 6. A perceptron.

neuron, or *perceptron*, consists of a number of weighted connections $\mathbf{w} = (w_1, w_2, \ldots, w_n)^T$, a bias b and an activation function f. When a new input vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T$ is provided, the perceptron performs a scalar product among the weights and the input vector and adds up the bias

$$b + \sum_{i=1}^{n} w_i \, x_i = b + \mathbf{w}^T \mathbf{x} \tag{1}$$

after this, the activation function is applied producing the neuron output: $y = f(b + \mathbf{w}^T \, \mathbf{x})$.

The composition of multiple perceptrons in a cascade of layers realizes an NN. In the rest of the paper, we will use the following notation referring to a general two layer NN:

- n is the number of inputs of the NN, by this $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T$ is the input vector;
- n_h is the number of neurons in the hidden layer (the first layer of the NN);
- n_o is the number of neurons in the output layer;
- $\mathbf{W}_h$ is a matrix of size $n_h \times n$ whose elements are the weights of the connections between the inputs and the hidden layer, that is: $\mathbf{W}_h(i,j) = w_{h;i,j}$ is the coefficient used to weight the connection between the jth input and the ith node of the hidden layer;
- $\mathbf{W}_o$ is a matrix of size $n_o \times n_h$ whose elements are the weights of the connections between the hidden and the output layers, that is: $\mathbf{W}_o(i,j) = w_{o;i,j}$ is the coefficient used to weight the connection between the output of the jth node of the hidden layer and the ith node of the output layer;
- $\mathbf{y}_h$ is a vector of length n_h with the outputs of the neurons of the hidden layer;
- $\mathbf{y}_o$ is a vector of length n_o with the outputs of the neurons of the output layer;
- $\mathbf{b}_h$ is a vector of length n_h that contains the biases of the neurons of the hidden layer;
- $\mathbf{b}_o$ is a vector of length n_o that contains the biases of the neurons of the output layer.

With the above notation, the output of the ith neuron in the hidden layer is

$$y_{h;i} = f\left(b_{h;i} + \sum_{j=1}^{n} w_{h;i,j} x_j\right) \tag{2}$$

while the output of the entire hidden layer can be written in matrix form in the following way:

$$\mathbf{y}_h = f(\mathbf{b}_h + \mathbf{W}_h \, \mathbf{x}) \tag{3}$$

where the activation function f is applied component-wise to all the values of the input vector. Similarly, the output of the kth neuron of the output layer is

$$y_{o;k} = b_{o;k} + \sum_{i=1}^{n_h} w_{o;k,i} y_{h;i} \tag{4}$$

that in matrix form can be written as

$$\mathbf{y}_o = \mathbf{b}_o + \mathbf{W}_o \mathbf{y}_h \tag{5}$$

and $\mathbf{y}_o$ is the output of the NN. Note that the neurons of the output layer do not apply any activation function.[1]

In our scenario, the number of inputs is dictated by the number of features the classifier relies on, while the number of output layers corresponds to the number of diseases the NN should distinguish, so we have $n = 4$ and $n_o = 6$. The only degree of freedom we have, then, is n_h, the number of neurons in the hidden layer. To choose n_h, we carried out some tests trying to reach the same accuracy provided by the LBP-based classifier. A training set was built by using as samples the pair $(\mathbf{f}, \mathbf{o})$, where, as said before, $\mathbf{f} = (f_1, f_2, f_3, f_4)^T$ and $\mathbf{o}$ is a six-component vector, having value equal to 1 for the index of the class the ECG signal belongs to and 0 elsewhere.[2] In our experiments, we used a dataset of 200 ECG signals (and the corresponding 200 feature vectors) taken from PhysioBank [26]. Specifically, we split the dataset into a training set (containing 140 ECG sequences) and a test set (with the remaining 60 signals). While the size of the dataset may not be realistic for real life applications, we decided to use it since this dataset is often used in relevant literature on ECG classification; for instance, it is the same used in [15]. As it will be clear from the following discussion, the size of the dataset does not have a direct impact on the structure of the proposed protocols (while it surely has a great impact on the training phase); however, it is possible that for larger datasets a larger number of features is needed thus impacting on the complexity of the overall protocols.

As activation function we used a symmetric saturating linear function, defined by

$$f(x) = \mathsf{SATLIN}(x) = \begin{cases} 1, & \text{if } x > 1 \\ x, & \text{if } -1 < x < 1 \\ -1, & \text{if } x < -1. \end{cases} \tag{6}$$

Training was performed by using the Levenberg–Marquardt regularized learning algorithm [27]. Using this setup and 100 epochs for training, we obtained the classification accuracy shown in Fig. 7. Considering the results of Fig. 7, an NN with $n_h = 6$ neurons in the hidden layer is the smallest NN giving a classification accuracy larger than 86.30%, hence justifying the choice of an NN with the topology indicated in Fig. 8 (for the sake of simplicity the figure shows just a part of the involved variables). Note that the final classification is obtained by choosing the maximum among the outputs of the NN. A compact representation of the NN shown in Fig. 8 is given in Fig. 9 where the entire flow needed to evaluate the NN is shown. In particular, we recall that in formulas we have

$$\mathbf{y}_o = \mathbf{b}_o + \mathbf{W}_o\, \mathsf{SATLIN}(\mathbf{b}_h + \mathbf{W}_h\, \mathbf{f}) \tag{7}$$

and finally the classification result is computed as

$$o = \arg\max \mathbf{y}_o. \tag{8}$$

[1]More formally, the activation function of the neurons of the output layer is the identity function.

[2]This kind of neural network is often called NN with *fired* output.

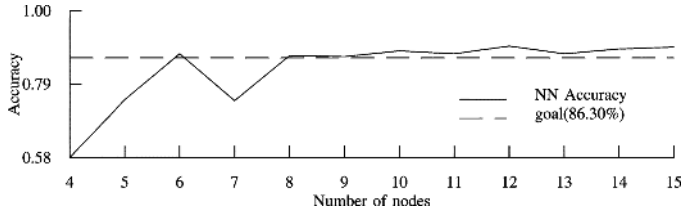

Fig. 7. Classification accuracy as a function of the number of nodes in the hidden layer and SATLIN as activation function.

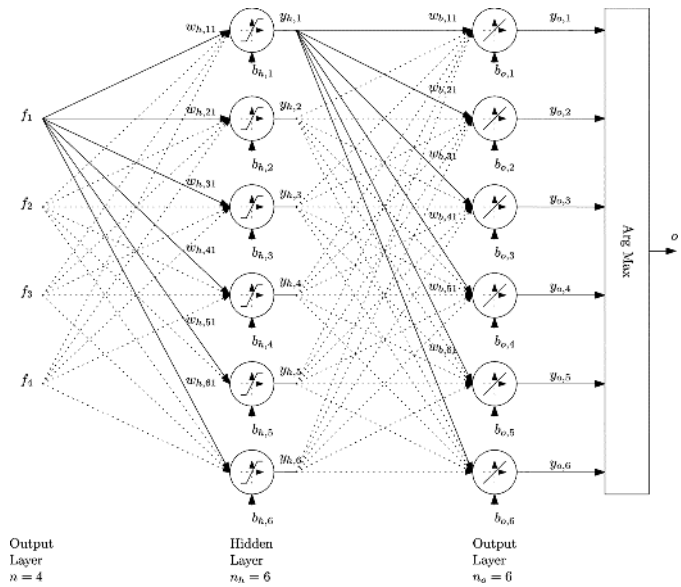

Fig. 8. NN topology.

III. Quantized Classifiers

In this section, we introduce the *quantized* version of the classifiers described so far. By quantized classifiers we mean a version of the classifiers that works only with integer numbers. This is a necessary step since the cryptosystem underlying any SSP protocol can handle only integer numbers. For this reason, we must adapt the algorithms described in Section II to let them work with integer numbers. To do so, we first define the quantized classifiers rigorously by leaving undefined the number of bits that are used to represent the inputs and the classifier parameters; then we describe the experiments we carried out to determine the exact number of bits that are needed to mimic a floating point implementation of the classifiers from the point of view of classification accuracy.

Passing from an algorithm implemented in floating point arithmetic to one working with integer numbers (fixed point arithmetic with no truncation) requires that the inputs and the parameters defining the LBP and the NN are quantized and

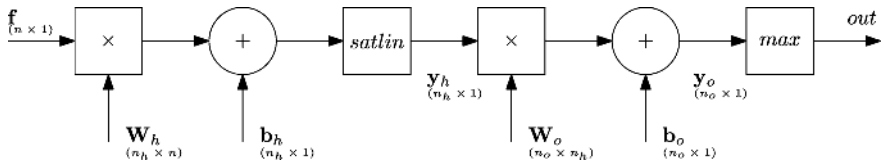

Fig. 9. Chain blocks to compute the output of an NN.

represented by a suitable number of bits, so that the final output does not differ significantly from the results that would have been obtained with a floating point implementation. Generally speaking, given a positive floating point number a, we can construct its quantized version multiplying it by a positive integer value q and rounding the result to the nearest integer, as specified by the following mapping:

$$a \rightarrow a^{\ell} = \lfloor qa \rceil \tag{9}$$

where we have indicated explicitly that $\lfloor qa \rceil$ is an integer number that requires $\ell = \lceil \log_2 \lfloor qa \rceil \rceil$ bits to be represented.

A. Quantized Linear Branching Program Classifier

We start by giving a formal definition of quantized LBP. The LBP is a natural generalization of binary classification trees and ordered binary decision diagrams (OBDDs). Compared to the above, LBPs have a more general branching condition that depends on the comparison of a linear combination of the inputs with a given threshold. More formally, a quantized LPB is defined as follows [17].

Definition 3.1 (Linear Branching Program (LBP)): Let $\mathbf{x}^{\ell} = \left(x_1^{\ell}, \ldots, x_n^{\ell}\right)^T$ be the **attribute vector** consisting of signed ℓ-bit integer values. A binary **linear branching program (LBP)** $\mathcal{L}$ is a triple $\langle \{P_1, \ldots, P_z\}, \text{Left}, \text{Right} \rangle$. The first element is a set of z nodes consisting of d **decision nodes** $P_1, \ldots, P_d$ followed by $z-d$ **classification nodes** $P_{d+1}, \ldots, P_z$. Decision nodes P_i, $1 \leq i \leq d$ are the internal nodes of the LBP. Each $P_i := \left\langle \mathbf{a}_i^{\ell}, t_i^{\ell'} \right\rangle$ is a pair, where $\mathbf{a}_i^{\ell} = \left(a_{i,1}^{\ell}, \ldots, a_{i,n}^{\ell}\right)^T$ is the **linear combination vector** consisting of n signed ℓ-bit integer values and $t_i^{\ell'}$ is the signed $\ell' = (2\ell + \lceil \log_2 n \rceil - 1)$-bit integer **threshold** value with which $\left(\mathbf{a}_{\mathbf{i}}^{\ell}\right)^T \mathbf{x}^{\ell} = \sum_{j=1}^{n} a_{i,j}^{\ell} x_j^{\ell}$ is compared in this node. $\text{Left}(i)$ is the index of the next node if $\left(\mathbf{a}_{\mathbf{i}}^{\ell}\right)^T \mathbf{x}^{\ell} \leq t_i^{\ell'}$; $\text{Right}(i)$ is the index of the next node if $\left(\mathbf{a}_{\mathbf{i}}^{\ell}\right)^T \mathbf{x}^{\ell} > t_i^{\ell'}$. The functions $\text{Left}(\cdot)$ and $\text{Right}(\cdot)$ are such that the resulting directed graph is acyclic. Classification nodes $P_j := \langle c_j \rangle$, $d < j \leq z$ are the leaf nodes of the LBP consisting of a single classification label c_j each.

