General Function Evaluation in a STPC Setting via Piecewise Linear Approximation

Tommaso Pignata [1], Riccardo Lazzeretti [2], Mauro Barni [3]

Information Engineering Department, University of Siena, Italy

[1] pignata.tommaso@gmail.com, [2] riccardo.lazzeretti@gmail.com, [3] barni@dii.unisi.it

***Abstract*—While in theory any computable functions can be evaluated in a Secure Two Party Computation (STPC) framework, practical applications are often limited for complexity reasons and by the kind of operations that the available cryptographic tools permit. In this paper we propose an algorithm that, given a function $f()$ and an interval belonging to its domain, produces a piecewise linear approximation $\widetilde{f}()$ that can be easily implemented in a STPC setting. Two different implementations are proposed: the first one relies completely on Garbled Circuit (GC) theory, while the second one exploits a hybrid construction where GC and Homomorphic Encryption (HE) are used together. We show that from a communication complexity perspective the full-GC implementation is preferable when the input and output variables are represented with a small number of bits, otherwise the hybrid solution is preferable.**

I. Introduction

In the last years an increasing attention has been given to the development of tools for processing encrypted signals [1]. This interest is due to the call for security stemming from applications where two or more non-trusted parties wish to collectively process one or more signals to reach a common goal. While the parties involved have in common the same goal, they do not trust each other and are not available to disclose their private inputs to the other parties. Therefore there is the necessity 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, say Alice, owns a signal that has to be processed in some way by the other party, hereafter referred to as Bob. Since Alice and Bob do not trust each other, Bob is required to process the signal owned by Alice without getting any information about it, not even the result of the processing. At the same time, Bob wants to protect the information it uses to process the signal provided by Alice.

Several cryptographic primitives exists, that once coupled with a suitable design of the underlying signal processing algorithms, allow one 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 s.p.e.d. (standing for Signal Processing in the Encrypted Domain), or SMPC (for Secure Multi-Party Computation) techniques (STPC: Secure Two Party Computation, if only two parties are involved). The main STPC protocols, used in privacy preserving applications, are based on Homomorphic Encryption (HE) and Garbled Circuits (GC).

The number of possible applications of STPC techniques is virtually endless, but in practice s.p.e.d. applications are often limited by the kind of operations that can be performed on the data. HE allows one to evaluate only linear operations on encrypted data and in the past years many papers have been published with protocols (having high complexity) able to compute particular functions such as bit decomposition, comparison [2], division [3], etc. On the other side GC allows one to evaluate any function that can be represented with an acyclic binary circuit. Some functionalities have simple implementations, such as comparison, addition, multiplexer, etc., while others have higher complexity, such as product, division [4], logarithm [5], etc. Hybrid protocols, based on HE and GC, have been proposed to take advantage of the strong points of the two techniques, so that complex protocols are developed as a concatenation of subprotocols, some of them implemented by using HE and other by using GC [9], [10]. To connect HE and GC subprotocols, an interface protocol has been proposed in [11]. Recently Fully Homomorphic Encryption (FHE) schemes [6], [7], [8] have been proposed. They permit to evaluate any boolean function without interaction by a party owning only the public key. Unfortunately, FHE is still not efficient, due to the bit-length of ciphertexts (each one encrypting a single bit) and public key. Despite the great deal of research carried out so far, we are still lacking a generic and efficient way to implement any function, such as trigonometric, hyperbolic and statistical functions, in a STPC framework.

In this paper we propose a general approach that permits one to evaluate any function having limited domain and codomain through a piecewise linear approximation. We suppose that both the input and output values of the function are secret (input comes from previous computation and output is used for further computation, hence they can not be revealed to the parties involved to avoid any leakage of information) while the function is part of a known protocol and hence is public.

The main contribution of this paper is a two-phase protocol that permits to evaluate a linear approximation of a function $f()$, given parameters like the maximum representation error and the accuracy representation of variables, in a STPC scenario. In the first phase the function is processed in the plain domain so that a particular piecewise linear representation, permitting an easy and efficient implementation in the encrypted domain, is obtained. In the second phase the approximation is implemented by using STPC tools. We provide two different solutions: the first one based only on GC and the second one based on a hybrid protocol, obatined by combining GC and HE. Finally we show that the full-GC solution is preferable when the input is represented with a small number of bits, otherwise the hybrid solution is preferable.

This paper is organized as follows. In Section II the main cryptographic tools are presented. The main idea of the protocol is described in Section III, while its STPC implementations are presented in Section IV and finally compared in Section V. We conclude the paper with some conclusions and proposals for future works in Section VI.

II. Cryptographic Tools

We start by presenting the cryptographic primitives our protocol relies on, namely Garbled Circuits (GC) and Homomorphic Encryption (HE).