Evaluation of the LBP $\mathcal{L}$ on an attribute vector $\mathbf{x}^{\ell}$ proceeds as follows. We start with the first decision node P_1. If $\left(\mathbf{a}_{\mathbf{1}}^{\ell}\right)^T \mathbf{x}^{\ell} > t_1^{\ell'}$, move to node $\text{Left}(1)$, else to $\text{Right}(1)$. Repeat this process recursively (with corresponding $\mathbf{a}_{\mathbf{i}}^{\ell}$ and $t_i^{\ell'}$), until reaching one of the classification nodes and obtaining the classification $c = \mathcal{L}(\mathbf{x}^{\ell})$.

It is evident how the LBP structure defined above can be used to implement the QDF classifier described in Section II-A on integer values. In our specific case, we have $\mathbf{a}_i = \boldsymbol{\beta}_i^T$, $\mathbf{x} = \mathbf{f}^c$, and $t_i = 0$. As to quantization, the situation is rather simple, since the size in bits of the intermediate values of the computations do not increase when the tree representing the LBP is traversed, hence we only need to define the number of bits necessary to represent the components of the feature vector and the elements of the projection matrix $\mathbf{B}$, i.e., we have to determine the value of ℓ that ensures the same classification accuracy of a floating point implementation of the LBP (see Section III-C).

B. Quantized Neural Network Classifier

We now consider the quantized version of the NN classifier. In an NN there are several parameters that need to be quantized, thus we introduce q_i, q_h, q_o that are the multipliers used to quantize the inputs of the NN, and the parameters (weights and biases) of the hidden and the output layers, respectively. We also define ℓ^i, ℓ^h, and ℓ^o, respectively, as the number of bits needed to represent the quantized version of the inputs and the quantized parameters of the hidden and output layers, including the sign bits.

In the following, we will use the notation introduced in Section II-B. Working with quantized values, the quantized output vector $\mathbf{y}_{h_q}$ of the hidden layer is

$$\mathbf{y}_{h_q} = \mathsf{QSATLIN}\left(\lfloor q_i q_h \mathbf{b}_h \rceil + \lfloor q_i q_h \mathbf{W}_h \, \mathbf{f} \rceil\right) \tag{10}$$

where the biases $\mathbf{b}_h$ have been multiplied by both q_i and q_h to make the bias homogeneous with the term $q_i q_h \mathbf{W}_h \, \mathbf{f}$. As to the SATLIN function of (6), we have replaced it with its quantized version defined as follows:

$$\mathsf{QSATLIN}(x) = \begin{cases} q_i q_h, & \text{if } x \geq q_i q_h \\ x, & \text{if } -q_i q_h < x < q_i q_h \\ -q_i q_h, & \text{if } x \leq -q_i q_h \end{cases} \tag{11}$$

where saturation occurs when the magnitude of the input is equal to $q_i q_h$ that corresponds to a unitary magnitude of the non-quantized inputs. Due to saturation, the output of the $\mathsf{QSATLIN}$ function requires fewer bits than its input to be represented, namely $\ell^q = 1 + \lceil \log_2(q_i q_h) \rceil$ at most (the first bit represents the sign), so each component in $\mathbf{y}_{h_q}$ requires at most ℓ^q bits. This should be contrasted with the number of bits needed to represent the input of the $\mathsf{QSATLIN}$ function. Such an input, in fact, is the result of the product between the inputs and the weights (it is a scalar product), which needs $1+3+(\ell^i-1)+(\ell^h-1) = \ell^i + \ell^h + 2$ bits, where the three additional bits are needed because we are adding five values (four for the scalar product between inputs and weights and one for the bias) so, this operation requires at most $\lceil \log_2(5) \rceil = 3$ additional bits. As already stated, after the application of $\mathsf{QSATLIN}$, the number of bits needed to represent the output is reduced to ℓ^q.

A similar analysis can be applied to the output layer, where we have

$$\mathbf{y}_{o_q} = \lfloor q_o q_h q_i \mathbf{b}_o \rceil + \lfloor q_o \mathbf{W}_o \, \mathbf{y}_{h_q} \rceil \tag{12}$$

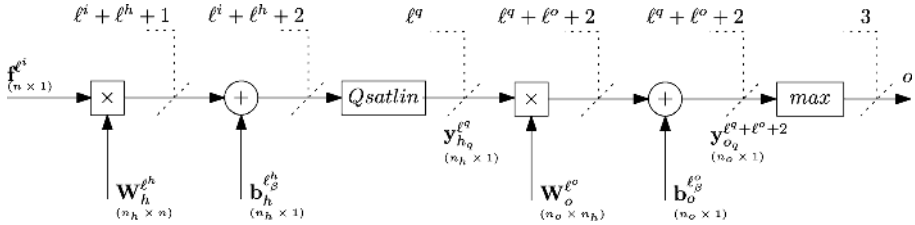

Fig. 10. Quantized NN. In our case, we have $n = 4$, $n_h = 6$, and $n_o = 6$.

and where each component of $\mathbf{y}_{o_q}$ requires $1 + 3 + (\ell^q - 1) + (\ell^o - 1) = \ell^q + \ell^o + 2$ bits (as before, one bit is used to represent the sign). At this point, the output of the maximum function that completes the classification is

$$o = \arg\max\{\mathbf{y}_{o_q}\} \tag{13}$$

and the number of bits necessary to represent it is the logarithm of the length of $\mathbf{y}_{o_q}$, since o is just the index of the biggest component. We are now ready to give a rigorous description of the number of bits necessary at each step of the NN. Specifically, the following definitions hold:

- $\mathbf{f}^{\ell^i}$, the inputs of the NN, are signed integers represented with ℓ^i bits;
- $\mathbf{W}_h^{\ell^h}$, the weights in the hidden layer, are signed integers represented with ℓ^h bits;
- $\mathbf{b}_h^{\ell_\beta^h}$, the biases in the hidden layer, are signed integers represented with $\ell_\beta^h = \ell^i + \ell^h - 1$ bits;
- $\mathbf{W}_o^{\ell^o}$, the weights in the output layer, are signed integers represented with ℓ^o bits;
- $\mathbf{b}_o^{\ell_\beta^o}$, the biases in the output layer, are signed integers represented with $\ell_\beta^o = \ell^q + \ell^o - 1$ bits.

We can summarize the operation of a quantized NN with the required bit-lengths at each step, with the following formula (see also Fig. 10):

$$\mathbf{y}_{o_q} = \underbrace{\mathbf{b}_o^{\ell_\beta^o} + q_o \mathbf{W}_o^{\ell^o} \,\mathsf{QSATLIN}\left(\underbrace{\underbrace{\mathbf{b}_h^{\ell_\beta^h} + \underbrace{\mathbf{W}_h^{\ell^h}\,\mathbf{f}^{\ell^i}}_{\ell^i+\ell^h+1}}_{\ell^i+\ell^h+2}}_{\ell^q}\right)}_{\ell^q+\ell^o+2}. \tag{14}$$

Finally, the NN classification result is $o = \arg\max\{\mathbf{y}_o\}$, which can be represented with $\lceil \log_2 n_o \rceil = 3$ bits.

We conclude this section by highlighting the fact that due to the presence of the saturation function the magnitudes of the intermediate results of the NN are kept bounded, as opposed to what happens in general with cascade quantized algorithms where the bit size of the quantities involved in the computation grows linearly with the number of subsequent multiplications to be performed. This is a very important property of our proposed implementation of a quantized NN, that will permit us to reduce the complexity of the secure protocol for the NN classifier.

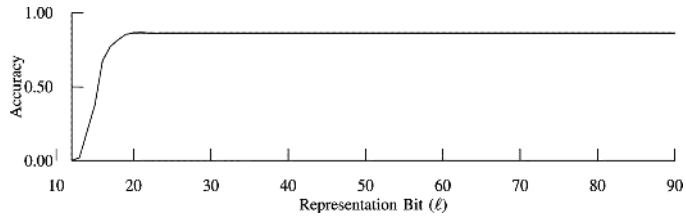

Fig. 11. Classification accuracy of LBP as a function of the input bit length.

TABLE I
LBP PARAMETERS

Description	Symbol	Value
# inputs	n	15
# nodes	z	12
# decision nodes	d	6
# bits to represent inputs and attribute vectors	ℓ	24
# bits for projections	ℓ'	51

C. Representation Versus Classification Accuracy

Having defined the quantized version of the LBP and NN classifiers, we can determine the minimum number of bits necessary to represent the values involved in the computations so to obtain the same accuracy of a floating point implementation of the classifiers. This is a crucial step, since as it will be shown later on, the size in bits of the input features and that of the classifier parameters have an immediate impact on the complexity of the SSP implementation of the classifiers.

1) Quantized Implementation of the LBP Classifier: Some LBP parameters depend on the protocol: the LBP has an input vector with $n = 15$ elements and is characterized by $z = 12$ nodes, whereof $d = 6$ are decision nodes. The bit length necessary to represent the elements of the input vector $\mathbf{f}^{\mathbf{c}}$ and the matrix $\mathbf{B}$ was obtained by running a simulation on the PhysioBank [26] database of ECG signals. We evaluated the classification accuracy by changing the bitlength, obtaining the results shown in Fig. 11. The accuracy of a floating point implementation (86.30%) is obtained for $\ell \geq 24$ bits. The results of the scalar products between the input vector and the projection vectors $\boldsymbol{\beta}_i$ can be represented with $\ell' = 2\ell + \lceil \log_2 n \rceil - 1 = 51$ bits. A summary of the parameters of the quantized LBP is given in Table I.

2) Quantized Implementation of the NN Classifier: As shown in Section II-B, the NN classifier has an input vector with $n = 4$ elements and is characterized by $n_h = 6$ hidden neurons and $n_o = 6$ output neurons. In order to determine the minimum number of bits necessary to reach the same classification accuracy of the LBP classifier, we run a simulator that evaluates the classification accuracy of the NN in the case $\ell^i = \ell^h = \ell^o$,

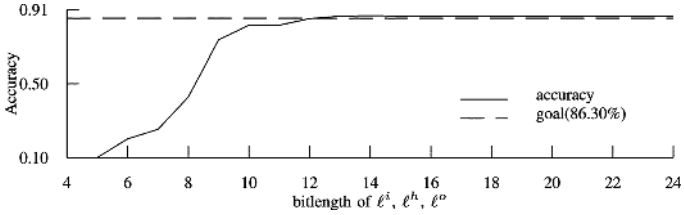

Fig. 12. Classification accuracy as a function of ℓ^i, ℓ^h, ℓ^o.

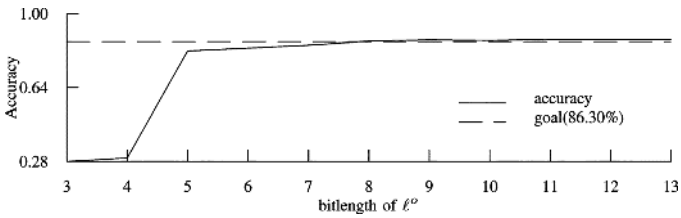

Fig. 13. Classification accuracy in function of ℓ^o, with $\ell^i = \ell^h = 13$.

TABLE II
NN PARAMETERS

Description	Symbol	Value
# neurons in input layer	n	4
# neurons in hidden layer	n_h	6
# neurons in output layer	n_o	6
# bits to represent inputs	ℓ^i	13
# bits to represent hidden layer parameters	ℓ^h	13
# bits to represent output layer parameters	ℓ^o	8
# bits to represent output of QSATLIN	ℓ^q	18
Maximum value of the output of QSATLIN	$q_i q_h$	131008

obtaining the results shown in Fig. 12. To guarantee the same classification obtained by the LBP classifier, we must let $\ell^i = \ell^h = \ell^o = 13$ bits. These numbers of bits (namely ℓ^i and ℓ^h) are those necessary to correctly represent the integer part of the input features and NN parameters multiplied by the quantization factors q_i and q_h [see (9)]. Specifically, according to the experiments we carried out with the PhysioBank database, they correspond to $q_i = 511.87$ and $q_h = 255.94$.

The result of the scalar products in the hidden layer are used as input in the $\mathsf{QSATLIN}(\cdot)$ function that gives an output with magnitude bounded by $q_i q_h$, hence it can be represented with $\ell_q = \lceil \log_2 q_i q_h \rceil + 1$ bits. From the values given above we obtain $q_i q_h = 131\,008$ yielding $\ell_q = 18$ bits, including the sign bit. As a further test we tried to reduce the number of bits used for the output layer once the bit length of the parameters of the hidden layer has been fixed. As shown in Fig. 13, letting $\ell_o = 8$ is sufficient to guarantee the same accuracy. A summary of the parameters we used for the NN classifier are given in Table II.

D. Security Requirements for SSP-Based ECG Classification

We conclude this section by analyzing the security requirements for an SSP implementation of the two classification algorithms described so far.

In our scenario $\mathcal{C}$ (a patient) owns an ECG beat signal and asks $\mathcal{S}$ (e.g., a remote health-care system) to determine which class his beat belongs to. $\mathcal{C}$ requires that $\mathcal{S}$ does not get any information about the ECG signal, not even the final result of the classification. At the same time, $\mathcal{S}$ does not want to reveal

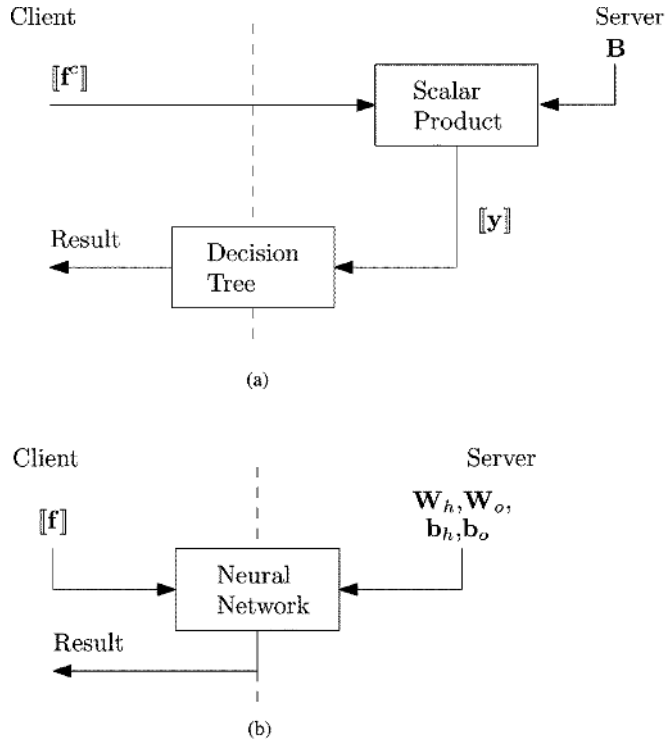

Fig. 14. High level protocols: (a) QDF plus LBP classifier; (b) NN classifier. Values in brackets are assumed to be encrypted with the public key of $\mathcal{C}$.

the parameters of the classification algorithm it is using, since they represent a valuable asset to protect.

From the point of view of security requirements, the QDF and NN classifiers are quite similar. In both cases (see Fig. 14), $\mathcal{C}$ does not want to reveal anything about the feature vector extracted from the ECG signal, namely the plain features $\mathbf{f}$ in the NN case, and the extended feature vector $\mathbf{f}^c$ in the QDF case. In the QDF case, $\mathcal{S}$ does not want to reveal any information about the projection matrix $\mathbf{B}$ and the actual form of the classification tree following the projection.

In the NN case, $\mathcal{S}$ wants to keep secret the NN parameters, namely the weights $\mathbf{W}_h$, $\mathbf{W}_o$ and also the biases $\mathbf{b}_h$ and $\mathbf{b}_o$. Note that we assume that $\mathcal{C}$ knows the general form of the classifier used by $\mathcal{S}$, i.e., a QDF classifier or an NN. $\mathcal{C}$ also knows the overall size of the classifier, that is the size of the projection matrix $\mathbf{B}$ and the size of the classification tree in the QDF case, and the number of layers and neurons in the NN case. This is a reasonable assumption since the domain specific knowledge needed to classify the ECGs and the knowledge got from the training, a knowledge that $\mathcal{S}$ may want to protect, reside in the classification tree and the matrix $\mathbf{B}$ for the QDF Classifier and in the weights and biases of the NN in the case of the NN Classifier.

IV. MODULAR DESIGN OF EFFICIENT SFE PROTOCOLS

In this section, we summarize the framework for modular design of efficient two-party secure function evaluation (SFE) protocols of [20]. We use this framework to intuitively describe our protocols for privacy-preserving classification of ECG data later in Section V as a sequence of operations on encrypted data.

1) Security Model: The standard approach for formalizing and proving security of cryptographic protocols is to consider

adversaries with different capabilities as summarized in the following. We refer to [20] and [28] for a detailed discussion.

Our protocols presented in this paper are secure against *semihonest* (or *passive*) adversaries that honestly follow the protocol but try to infer additional information from the transcript of messages seen. At first, it may appear contrived and trivial. Consideration of semihonest adversaries, however, is important in many typical practical settings. First, even externally unobservable cheating, such as poor random number generation, manipulations under encryption, etc., can be uncovered by an audit or reported by a conscientious insider, and cause negative publicity. Therefore, especially if the gain from cheating is low, it is often reasonable to assume that a well-established organization will exactly follow the protocol (and thus can be modeled as semihonest). Further, even if players are trusted to be *fully honest*, it is sometimes desired to ensure that the transcript of the interaction reveals no information. This is because in many cases, it is not clear how to reliably delete the transcript due to lack of control of the underlying computing infrastructure (network caching, virtual memory, etc.). Running an SFE protocol ensures that player's input cannot be subsequently revealed even by forensic analysis. At the same time, designing and evaluating the performance of protocols in the semihonest model is a first stepping stone towards protocols with stronger security guarantees. Indeed, most protocols and implementations of protocols for practical privacy-preserving applications focus on the semihonest model [17], [29]–[32].

The strongest security guarantees are against *malicious* (or *active*) adversaries that are allowed to arbitrarily deviate from the protocol, aiming to learn private inputs of the other parties and/or to influence the outcome of the computation. Not surprisingly, protection against such very powerful adversaries is relatively expensive as discussed in [20]. The basic transformation to convert a protocol secure against semihonest adversaries into one that provides security against malicious adversaries is to let each party prove in zero-knowledge that it behaves correctly [33]. Although these zero-knowledge proofs result only in a linear blowup of the protocol complexity, their efficient implementation is far from trivial and results in a substantially decreased performance (e.g., complexity of securely evaluating AES using garbled circuits is increased to approximately 800 times more communication and 160 times longer computation [19]).[3]

As the overhead for getting full-fledged security against both parties being malicious is too large for practical applications, we advocate the usage of *hybrid security* instead, where players are not equal in their capabilities, trustworthiness, and motivation. In particular, in our medical scenario, it is reasonable to assume that the service provider $\mathcal{S}$ has strong incentives not to cheat in the protocol (act semihonestly) as his cheating attempts might be detected and ruin his reputation and business model, whereas $\mathcal{C}$ may be much more willing to cheat (act maliciously). As described in [20], such protocols with asymmetric assumptions on the two players can be constructed efficiently, where the overhead is very moderate. In particular for garbled circuit-based protocols, only the underlying oblivious transfer protocol needs to be replaced with a slightly (about two times) less efficient one. Thus, protocols which are based only on garbled circuits (such as our protocol presented in Section V-B) are good candidates for settings with corresponding trust relationships.

[3]To the best of our knowledge, [19], which is an extension of [34], are the only works that provide an implementation and performance measurements for secure two-party computation in the malicious model.

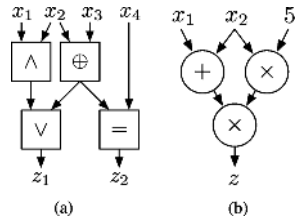

Fig. 15. Function representations. (a) Boolean circuit. (b) Arithmetic circuit.

Notation: By following the notation used until now, we call the two semihonest SFE participants *client* $\mathcal{C}$ and *server* $\mathcal{S}$. This naming choice is influenced by the asymmetry in the SFE protocols, which fits into the client–server model. We stress that while in most real-life two-party SFE scenarios this client–server relationship in fact exists, we do not limit ourself to this setting.

2) Security Parameters: We use the following security parameters: symmetric security parameter t and asymmetric security parameter T.

3) Function Representation: Given a function f that should be computed securely, the first task during the design of the corresponding SFE protocol is to find a suitable representation for f. Well-established representations which allow efficient SFE protocols are boolean circuits and arithmetic circuits as shown in Fig. 15. The representation determines the size of the function, e.g., multiplication is an arithmetic circuit with a single multiplication gate while its representation as boolean circuit is substantially larger. As described below, the online phase for SFE of boolean circuits is substantially more efficient than SFE of arithmetic circuits, so especially nonlinear functions such as comparisons benefit from boolean circuits [18]. The framework of [20] allows us to modularly compose functions from building blocks which are compactly represented as boolean or arithmetic circuits and then convert back and forth between the representations under encryption.