WIFS'2012, December, 2-5, 2012, Tenerife, Spain. 978-1-4244-9080-6/10/$26.00

Throughout the paper we adopt the semi-honest security model, where the parties involved follow the protocol as prescribed but try to learn as much as possible from the messages exchanged and their private inputs. The security of the part of our protocol based on GC is guaranteed by the security of the GC primitive, while the part implemented by using HE is secure, given the IND-CPA property of the encryption protocol used. The security of the hybrid protocol composed by more secure sub-protocols (some based on GC and some on HE) is proven in [11].

A. Garbled Circuit

In our protocol we adopt the GC construction outlined in [12] for the secure evaluation of functionality represented by a Boolean circuit C in the two party setting. Security of the approach has been proved in [13]. We summarize the main ideas in the following. First the **constructor** (say Bob), creates a *garbled circuit* $\widetilde{C}$: for each wire W_i of the circuit, he randomly chooses a *complementary garbled value* $\widetilde{W}_i = \langle \widetilde{w}_i^0, \widetilde{w}_i^1 \rangle$ consisting of two secrets, $\widetilde{w}_i^0$ and $\widetilde{w}_i^1$, where each $\widetilde{w}_i^j$ is the *garbled value* of W_i's value j and is represented with t bits, (t is a security parameter); further, for each gate G_i, Bob creates a *garbled table* $\widetilde{T}_i$ such that given a set of garbled G_i's inputs, $\widetilde{T}_i$ allows one to recover the garbled value of the corresponding G_i's output, and nothing else. The Garbled tables, having size $4t$ bits each, are transmitted to the **evaluator** (say Alice). Then garbled values corresponding to Alice's input wires x_j are (obliviously) transferred to Alice. An 1-out-of-2 Oblivious Transfer (OT) protocol [14] is a two-party protocol, where Bob inputs a pair of ℓ-bit strings $S = \langle s^0, s^1 \rangle$ with $s^0, s^1 \in \{0,1\}^\ell$ and Alice inputs a choice bit $b \in \{0,1\}$. At the end of the protocol, Alice learns s^b, but nothing about s^{1-b} whereas Bob learns nothing about b. Within a GC protocol, the OT protocol is used to associate the t-bit long secrets to the input bits of Alice: Bob inputs complementary garbled values $\widetilde{W}_j$ into the protocol; Alice inputs x_j and obtains $\widetilde{w}_j^{x_j}$ as output. Now, Alice can evaluate the garbled circuit $\widetilde{C}$ to obtain the garbled output simply by evaluating the garbled circuit gate by gate, using the garbled tables $\widetilde{T}_i$.

The basic GC protocol outlined above can be improved in many ways as shown in [15] and [16]. In particular, in [15] the authors suggest to use a point and permute technique to decrypt directly the correct row of the table and replace encryptions by Hash functions, the scheme proposed in [16] allows "free" evaluation of XOR gates so that a XOR gate has no garbled table associated and its evaluation consists of XOR-ing its garbled input values, resulting in *no communication* and *negligible computation*. Several OT implementations have been proposed, such as the one described in [17] that is secure against malicious Bob and semi-honest Bob in the random-oracle model and has asymptotically communication complexity approximately equal to 6t bits. Considering that OT can be precomputed [18], many OT's can be evaluated off-line on random values (independent from the real values used during the circuit evaluation) and resulting in an on-line communication complexity approximately equal to 2t bits.

For the implementation of our protocol, we need many GC sub-protocols implementing the following functionalities, whose description can be found in [19], [16]. The sub-circuits used have a number of gates that depends linearly on the input bit-length, except for the product circuit which has a squared complexity.

B. Homomorphic Encryption

With a semantically secure, additively homomorphic, asymmetric encryption scheme, it is possible to compute the encryption of $[\![a+b]\!]$ by the encryptions of $[\![a]\!]$ and $[\![b]\!]$, i.e. $[\![a+b]\!] = [\![a]\!] \boxplus [\![b]\!]$. An additive homomorphic scheme also permits to compute the product between an encrypted number and a public factor by repeated sums, i.e. $[\![ab]\!] = [\![a]\!] \boxdot b$. A commonly used additively homomorphic cryptosystem is the Paillier cryptosystem [20] which has plaintext space $\mathbb{Z}_N$ and ciphertext space $\mathbb{Z}_{N^2}^*$, where N is a T-bit RSA modulus and a ciphertext is represented with $2T$ bits. By using Paillier cryptosystem it is possible to evaluate the encryption of the sum of two values through the product of the corresponding ciphertexts $[\![a+b]\!] = [\![a]\!] \cdot [\![b]\!]$ and the product between two values, one of them public, through exponentiation: $[\![ab]\!] = [\![a]\!]^b$.