In the following, we summarize efficient methods for SFE of arithmetic and boolean circuits, and conversions between them. For a comprehensive description we refer to [20].

A. Homomorphic Encryption (HE) for Arithmetic Circuits

1) Additively Homomorphic Encryption: Encryption schemes such as [21] and [35] are semantically secure encryption schemes with plaintext space P and ciphertext space C that allow addition under encryption. They allow us to compute the operation $+$ on plaintexts by the corresponding operation $\cdot$ on ciphertexts, which satisfies $\forall\ x, y \in P : \mathrm{Dec}([\![x]\!] \cdot [\![y]\!]) = \mathrm{Dec}([\![x + y]\!])$. (We write $[\![x]\!]$ for homomorphic encryption of plaintext x.) This naturally allows for multiplication with a plaintext constant a using repeated doubling and adding: $\forall\ a \in \mathbb{N}$, $x \in P : \mathrm{Dec}([\![x]\!]^a) = \mathrm{Dec}(\underbrace{[\![x]\!] \cdot \cdots \cdot [\![x]\!]}_{a}) = \mathrm{Dec}([\![ax]\!])$.

SFE of arithmetic circuits can be naturally based on additively homomorphic encryption as follows: $\mathcal{C}$ generates a keypair for the homomorphic cryptosystem and sends the public key together with his inputs encrypted under the public key to $\mathcal{S}$. $\mathcal{S}$ uses the homomorphic property to evaluate the arithmetic circuit on the encrypted data. If the cryptosystem is only additively homomorphic, multiplication under encryption requires the help of $\mathcal{C}$ in a single round of interaction (details in [20]). Finally, $\mathcal{S}$ sends the encrypted outcome of the computation back to $\mathcal{C}$ who can decrypt.

2) Instantiation of HE: We use the additively homomorphic cryptosystem of Paillier [21] where the public key is an RSA modulus n, i.e., the product of two large primes p, q of bit length $T-1$ each, and the secret key is the factorization of n. The extension described in [35, Sec. 6] allows us to precompute expensive modular exponentiations of the form $r^n \bmod n^2$ in a setup phase, such that only two modular multiplications per encryption are needed in the online phase. As $\mathcal{C}$ knows the factorization of n, he can use the Chinese remainder theorem to efficiently precompute these exponentiations and for efficient decryption. The length of ciphertexts is $2T$ (recall, T is the asymmetric security parameter), as Paillier has ciphertext space $C = \mathbb{Z}^*_{n^2}$.

3) Packing: As the plaintext space (e.g., $P = \mathbb{Z}_n$ for the Paillier cryptosystem [21]) is often larger than the encrypted values, $\mathcal{S}$ can *pack* multiple values under encryption using Horner's method before sending them to $\mathcal{C}$ to reduce communication and number of decryptions by $\mathcal{C}$. For a comprehensive treatment of packing in a SFE framework, see [36].

4) Fully Homomorphic Encryption: As described in [20], the interactive approach for multiplication currently results in faster SFE protocols than using schemes which also provide one (e.g., [37]) or arbitrarily many (e.g., [38]–[40]) multiplications under encryption, called *fully homomorphic encryption*.

B. Garbled Circuits (GCs) for Boolean Circuits

GCs are an efficient method for SFE of boolean circuits. The general idea of GCs, going back to Yao [11], is to encrypt (*garble*) each wire with a symmetric encryption scheme. In contrast to homomorphic encryption (cf., Section IV-A), the encryptions/garblings here cannot be operated on directly, but require helper information which is generated and exchanged in the setup phase in form of a *garbled table* for each gate. On the other hand, the online phase of GCs is highly efficient as it requires only symmetric cryptographic operations (cf., Section IV-A1).

1) Yao's Protocol: On a high-level, Yao's GC protocol works as follows: in the setup phase, the *constructor* ($\mathcal{S}$) generates an encrypted version of the function f (represented as boolean circuit), called *garbled circuit* $\widetilde{f}$. For this, he assigns to each wire W_i of f two randomly chosen garbled values $\widetilde{w}_i^0$, $\widetilde{w}_i^1$ (symmetric keys) that correspond to the respective values 0 and 1. Note that $\widetilde{w}_i^j$ does not reveal any information about its plain value j as both keys look random. Then, for each gate of f, the constructor creates helper information in the form of a *garbled table* $\widetilde{T}_i$ that allows us to decrypt only the output key from the gate's input keys. The garbled circuit $\widetilde{f}$ consists of the garbled tables of all gates and is sent to the *evaluator* ($\mathcal{C}$). Later, in the online phase the *evaluator* ($\mathcal{C}$) obliviously obtains the garbled values $\widetilde{x}$ and $\widetilde{y}$ corresponding to the plain inputs x and y of $\mathcal{C}$ and $\mathcal{S}$, respectively, (see below). Afterwards, $\mathcal{C}$ evaluates the garbled circuit $\widetilde{f}$ on $\widetilde{x}, \widetilde{y}$ by evaluating the garbled gates one-by-one using their garbled tables. Finally, $\mathcal{C}$ obtains the corresponding garbled output values $\widetilde{y}$ which allow $\mathcal{S}$ to decrypt them into the corresponding plain output $z = f(x, y)$.

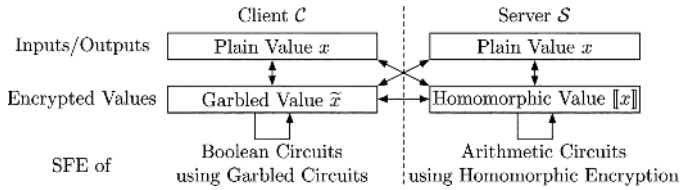

Fig. 16. Hybrid SFE protocols.

For converting a plain input bit y_i of $\mathcal{S}$ into its garbled equivalent, $\mathcal{S}$ simply sends the key $\widetilde{y}_i^{y_i}$ to $\mathcal{C}$. Similarly, $\mathcal{C}$ must obtain the garbled bit $\widetilde{x}_i$ corresponding to his input bit x_i, but without $\mathcal{S}$ learning x_i. This can be achieved by running (in parallel for each bit x_i of x) a 1-out-of-2 *oblivious transfer (OT)* protocol. OT is a cryptographic protocol into which $\mathcal{C}$ inputs his choice bit $b = x_i$ and $\mathcal{S}$ inputs two strings $s^0 = \widetilde{x}_i^0$ and $s^1 = \widetilde{x}_i^1$. The protocol guarantees that $\mathcal{C}$ obtains only the chosen string $s^b = \widetilde{x}_i^{x_i} = \widetilde{x}_i$ while $\mathcal{S}$ learns no information on $b = x_i$.

We emphasize that GCs cannot be evaluated twice, and refer to [41] for a proof of security for Yao's protocol in the semihonest model and to [20] for a summary of different GC methods and converting garbled outputs into plain values.

2) Implementation Details (cf., Appendix A): We give implementation details for efficient constructions of GC and OT in Appendix A1. In our protocols, we compose the functionality to be evaluated from several standard blocks (cf., Appendix A2 for details): comparison (CMP) and multiplication (MUL) of two unsigned ℓ-bit integer values, addition (ADD) and subtraction (SUB) of two signed ℓ-bit integer values in two's complement, multiplexing between two ℓ-bit values (MUX), maximum index of n unsigned ℓ-bit values (ARGMAX), and adding/subtracting an unsigned ℓ-bit value to/from a signed ℓ-bit value depending on a control input (ADDSUB).

C. Hybrid SFE of Mixed Representations

The SFE framework proposed in [20] allows us to modularly compose SFE protocols as a sequence[4] of operations on encrypted data as shown in Fig. 16: both parties have *plain values* as their inputs into the protocol. These plain values, denoted as x, are first encrypted by converting them into their corresponding encrypted value: a *garbled value*, denoted as $\widetilde{x}$, held by $\mathcal{C}$, or a *homomorphic value*, denoted as $[\![x]\!]$ held by $\mathcal{S}$, depending on which operations should be applied. After encryption, the function is securely evaluated on the encrypted values, which may involve conversion of the encryptions into the respective other type of encryption (see below). Finally, the encrypted output values are revealed and can be decrypted by converting them into the corresponding plain output values. In the following, we describe how to efficiently convert between the two types of encryptions.

[4]As all building blocks are proven secure against semihonest adversaries, their sequential composition inherits this security property (see, e.g., [42]).

1) Conversion Between Garbled and Homomorphic Values: To convert a homomorphic value $[\![x]\!]$ into a garbled value $\widetilde{x}$, $\mathcal{S}$ adds a random mask r under homomorphic encryption, sends the blinded value $[\![\bar{x}]\!] = [\![x]\!] \cdot [\![r]\!]$ to $\mathcal{C}$ who decrypts and both parties evaluate a garbled subtraction circuit which takes off the random mask under "garbled encryption." Parallel conversion of multiple values can be optimized by *packing* multiple values into one ciphertext before blinding. A similar method can be used for converting a garbled value $\widetilde{x}$ into an homomorphic value $[\![x]\!]$. For details we refer to [20].

V. Privacy-Preserving ECG Classification

We now apply the framework described in Section IV to the LBP and NN approaches to ECG classification. Remember that in our scenario the patient (playing the role of $\mathcal{C}$) has as input a quantized ECG signal and the medical service provider ($\mathcal{S}$) has a private classifier. After the protocol run, $\mathcal{C}$ knows the classification of each single heart beat of ECG signal, while $\mathcal{S}$ learned nothing about $\mathcal{C}$'s input.

A. Privacy-Preserving LBP Classifier

The protocols described in [17] allow secure evaluation of private linear branching programs (LBPs) which can be used for privacy-preserving ECG classification [7]. In the following, we describe the hybrid protocol which is based on a combination of homomorphic encryption and garbled circuits and achieves better performance than the protocol based on GCs only in the ECG classification setting.

We also propose a new optimization in case $\mathcal{S}$ is willing to reveal additional information about the classifier to $\mathcal{C}$.

1) Protocol Description: As described in Section III-A, the input provided by $\mathcal{C}$ is a quantized attribute vector $\mathbf{f}^{c,\ell} = \left(f_1^{c,\ell}, \ldots, f_n^{c,\ell}\right)^T$ of signed ℓ-bit integer values, and the input provided by $\mathcal{S}$ is the LBP $\mathcal{L}$. On a high level, the protocol of [17] proceeds as follows: First, the attribute vector $\mathbf{f}^{c,\ell}$ is projected onto the rows of the projection matrix $\mathbf{B}$ by using homomorphic encryption. The resulting d homomorphically encrypted ℓ'-bit signed integer values are converted into their corresponding garbled values which are compared with the threshold values $t_i^{\ell'}$ of $\mathcal{L}$ using garbled comparison circuits (observe that in our specific case all thresholds are equal to 0), and classified with a garbled d-input gate. Finally, the garbled output is decrypted and revealed to $\mathcal{C}$.

In more detail, the protocol works as follows:

1) To convert the quantized attribute vector $\mathbf{f}^{\ell}$ into its corresponding homomorphic vector held by $\mathcal{S}$, $\mathcal{C}$ encrypts each component with the homomorphic encryption system and sends the ciphertexts $[\![f_1^{c,\ell}]\!], \ldots, [\![f_n^{c,\ell}]\!]$ to $\mathcal{S}$.
2) Then, $\mathcal{S}$ computes for $i = 1, \ldots, d$ the homomorphic values $[\![y_i^{\ell'}]\!] = [\![\left(\beta_{\mathbf{i}}^{\ell}\right)^T \mathbf{f}^{\mathbf{c},\ell}]\!] = [\![\sum_{j=1}^{n} B_{i,j}^{\ell} f_j^{c,\ell}]\!]$ using the properties of the homomorphic cryptosystem without interaction with $\mathcal{C}$.
3) The d homomorphic ℓ'-bit values $[\![y_i^{\ell'}]\!]$ are converted into their garbled equivalents $\widetilde{y}_i^{\ell'}$ using the conversion protocol described in Section IV-C.
4) The garbled values $\widetilde{y}_i^{\ell'}$ are compared with the threshold values $\widetilde{t}_i^{\ell'}$ using d garbled comparison circuits for ℓ'-bit values.
5) For small values of d, the resulting garbled bits can be used as input into a garbled d-input gate to perform the final classification and output the outcome of the classification to $\mathcal{C}$.

For a detailed description of the protocol and its proof of security we refer to [17].

TABLE III
Complexity of LBP-Based ECG Classification Protocol

Complexity		Phase	Setup	Online
Communication		Moves	4	4
		Data [bits]	$3,123t + 6t^2$	$32T + 612t$
Computation	Client $\mathcal{C}$	H	$306 + 2t$	613
		EC mult	$2t + 1$	
		Paillier enc		15
		Paillier dec		1
	Server $\mathcal{S}$	H	$3,124 + t$	
		EC mult	$2t$	
		Paillier exp		91

2) Efficiency Analysis: As calculated in Section III-C1, the parameters for ECG classification are $\ell = 24$, $\ell' = 51$, $n = 15$, $d = 6$, $z = 12$, the symmetric security parameter is t, and the asymmetric security parameter is T. This yields the following costs for the LBP-based ECG classification protocol.

Abstractly, the following computations need to be performed under encryption: in Step 1, $\mathcal{C}$ performs $n = 15$ Paillier encryptions and sends $n = 15$ Paillier ciphertexts [$15 \cdot 2T = 30T$ bits] to $\mathcal{S}$. In Step 2, $\mathcal{S}$ performs $dn = 90$ modular exponentiations of Paillier ciphertexts and $d(n-1) = 84$ modular multiplications to combine the results. In Step 3, $\mathcal{S}$ packs the homomorphic values into one Paillier ciphertext [1 exponentiation, $2T$ bits] to $\mathcal{C}$ who decrypts once. Afterwards, both parties run the online phase of $d\ell' = 306 > t$ parallel OTs [$306 \cdot 2t = 612t$ bits] and evaluate a garbled subtraction circuit [$d\ell' = 306$ garbled non-XOR gates] to take off the random mask. In Step 4, $\mathcal{C}$ evaluates d garbled comparison circuits for ℓ'-bit integers [$d\ell' = 306$ garbled non-XOR gates]. Finally, in Step 5, $\mathcal{C}$ evaluates a garbled d-input gate.

The resulting complexity of the LBP-based ECG classification protocol is summarized in Table III.

a) Online Phase: The communication complexity in the online phase is approximately $2T + 30T + 612t = 32T + 612t$ bits. The computations performed by $\mathcal{C}$ consist of 15 Paillier encryptions, 1 decryption, and $306 + 306 + 1 = 613$ invocations of H for evaluating the garbled gates. The computations of $\mathcal{S}$ are dominated by $90 + 1 = 91$ modular exponentiations on Paillier ciphertexts. The online phase requires four moves.

b) Setup Phase: In the setup phase, the garbled circuits are generated by $\mathcal{S}$ [$2 \cdot 306 \cdot 4 + 2^d = 2512$ invocations of H] and transferred to $\mathcal{C}$ [$(2 \cdot 306 \cdot 3 + (2^d - 1)) \cdot (t+1) \approx 1899t$ bits]. Additionally, $306 > t$ parallel OTs need to be precomputed as described in Section A1.

3) Optimization: If $\mathcal{S}$ does not want to hide from $\mathcal{C}$ that indeed all thresholds $t_i^{\ell'}$ are zero, we can omit the comparison and directly use the most significant bit instead. By doing this, we no longer need the comparison circuits and save 306 non-XOR gates. This yields a more efficient protocol where in the setup phase $4 \cdot 306 = 1224$ less invocations of H by $\mathcal{S}$ and $3 \cdot 306t = 918t$ less communication are needed and in the online phase $\mathcal{C}$ performs 306 less invocations of H.

B. Privacy-Preserving NN Classifier

In the following, we describe a novel privacy preserving protocol for the NN-based classifier described in Section III-B.

With respect to previous works in NN computation in an SFE framework (e.g., [43], where the activation functions are implemented by introducing a rather security-critical multiplicative blinding step), our solution is provably secure and computationally efficient. Specifically, the general structure of our protocol follows the approach of [14] where the quantized NN for classification (in our case the NN shown in Fig. 10) is represented as a boolean circuit which is evaluated securely with a garbled circuit protocol. While [14] gives only concrete circuit instantiations for threshold functions, our NN uses the QSATLIN function (as defined in Section III-B) for which we give an efficient circuit instantiation.

1) Boolean Circuit: To simplify the presentation, we directly substitute variable names with the parameters determined in Section III-C2 and the sizes of circuit building blocks of Table VI.

a) Inputs: At the input of the circuit, we have the vector $\mathbf{f}^{\ell^i}$ ($n \cdot \ell^i = 4 \cdot 13 = 52$ wires) provided by $\mathcal{C}$. The inputs provided by $\mathcal{S}$ are

- $\mathbf{W}_h^{\ell^h}$ [$n_h \cdot n \cdot \ell^h = 6 \cdot 4 \cdot 13 = 312$ wires];
- $\mathbf{b}_h^{\ell_\beta^h}$ [$n_h \cdot (\ell^i + \ell^h - 1) = 6 \cdot 25 = 150$ wires];
- $\mathbf{W}_o^{\ell^o}$ [$n_o \cdot n_h \cdot \ell^o = 6 \cdot 6 \cdot 8 = 288$ wires];
- $\mathbf{b}_o^{\ell_\beta^o}$ [$n_h \cdot (\ell^q + \ell^o - 1) = 6 \cdot 25 = 150$ wires].

b) Gates: The circuit is constructed by instantiating the blocks of Fig. 10 with circuit building blocks as follows.

The inputs $\mathbf{f}^{\ell^i}$ and $\mathbf{W}_h^{\ell^h}$ are given in sign-magnitude representation s.t. we can easily compute their component-wise product: the magnitude is the product of the input magnitudes using $n \cdot n_h$ MUL blocks for 12-bit unsigned integers ($4 \cdot 6 \cdot (2 \cdot 12^2 - 12) = 6624$ non-XOR gates); the sign is computed "for free" by XORing the input signs. Now, depending on the sign, the magnitudes of the products are added to or subtracted from $\mathbf{b}_h^{\ell_\beta^h}$ by using ADDSUB blocks (at most $n_h \cdot (n \cdot (\ell^i + \ell^h + 1)) = 6 \cdot 4 \cdot 27 = 648$ non-XOR gates). The output $\mathbf{b}_h^{\ell_\beta^h} + \mathbf{W}_h^{\ell^h} \mathbf{f}^{\ell^i}$ is a vector with $n_h = 6$ components of $\ell^i + \ell^h + 2 = 28$-bit signed values in 2's complement representation.

Afterwards, for each of the $n_h = 6$ neurons the QSATLIN activation function is evaluated: first, the 28-bit input (let us call it x) is converted from 2's complement into sign/magnitude representation with an ADDSUB block (27 non-XOR gates). Afterwards, the QSATLIN function is computed according to (11) as $\mathsf{QSATLIN}(x) = \mathsf{sign}(x) \cdot \mathsf{min}(\mathsf{abs}(x), q_i q_h)$. The minimum is computed by comparing the magnitude of x with 1 using a CMP block (27 non-XOR gates). Depending on the outcome of this comparison, the 17-bit magnitude of the outcome is either the magnitude of x or $q_i q_h$ selected with a MUX block [17 non-XOR gates]. Overall, the conversion and computation of the $n_h = 6$ QSATLIN functions requires $6 \cdot (2 \cdot 27 + 17) = 426$ non-XOR gates. The output $\mathbf{y}_{h_q}$ is a vector of $n_h = 6$ components of $\ell^q = 18$-bit signed values in sign/magnitude representation.

The value $\mathbf{y}_{o_q} = \mathbf{b}_o^{\ell_\beta^o} + \mathbf{W}_o^{\ell^o} \mathbf{y}_{h_q}$ is computed similarly to the computation of $\mathbf{b}_h^{\ell_\beta^h} + \mathbf{W}_h^{\ell^h} \mathbf{f}^{\ell^i}$ described before and requires a circuit of at most $n_o n_h (2(\ell^q - 1)(\ell^o - 1) - (\ell^q - 1)) + n_o n_h (\ell^q + \ell^o + 2) = 6 \cdot 6 \cdot (2 \cdot 17 \cdot 7 - 17) + 6 \cdot 6 \cdot 28 = 7956 + 1008$ non-XOR gates. The output $\mathbf{y}_{o_q}$ is a vector with $n_o = 6$ components of $\ell^q + \ell^o + 2 = 28$-bit signed values in 2's complement representation.

Finally, the index of the maximum value is determined with an ARGMAX block ($28 \cdot (2n_o - 3) + (n_o + 1) = 28 \cdot (2 \cdot 6 - 3) + (6 + 1) = 259$ non-XOR gates).

c) Outputs: The output of the circuit for $\mathcal{C}$ is $\mathbf{o}$ (three wires).

d) Summary: In total, the circuit has 52 input wires of $\mathcal{C}$, $312 + 150 + 288 + 150 = 900$ input wires of $\mathcal{S}$, at most $6624 + 648 + 426 + 7956 + 1008 + 259 < 17\,000$ non-XOR two-input gates and three output wires for $\mathcal{C}$.