C. Hybrid protocols

The use of hybrid protocols, such as in [5], [9], permits to efficiently evaluate functionalities for which full-HE or full-GC solutions would not be efficient (or even impossible). Due to the different representation of the data for GC and HE, conversion from homomorphic ciphertexts to garbled secrets (or vice versa) must be performed by using additive blinding. In particular, by referring to the protocols described in [11], it is easy to derive that the conversion of an ℓ-bit long value from HE to GC requires the on-line transmission of additional $2T+7\ell t$ bits, while conversion from GC to HE requires an overhead of $2T+(\ell+\tau)5t$ bits, where τ is an obfuscation security parameter (usually $\tau = 80$).

III. Function approximation in ciphertext space

Given a generic, but limited, function $f()$ with domain $[x_a, x_b)$ and matched codomain $[y_a, y_b]$, our goal is to find an approximation that can be efficiently calculated in the ciphered domain.The easiest solution consists to represent $f()$ through a piecewise linear function $\widetilde{f}()$ in a discrete space, obtained by quantizing the domain, i.e $[x_a, x_b)$ is mapped into a sequence of integer numbers $\{\hat{x}_0, \ldots, \hat{x}_n\}$. Each segment of the approximation is fully characterized by its extreme points, i.e. the k-th segment is characterize by the couples $(\hat{x}_k, \hat{y}_k)$ $(\hat{x}_{k+1}, \hat{y}_{k+1})$. Considering that each extreme is common to two segments, the $N+1$ couples $(\hat{x}_0, \hat{y}_0)$ $(\hat{x}_1, \hat{y}_1)$ $\ldots$ $(\hat{x}_N, \hat{y}_N)$ are sufficient to describe N segments.

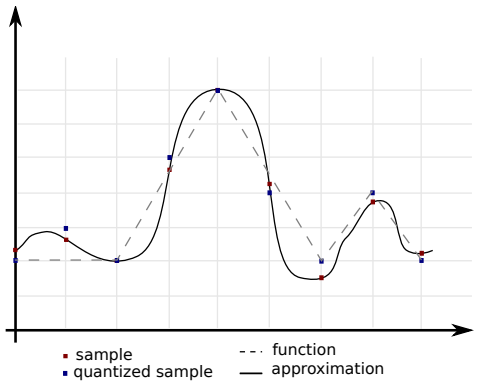

Figure 1. Sampled function f and its approximation.

Considering that STPC protocols work with integer values, we need to sample and quantize the values of $f()$, introducing an approximation error ϵ_d that can be reduced at will. ϵ_d decreases by choosing a smaller step during the sampling of the input x or by using an higher quantization factor on the output y. These parameters affect the number of bits b_x and b_y used to represent the values $\hat{x}_0 \ldots \hat{x}_n$ and $\hat{y}_0 \ldots \hat{y}_n$ respectively. Note that $\hat{x} = \lceil \left(\frac{2^{b_x}}{x_b - x_a} \right) x \rfloor$ and

$\hat{y} = \lceil \left(\frac{2^{b_y}}{y_b - y_a} \right) y \rfloor$ and the sampled and quantized function is $\widehat{f}()$. We underline that, being x the output of a previous secure computation, the bit-length b_x, and hence the quantization step, are usually imposed by the operations previously performed.

Two steps are needed to calculate the approximated value $\hat{y}$ of the function in a generic point $\hat{x} \in \{\hat{x}_0 \dots \hat{x}_n\}$:

1) the interval c containing $\hat{x}$ is selected,
2) $\widetilde{f}(\hat{x})$ is computed by through linear interpolation.

The first step requires $(N-1)$ comparisons because we know that we always have $\hat{x} \geq \hat{x}_0$ and $\hat{x} < \hat{x}_n$. Obviously only an interval c exists such that $\hat{x}_c \leq \hat{x} < \hat{x}_{c+1}$. The second step consists in a linear interpolation inside the line previously selected, computed by the formula:

$$\hat{y} = \frac{\hat{y}_{c+1} - \hat{y}_c}{\hat{x}_{c+1} - \hat{x}_c}(\hat{x} - \hat{x}_c) + \hat{y}_c. \tag{1}$$

Taking into account that (1) needs to be calculated on discrete values, the ratio

$$m_c = \frac{(\hat{y}_{c+1} - \hat{y}_c)}{(\hat{x}_{c+1} - \hat{x}_c)} \tag{2}$$

introduces an additional error when the numerator is not a multiple of denominator. To solve the problem, all the coefficients m_i need to be quantized. To avoid introducing other errors, we can multiply them by $\alpha = \text{lcm}\{\hat{x}_{i+1} - \hat{x}_i\}_{i=0}^{N-1}$, so that for each interval i, $(\hat{x}_{i+1} - \hat{x}_i)$ always divides $\alpha(\hat{y}_{i+1} - \hat{y}_i)$. Of course the final result is amplified by a factor α. Note that the term (2) of each interval can be pre-computed since it does not depend on $\hat{x}$. A different solution consists to evaluate the division in the ciphered space, but this operation has a high complexity [3], [4].