2) Protocol Description: As the NN classifier can be represented as a reasonably small boolean circuit C (cf., Section V-B1), it can be evaluated securely with Yao's garbled circuit (GC) protocol as described in Section IV-B. The inputs to the protocol are the quantized inputs of $\mathcal{C}$: $\mathrm{in}_\mathcal{C} = (\mathbf{f}^{\ell^i})$ and $\mathcal{S}$: $\mathrm{in}_\mathcal{S} = \left(\mathbf{W}_h^{\ell^h}, \mathbf{b}_h^{\ell_\beta^h}, \mathbf{W}_o^{\ell^o}, \mathbf{b}_o^{\ell_\beta^o}\right)$. These plain inputs are converted into their corresponding garbled inputs $\widetilde{\mathrm{in}}_\mathcal{C}$, $\widetilde{\mathrm{in}}_\mathcal{S}$ provided to $\mathcal{C}$ who uses them to evaluate a garbled circuit $\widetilde{C}$ created by $\mathcal{S}$ to obtain the garbled output $\widetilde{z} = \widetilde{C}(\widetilde{\mathrm{in}}_\mathcal{C}, \widetilde{\mathrm{in}}_\mathcal{S})$. Finally, the garbled output is converted into the plain value $\mathbf{o} = z$ output to C.

3) Efficiency Analysis: The costs for this NN-based ECG classification protocol with symmetric parameter t are summarized in Table IV:

a) Setup Phase: In the setup phase, the garbled circuit $\widetilde{C}$ is generated by $\mathcal{S}$ ($17\,000 \cdot 4 = 68\,000$ invocations of H) and transferred to $\mathcal{C}$ ($17\,000 \cdot (t + 1) \approx 17\,000t$ bits). Additionally, $|\mathrm{in}_\mathcal{C}| = 52 \leq t$ parallel OTs need to be precomputed as described in Appendix A1 to convert $\mathcal{C}$'s input $\mathrm{in}_\mathcal{C}$ into its garbled version $\widetilde{\mathrm{in}}_\mathcal{C}$. The setup phase requires three moves.

b) Online Phase: In the online phase, $\mathcal{C}$ obtains the garbled inputs $\widetilde{\mathrm{in}}_\mathcal{S}$ corresponding to $\mathcal{S}$'s input $\mathrm{in}_\mathcal{S}$ ($900 \cdot (t + 1) \approx 900t$ bits) and executes the online phase of 52 parallel OTs ($\approx 104t$ bits) which requires two moves. Evaluation of $\widetilde{C}$ requires 17 000 invocations of H.

TABLE IV
COMPLEXITY OF NN-BASED ECG CLASSIFICATION PROTOCOL

Complexity		Phase	Setup	Online
Communication		Moves	3	2
		Data [bits]	$17,312t$	$1,004t$
Computation	Client $\mathcal{C}$	H	52	17,000
		EC mult	104	
	Server $\mathcal{S}$	H	68,104	
		EC mult	105	

C. Comparison

Finally, we compare the two approaches we have investigated for ECG classification, from an efficiency point of view.

1) Communication Complexity: For the length of the security parameters we consider short-term security ($t = 80$ and $T = 1024$ bits) and long-term security ($t = 128$ and $T =$

3072 bits). In both cases, the NN-based protocol requires approximately 5 times more communication in the setup phase (e.g., 169 kByte compared to 33.6 kByte for short-term security). For short-term security, both protocols have similar online communication complexity of 10 kByte, while for long-term security the NN-based protocol has 38% less communication (15.7 kByte compared to 21.6 kByte).

2) Computation Complexity: The LBP-based protocol has a larger computation complexity than the NN-based protocol as it requires more elliptic curve multiplications and operations on Paillier ciphertexts which are substantially more expensive than evaluations of the cryptographic hash function H.

3) Timing Complexity: To compare the timings of both protocols, we refer to two prototype implementations of similar protocols which both measure the runtime on two Intel Core Duo's running at 3.0 GHz, with 4 GB of RAM connected by a 1-GB ethernet. We leave a more detailed analysis of the run-times of both protocols with a prototype implementation of both protocols as future work.

The prototype implementation of the LBP-based protocol for ECG classification reported in [7] requires a total computation time of 18.7 s for short-term security parameters. To the best of our knowledge, this is the only implementation of LBPs.[5]

In comparison, the prototype implementation of [19] measures SFE of a functionality of similar size to the NN-classifier (an AES circuit consisting of 11 286 non-XOR gates) and requires in total only 7 s for long-term security parameters. As shown in [45], most of the complexity of GC-based SFE protocols can be precomputed in a setup/precomputation phase resulting in an online phase which is approximately ten times faster in the semihonest model resulting in an estimated setup time of 6 s and an online time of 1 s. One may wonder whether a classification time in the order of a few seconds is affordable in real life applications. The ultimate answer to this question depends on the application at hand; however, we can observe that a running time of less than 1 s would be enough for applications wherein heart beats are classified at the same pace at which they are produced. While our protocols are not that fast, their performances are not far away from the above so-to-say real-time requirements thus witnessing the validity of our solutions.

For security against stronger covert adversaries (who can be caught in a cheating attempt with a fixed probability) or malicious adversaries (who are caught cheating with overwhelming probability), the runtimes measured in [19] are 1 min and 18.6 min, respectively. Furthermore, the computation and communication complexity of GC-based protocols can be substantially improved by using hardware accelerators for GC evaluation [46], trusted hardware tokens for local GC creation [47], possibly in a cloud computing scenario where GCs are evaluated in parallel in the cloud [48].

4) Summary: In summary, we can state that the LBP classifier is preferable from a communication complexity point of view when considering the total amount of data sent in setup plus online phase. The reason is that the 12-bit values to be multiplied are too small to benefit from homomorphic encryption and hence must be implemented most efficiently with the GC approach. In this way, multiplications must be implemented at the logic circuit level with a number of gates (that ultimately determines the communication complexity of the setup phase of the protocol) depending quadratically[6] on the bit size of the factors. For a detailed comparison of protocols for secure multiplication, based on homomorphic encryption or garbled circuits, we refer to [45].

On the other side, the NN protocol relies only on fast symmetric encryption operations, hence resulting in a better performance from a computational complexity perspective, an advantage that becomes more significant for long-term security, since the security parameters of asymmetric cryptosystems are going to increase more rapidly than those of symmetric cryptosystems.

By considering the classifier structures underlying the two protocols, we see that the NN ensures a twofold advantage since: 1) it allows us to work on a smaller feature vector (4 features instead of the 15 components of the composite feature vector required by the QDF-LBP classifier), and 2) it requires a smaller number of bits for the representation of the feature vector and the classifier parameters. This is partially due to the presence of hard limiting activation functions avoiding that the inner results of the computation grows in magnitude. It is thanks to the above properties that the GC implementation of the NN protocol does not pay a too large penalty for the necessity of working entirely with boolean instead of arithmetic circuits.

We conclude our discussion by observing that the complexity of both protocols depend on the number of features used to classify the ECG signals. In the NN case, the dependence of the size of the classifier on the number of features is not easy to determine. On one side it results in an increase of the size of the input layer of the NN, with a linear impact on the complexity of the part of the protocol corresponding to the computation of the input of the hidden layer. On the other side it is likely that the number of neurons in the hidden layer will have to increase as well thus resulting in a superlinear dependence of the complexity on the number of features. The overall complexity increase, however, is likely to be less than quadratic, given that the size of the output layer will remain constant. In the LBP case, the dependence is at least quadratic due to the inclusion within the composite feature vector of quadratic terms[7]. For this reason, we expect that the NN structure is going to become even more advantageous if the number of features considered by the classifier increases.

VI. Conclusion

The need for privacy protection is steadily increasing in our society due to the wide diffusion of online distributed services offered by nontrusted parties having potential access to private information like users' preferences or other personal data. This need is even more pressing in settings where the information to be protected is related to the health of the users: with the appearance of more and more online medical repositories, it is simple to imagine that in a few years the approach to health care

[5]SFE of the less generic (and hence not directly applicable to our scenario) branching programs functionality takes 5 s and 361 kB for 15 nodes and 100 attributes [44].

[6]We note that for multiplications of larger values with $\ell \geq 20$ bits, the overhead is less than quadratic as shown in [45].

[7]Of course the size of the decision tree may change as well

will be completely different from the actual one and it is of the outmost importance that manipulation of sensible data does not compromise the privacy of users.

In this paper, we have shown how SSP technology may help to achieve the twofold goal of allowing the processing of biomedical signals while ensuring the privacy of the signal's owner. The proposed protocols address the classification of ECG signals and rely on some innovative SFE constructions and on a proper design of the classification algorithms so to ease their implementation in an SSP framework.

More specifically, we have described two alternative ECG classification protocols: the former is based on a QDF classifier and is implemented by relying on a hybrid approach wherein homomorphic encryption and garbled circuit theories are used together; the latter implements an NN classifier and relies only on garbled circuit constructions. This approach was possible due to the capacity of the NN to limit the size in bits of the input, output, and inner values of the computation.

While both protocols are rather efficient, thus opening promising directions for real-world applications, the QDF classifier is (slightly) preferable from the point of view of communication complexity, while the NN classifier is (slightly) preferable from a computational complexity perspective.

As we said, the promising results of our research pave the way to a number of interesting research directions that can be exploited in the future. First of all, new more efficient implementations of the underlying building blocks can be sought for to further improve the efficiency of the classifiers. Second, the results we obtained for the particular case of ECG classification should be extended to more general setups with the goal of deriving some general conclusions about the suitability of the QDF and the NN approaches to classification in an SSP framework. Security models more stringent than the semihonest model should also be considered, attempting to develop efficient constructions in the presence of malicious adversaries. Last but not least, some new advances in cryptography, like the recently proposed fully homomorphic cryptosystems [38], [39], [49] could open new research directions finally leading to a brand new class of efficient SSP protocols.

APPENDIX

A. Implementation Details

In this section, we summarize most efficient methods and primitives for implementing garbled circuits (Appendix A1) and the evaluated circuit building blocks (Appendix A2).

1) Efficient Implementation of Garbled Circuits: For implementing GCs and OT with maximum efficiency, we use a random oracle H which can be instantiated with a suitably chosen cryptographic hash function such as SHA-256 [50].

Instantiation of GC: We use the currently most efficient GC method of [19] which is provably secure in the random oracle model and provides "free XOR" gates, i.e., garbled XOR gates require no garbled table and negligible computation only. Evaluation of a garbled d-input non-XOR gate requires one invocation of H; to create the garbled table with $2^d - 1$ entries 2^d invocations of H are needed; each table entry and each garbled value has size $t + 1$ bits (recall, t is the symmetric security parameter).

TABLE V
COMPLEXITY OF n-PARALLEL OT PROTOCOL OF t-BIT STRINGS

Complexity		Phase	Setup	Online
For $n \leq t$: [53]				
Communication		Moves	3	2
		Data [bits]	$6nt$	$2nt$
Computation	Client $\mathcal{C}$	H	n	
		EC mult	$2n$	
	Server $\mathcal{S}$	H	$2n$	
		EC mult	$2n+1$	
For $n > t$: [52] + [53]				
Communication		Moves	4	2
		Data [bits]	$4nt + 6t^2$	$2nt$
Computation	Client $\mathcal{C}$	H	$n + 2t$	
		EC mult	$2t + 1$	
	Server $\mathcal{S}$	H	$2n + t$	
		EC mult	$2t$	

Oblivious Transfer (OT): To efficiently implement OT, [20] proposes to combine the following techniques resulting in the total complexity summarized in Table V.

Precomputing OT [51]: This construction allows us to precompute OTs in a setup phase, where both parties run the OT protocol on random inputs. Later, in the more time-critical online phase, they use these random inputs to mask their real inputs with a one-time pad. For n parallel OTs of $(t+1)$-bit strings, the online phase requires two moves and $n(1+2(t+1)) \approx 2nt$ bits datatransfer.

Extending OT Efficiently [52]: This technique allows us to reduce the computation complexity of the setup phase by replacing n parallel OTs of $(t+1)$-bit-strings with t parallel OTs of n-bit strings performed in the opposite direction. This construction requires $2n$ invocations of H by $\mathcal{S}$ and n invocations by $\mathcal{C}$. The protocol needs one message of size $2n(t+1) \approx 2nt$ bits in addition to those of the underlying OT protocol.

Efficient OT Protocols [53]: n parallel OTs of ℓ-bit strings can be implemented efficiently with the protocol of [53] over elliptic curves. This protocol consists of three messages in which $2n + 1$ elliptic curve points and $2n\ell$ encrypted bits are sent. Using point compression, each point can be represented with $2t + 1$ bits and hence the overall communication complexity of this protocol is $(2n + 1) \cdot (2t + 1) + 2n\ell \approx 4nt + 2n\ell$ bits. As computation, the sender has to perform $2n + 1$ point multiplications and $2n$ invocations of H, and the receiver performs $2n$ point multiplications and n invocations of H. This protocol is provably secure in the random oracle model.

2) Efficient Circuit Constructions: In our constructions, we use several standard circuit building blocks which are optimized for a small number of XOR gates. The size and references to the details of such optimized constructions are listed in Table VI. In particular, we need blocks for comparison (CMP) and multiplication (MUL) of two unsigned ℓ-bit integer values, as well as for addition (ADD) and subtraction (SUB) of two signed ℓ-bit integer values in two's complement. A multiplexer (MUX) can select one of two ℓ-bit inputs as output.

TABLE VI
EFFICIENT CIRCUIT CONSTRUCTIONS WITH FREE XOR

Functionality	#non-XOR 2-input gates	Reference
ADD	ℓ	[54]
SUB, CMP	ℓ	[18]
MUX	ℓ	[55]
MUL	$2\ell^2 - \ell$	[18]
ADDSUB	ℓ	derived from [54]
ARGMAX	$\ell(2n-3)+(n+1)$	derived from [18]

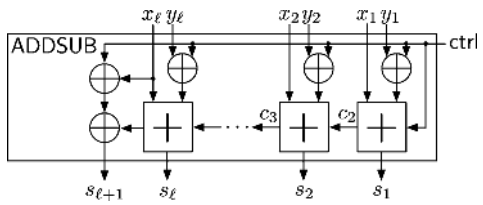

Fig. 17. Controlled addition or subtraction circuit (ADDSUB).

Maximum Index: To determine the maximum index of n unsigned ℓ-bit values, called ARGMAX block, we adopt the circuit for computing the maximum value and index of [18] and omit the last multiplexer for the maximum value resulting in a circuit of size $2\ell(n-1)+(n+1)-\ell = \ell(2n-3)+(n+1)$ non-XOR gates.

Controlled Addition or Subtraction: A block ADDSUB which can add/subtract the unsigned ℓ-bit value y to/from the signed ℓ-bit value x in 2's complement depending on a control bit ctrl can be naturally constructed from ADD: If $\mathsf{ctrl} = 1$, y must be subtracted from x. This can be done by converting y into two's complement and adding it to x. The resulting circuit is shown in Fig. 17. As each of the bit-addition blocks $(+)$ can be implemented with one non-XOR gate [18], this circuit has size ℓ non-XOR gates.

To convert an $(\ell+1)$-bit signed integer value x from 2's complement to sign/magnitude representation, the least significant ℓ bits of x are added to or subtracted from 0 depending on the sign of x, i.e., the most significant bit of x, using an ADDSUB block of size ℓ non-XOR gates.

REFERENCES

[1] Z. Erkin, A. Piva, S. Katzenbeisser, R. Lagendijk, J. Shokrollahi, G. Neven, and M. Barni, "Protection and retrieval of encrypted multimedia content: When cryptography meets signal processing," *EURASIP J. Inform. Sec.*, vol. 2007, p. 17, 2007.

[2] R. Agrawal and R. Srikant, "Privacy-preserving data mining," *SIGMOD Record*, vol. 29, no. 2, pp. 439–450, 2000.

[3] Y. Lindell and B. Pinkas, "Privacy preserving data mining," *J. Cryptology*, vol. 15, no. 3, pp. 177–206, 2008.

[4] J. Bringer and H. Chabanne, "An authentication protocol with encrypted biometric data," in *Proc. Progress Cryptology (AFRICACRYPT'08)*, 2008, pp. 109–124.

[5] "Homomorphic Encryption For Secure Watermarking," WO Patent WO/2006/129,293, Dec. 2006.

[6] E. Aimeur, G. Brassard, J. Fernandez, F. Onana, and Z. Rakowski, "Experimental demonstration of a hybrid privacy-preserving recommender system," in *Proc. Int. Conf. Availability, Rel. Security.*, 2008, pp. 161–170, IEEE Computer Society.

[7] M. Barni, P. Failla, V. Kolesnikov, R. Lazzeretti, A.-R. Sadeghi, and T. Schneider, "Efficient privacy-preserving classification of ECG signals," in *Proc. IEEE Int. Workshop Inf. Forensics Security (WIFS09)*, 2009, pp. 91–95.

[8] M. McBride, "Google health: Birth of a giant," *Health Manage. Technol.*, vol. 29, pp. 8–10, 2008.

[9] D. Blankenhorn, Microsoft HealthVault Is Nothing Like Google Health 2008 [Online]. Available: http://www.zdnet.com/blog/healthcare/microsoft-healthvault-is-nothing-like-google-health/742

[10] U. R. Acharya, J. Suri, J. A. E. Spaan, and S. M. Krishnan, *Advances in Cardiac Signal Processing*. New York: Springer, 2007.

[11] A. C. Yao, "How to generate and exchange secrets," in *Proc. IEEE Symp. Foundations Comput. Sci. (FOCS'86)*, 1986, pp. 162–167.

[12] D. Malkhi, N. Nisan, B. Pinkas, and Y. Sella, "Fairplay—A secure two-party computation system," in *Proc. USENIX Security Symp. (Security'04)*, San Diego, CA, 2004, pp. 287–302 [Online]. Available: http://www.cs.huji.ac.il/project/Fairplay

[13] V. Kolesnikov and T. Schneider, "A practical universal circuit construction and secure evaluation of private functions," in *Proc. Financial Cryptography Data Security (FC'08)*, 2008, vol. 5143, pp. 83–97, Springer, LNCS.

[14] A.-R. Sadeghi and T. Schneider, "Generalized universal circuits for secure evaluation of private functions with application to data classification," in *Proc. Int. Conf. Inf. Security Cryptology (ICISC'08)*, 2008, vol. 5461, pp. 336–353, Springer, LNCS.

[15] D. Ge, N. Srinivasan, and S. M. Krishnan, "The application of autoregressive modeling in cardiac arrhythmia classification," in *Advances in Cardiac Signal Processing*. New York: Springer, 2007, pp. 209–226.

[16] D. F. Ge, N. Srinivasan, and S. M. Krishnan, "Cardiac arrhythmia classification using autoregressive modeling," *BioMed. Eng. OnLine*, vol. 1, no. 1, pp. 5–5, 2002.

[17] M. Barni, P. Failla, V. Kolesnikov, R. Lazzeretti, A.-R. Sadeghi, and T. Schneider, "Secure evaluation of private linear branching programs with medical applications," in *Proc. Eur. Symp. Res. Comput. Security (ESORICS'09)*, 2009, vol. 5789, pp. 424–439, Springer, LNCS.

[18] V. Kolesnikov, A.-R. Sadeghi, and T. Schneider, "Improved garbled circuit building blocks and applications to auctions and computing minima," in *Cryptology Network Security (CANS'09)*, 2009, vol. 5888, pp. 1–20, Springer, LNCS.

[19] B. Pinkas, T. Schneider, N. P. Smart, and S. C. Williams, "Secure two-party computation is practical," in *Proc. Adv. Cryptology (ASIACRYPT 2009)*, 2009, vol. 5912, pp. 250–267, Springer, LNCS.

[20] V. Kolesnikov, A.-R. Sadeghi, and T. Schneider, Modular design of efficient secure function evaluation protocols Cryptology ePrint Archive, Rep. 2010/079, 2010 [Online]. Available: http://eprint.iacr.org

[21] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proc. Adv. Cryptology (EUROCRYPT'99)*, 1999, vol. 1592, pp. 223–238, Springer, LNCS.

[22] J. Rajan and P. Rayner, "Generalized feature extraction for time-varying autoregressive models," *IEEE Trans. Signal Process.*, vol. 44, no. 10, pp. 2498–2507, Oct. 1996.

[23] G. E. P. Box, G. M. Jenkins, and G. C. Reinsel, *Time Series Analysis*. San Francisco, CA: Holden-day, 1976.

[24] M. Kattan and R. Beck, "Artificial neural networks for medical classification decisions," *Archives Pathology Lab. Medicine (1976)*, vol. 119, no. 8, pp. 672–677, 1995.

[25] G. Arulampalam and A. Bouzerdoum, "Application of shunting inhibitory artificial neural networks to medical diagnosis," in *Proc. ANZIIS 2001. Proc. 7th Australian New Zealand Intelligent Inf. Syst. Conf.*, Perth, WA, 2001, pp. 89–94.

[26] A. L. Goldberger, L. A. N. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley, "Physiobank, physiotoolkit, and physionet: Components of a new research resource for complex physiologic signals," *Circulation*, vol. 101, no. 23, pp. e215–e220, Jun. 2000.

[27] M. Hagan and M. Menhaj, "Training feedforward networks with the marquardt algorithm," *IEEE Trans. Neural Netw.*, vol. 5, no. 6, pp. 989–993, Nov. 1994.

[28] Y. Lindell and B. Pinkas, "Secure multiparty computation for privacy-preserving data mining," *J. Privacy Confidentiality*, vol. 1, no. 1, pp. 59–98, 2009.

[29] M. Naor, B. Pinkas, and R. Sumner, "Privacy preserving auctions and mechanism design," in *Proc. ACM Conf. Electron. Commerce*, 1999, pp. 129–139.

[30] Z. Erkin, M. Franz, J. Guajardo, S. Katzenbeisser, I. Lagendijk, and T. Toft, "Privacy-preserving face recognition," in *Proc. Privacy Enhancing Technol. (PET'09)*, 2009, vol. 5672, pp. 235–253, Springer, LNCS.

[31] A.-R. Sadeghi, T. Schneider, and I. Wehrenberg, "Efficient privacy-preserving face recognition," in *12th Int. Conf. Inf. Security Cryptology (ICISC'09)*, 2009, vol. 5984, pp. 229–244, Springer, LNCS.
[32] M. Osadchy, B. Pinkas, A. Jarrous, and B. Moskovich, "SCiFI—A system for secure face identification," in *IEEE Symp. Security & Privacy (S&P'10)*., 2010, pp. 239–254.
[33] O. Goldreich, Foundations of cryptography Cambridge Univ. Press, 2004, vol. 2 [Online]. Available: http://www.wisdom.weizmann.ac.il/oded/foc-vol2.html
[34] Y. Lindell, B. Pinkas, and N. Smart, "Implementing two-party computation efficiently with security against malicious adversaries," in *Proc. Security Cryptography For Networks (SCN'08)*, 2008, vol. 5229, pp. 2–20, Springer, LNCS.
[35] I. Damgård and M. Jurik, "A generalisation, a simplification and some applications of paillier's probabilistic public-key system," in *Proc. Public-Key Cryptography (PKC'01)*, 2001, pp. 119–136, Springer, LNCS.
[36] T. Bianchi, A. Piva, and M. Barni, "Composite signal representation for fast and storage-efficient processing of encrypted signals," *IEEE Trans. Inf. Forensics Security*, vol. 5, no. 1, pp. 180–187, Mar. 2010.
[37] D. Boneh, E.-J. Goh, and K. Nissim, "Evaluating 2-DNF formulas on ciphertexts," in *Proc. Theor. Cryptography (TCC'05)*, 2005, vol. 3378, pp. 325–341, Springer, LNCS.
[38] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proc. ACM Symp. Theor. Comput. (STOC'09).*, 2009, pp. 169–178.
[39] M. Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, "Fully homomorphic encryption over the integers," in *Proc. Adv. Cryptology (EUROCRYPT'10)*, 2010, vol. 6110, pp. 24–43, Springer, LNCS.
[40] N. P. Smart and F. Vercauteren, "Fully homomorphic encryption with relatively small key and ciphertext sizes," in *Proc. Public Key Cryptography (PKC'10)*, 2010, vol. 6056, pp. 420–443, Springer, LNCS.
[41] Y. Lindell and B. Pinkas, "A proof of Yao's protocol for secure two-party computation," *J. Cryptology*, vol. 22, no. 2, pp. 161–188, 2009 [Online]. Available: http://eprint.iacr.org/2004/175
[42] O. Goldreich, *Foundations of Cryptography: Volume 2, Basic Applications*. Cambridge, U.K.: Cambridge Univ. Press, 2004.
[43] C. Orlandi, A. Piva, and M. Barni, "Oblivious neural network computing from homomorphic encryption," *EURASIP J. Inform. Sec.*, vol. 2007, p. 11, 2007.
[44] J. Brickell, D. E. Porter, V. Shmatikov, and E. Witchel, "Privacy-preserving remote diagnostics," in *Proc. ACM Comput. Commun. Security (ACM CCS'07).*, 2007, pp. 498–507.
[45] W. Henecka, S. Kögl, A.-R. Sadeghi, T. Schneider, and I. Wehrenberg, "TASTY: Tool for automating secure two-party computations," in *Proc. ACM Comput. Commun. Security (CCS'10).*, 2010, pp. 451–462.
[46] K. Järvinen, V. Kolesnikov, A.-R. Sadeghi, and T. Schneider, "Garbled circuits for leakage-resilience: Hardware implementation and evaluation of one-time programs," in *Proc. Cryptographic Hardware Embedded Syst. (CHES'10)*, 2010, vol. 6225, pp. 383–397, Springer, LNCS.
[47] K. Järvinen, V. Kolesnikov, A.-R. Sadeghi, and T. Schneider, "Embedded SFE: Offloading server and network using hardware tokens," in *Proc. Financial Cryptography Data Security (FC'10)*, 2010, vol. 6052, pp. 207–221, Springer, LNCS.
[48] A.-R. Sadeghi, T. Schneider, and M. Winandy, "Token-based cloud computing—Secure outsourcing of data and arbitrary computations with lower latency," in *Proc. Trust Trustworthy Comput. (TRUST'10)—Workshop Trust Cloud*, 2010, vol. 6101, pp. 417–429, Springer, LNCS.
[49] C. Aguilar, P. Gaborit, and J. Herranz, "Additively homomorphic encryption with d-operand multiplications," in *Adv. Cryptology (CRYPTO'10)*, Santa Barbara, CA, 2010, pp. 138–154, Springer, LNCS.
[50] NIST, U.S. National Institute of Standards and Technology, Federal information processing standards (FIPS 180–2). Announcing the secure hash standard Aug. 2002 [Online]. Available: http://csrc.nist.gov/publications/fips/fips180-2/fips-180-2.pdf
[51] D. Beaver, "Precomputing oblivious transfer," in *Proc. Adv. Cryptology (CRYPTO'95)*, 1995, vol. 963, pp. 97–109, Springer, LNCS.
[52] Y. Ishai, J. Kilian, K. Nissim, and E. Petrank, "Extending oblivious transfers efficiently," in *Proc. Adv. Cryptology (CRYPTO'03)*, 2003, vol. 2729, pp. 145–161, Springer, LNCS.
[53] M. Naor and B. Pinkas, "Efficient oblivious transfer protocols," in *Proc. ACM-SIAM Symp. Discrete Algorithms (SODA'01)*, 2001, pp. 448–457, Society for Industrial and Applied Mathematics.
[54] J. Boyar, R. Peralta, and D. Pochuev, "On the multiplicative complexity of boolean functions over the basis $(\wedge, \oplus, 1)$," *Theor. Comput. Sci.*, vol. 235, no. 1, pp. 43–57, 2000.
[55] V. Kolesnikov and T. Schneider, "Improved garbled circuit: Free XOR gates and applications," in *Proc. Int. Colloquium Automata, Languages Programming (ICALP'08)*, 2008, vol. 5126, pp. 486–498, Springer, LNCS.

Mauro Barni (S'90–M'96–SM'06) graduated in electronic engineering at the University of Florence in 1991. He received the Ph.D. degree in informatics and telecommunications in October 1995.

He has carried out his research activity for over 20 years, first at the Department of Electronics and Telecommunication, University of Florence, then at the Department of Information Engineering, University of Siena, Siena, Italy, where he works as Associate Professor. During the last decade he has been studying the application of image processing techniques to copyright protection and authentication of multimedia (digital watermarking). He is author/coauthor of about 250 papers published in international journals and conference proceedings, and holds three patents in the field of digital watermarking. He is coauthor of the book *Watermarking Systems Engineering: Enabling Digital Assets Security and other Applications*. He participated to several National and European research projects on diverse topics, including computer vision, multimedia signal processing, remote sensing, digital watermarking, IPR protection.

Dr. Barni serves as Associate Editor of the IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY and the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY. He was the founding editor of the *EURASIP Journal on Information Security*. He was the general chairman of the 2004 edition of IEEE workshop on Multimedia Signal Processing (MMSP'04) and the 2005 edition of the International Workshop on Digital Watermarking (IWDW'05). He is the chairman of the IEEE Information Forensic and Security Technical Committee (IFS-TC) of the IEEE Signal Processing Society. He is a senior member of EURASIP.

Pierluigi Failla was born in Grosseto, Italy, in 1981. He received the Masters degree (*summa cum laude*) in engineering in computer science from the University of Siena, Siena, Italy, in 2007, with a thesis about computing discrete Fourier tranform on encrypted data, and Ph.D. degree in information engineering in the area of privacy preserving algorithms and protocols.

From 2007 to 2010, he has been with the Department of Information Engineering, University of Siena. His main research interests are in the field of data and signal protection with particular regards to privacy preserving algorithm for biomedical and biometric application. Presently he is working as a Systems Analyst in the Research and Advanced System Design Group in Elt—Elettronica SpA, where he works mainly on cyber warfare and electronic warfare.

Riccardo Lazzeretti received the master degree (*cum laude*) in computer science engineering at the University of Siena, Siena, Italy, in 2007 with a thesis on to the lossless compression of encrypted images. He was a Ph.D. student in information engineering at the Department of Information Engineering, University of Siena, where currently he has a research grant.

His research activity focuses on multiparty computation for privacy-preserving signal processing. In particular he is working on the processing of encrypted ECG signals, and privacy protection of biometric data. In 2009–2010 he visited Philips Laboratories, The Netherlands, for six months where he developed a protocol for quality evaluation of encrypted signals.

Ahmad-Reza Sadeghi received the Ph.D. degree in computer science with the focus on privacy protecting cryptographic systems from the University of Saarland, Germany.

He is the head of the System Security Laboratory at Technical University Darmstadt as well as at Fraunhofer Institute for Secure Information Systems (SIT) in Darmstadt, and guest professor at Ruhr-University Bochum (RUB) in Germany. Prior to academia, he worked in research and development of telecommunications enterprises with, amongst others, Ericson Telecommunications. Currently, he leads several international research and development projects on design and implementation of trustworthy computing platforms, security hardware, particularly physically unclonable functions (PUF), cryptographic privacy-protecting systems, and cryptographic compilers, in particular for secure computation. He has served as a program chair or committee member for a variety of conferences and workshops on information security, trusted computing, and applied cryptography. His main research interests are security architectures, cryptographic protocols, and security hardware.

Thomas Schneider received the Masters degree in computer science with distinction from Friedrich-Alexander University Erlangen-Nuremberg, Germany, in 2008. He received the Ph.D. degree (*summa cum laude*) in Engineering Secure Two-Party Computation Protocols from Ruhr-University Bochum, Germany.

In 2007, he visited Alcatel-Lucent Bell Laboratories, Security Solutions/Cryptographic Systems, NJ, for six months as stipendiary of the German National Academic Foundation (Studienstiftung des Deutschen Volkes) doing research for his Master's thesis on "Practical Secure Function Evaluation." From 2008 to 2011, he was research assistant at the Horst Görtz Institute for IT Security, Ruhr-University Bochum, doing active research in several international projects on cryptographic compilers (CACE), processing of encrypted signals (SPEED), and the European Network of Excellence in Cryptology (ECRYPT II). Since March 2011, he has been a postdoctoral researcher at CASED, TU Darmstadt, Germany. His research interests include privacy-preserving protocols, in particular their efficiency, automatic generation, and combination with (un)trusted hardware.