Unluckily the α value can be very big, hence we decided to impose the length of the intervals (the distance between the extremes) in which the piecewise function is decomposed to be powers of two, i.e. $\hat{x}_{i+1} - \hat{x}_i = 2^{b_i}$ for some b_i. This constrain has two purposes:

- the value of α is easier to calculate, being the length of the longest intervals, hence representable with $b_v = \max_i b_i$ bits;
- the final result of the protocol is amplified by a factor α and has a representation accuracy of 2^{-b_v}.

The formula to calculate the function approximation for a general $\hat{x} \in \{\hat{x}_0 \dots \hat{x}_n\}$ then becomes

$$\hat{y}\alpha = M_c(\hat{x} - \hat{x}_c) + Q_c, \tag{3}$$

where

$$\begin{aligned} M_c &= (\hat{y}_{c+1} - \hat{y}_c)\frac{\alpha}{(\hat{x}_{c+1} - \hat{x}_c)}, \\ Q_c &= \hat{y}_c\alpha. \end{aligned} \tag{4}$$

We can easily observe that:

- the M_i terms are always integers and can be pre-computed;
- the Q_i terms can be obtain by left-shifting $\hat{y}_i$.

Having amplified the m_i coefficients by α, the approximation function is suitable to be computed in the ciphered space. The only drawback is that the result has some more least significant bits that can be simply ignored.

We now describe the procedure to find a piecewise linear approximation $\widetilde{f}()$ of a function $f()$, given a maximum representation error $\epsilon_{s_{\max}}$, arbitrarily chosen. First of all $f()$ is discretized: 2^{b_x} samples are chosen in the interval $[x_a, x_b]$ with constant step and the corresponding y values are quantized and represented by using b_y bits, so that their approximation also lies in a discrete space. The piecewise linear function $\widetilde{f}()$ we aim to is fully described by its discontinuity points$(\hat{x}_0, \hat{y}_0)$ $(\hat{x}_1, \hat{y}_1)$ $\dots$ $(\hat{x}_N, \hat{y}_N)$. The approximation must introduce an error $\epsilon_s \leq \epsilon_{s_{\max}}$. Given an interval in the space of samples, the approximation error $\epsilon_{s,i}$ of the piecewise linear function in the interval is the largest distance between a sample and the corresponding point on the segment, i.e. $\epsilon_{s,i} = \max_{\hat{x} \in \{\hat{x}_i \dots \hat{x}_{i+1}\}} |\hat{f}(\hat{x}) - \widetilde{f}(\hat{x})|$, where $\hat{f}()$ represents the discrete approximation of $f()$.

We propose a two-step algorithm that generates that piecewise linear function $\widetilde{f}()$ starting from the samples of $\hat{f}()$.

1) The first step is a bisection algorithm. At the beginning the whole function is approximated in the interval $\{\hat{x}_0 \dots \hat{x}_n\}$ with the segment connecting $(\hat{x}_0, \hat{y}_0)$ to $(\hat{x}_n, \hat{y}_n)$. We easily obtain that

$$\hat{y} = \frac{\hat{y}_n - \hat{y}_0}{\hat{x}_n - \hat{x}_0}(\hat{x} - \hat{x}_0) + \hat{y}_0 \ \forall \hat{x} \in \{\hat{x}_0 \dots \hat{x}_{n-1}\}, \tag{5}$$

where

$$\epsilon_s = \max_{i=0..N} \left\{ \left| \left[\frac{\hat{y}_n - \hat{y}_0}{\hat{x}_n - \hat{x}_0}(\hat{x}_i - \hat{x}_0) + \hat{y}_0 \right] - \hat{y}_i \right| \right\}. \tag{6}$$

If $\epsilon_s \leq \epsilon_{s_{\max}}$ the piecewise linear is fully characterized by the extremes $(\hat{x}_0, \hat{y}_0)$, $(\hat{x}_n, \hat{y}_n)$. Otherwise the error is too large and the function domain is split into two parts. The procedure is repeated on each half, trying to approximate each of them with a single segment and so on until $\epsilon_{s,i} \leq \epsilon_{s_{\max}}$ in each interval obtained. This procedure always ends after at most b_x iterations. At the end of this step a correct $\widetilde{f}$ is obtained even if it may be suboptimal.
2) Each pair of consecutive intervals is merged if
   - both the intervals have the same length, resulting in a merged interval that still respects the constraint $(\hat{x}_{(i+1)} - \hat{x}_i) = 2^{b_i}$;
   - the error of the approximation in the new interval is still lower than $\epsilon_{s_{\max}}$.

   This step is iterated until there are no more intervals that can be joined.

Let consider for example the function $y = \frac{1}{5}x^3 + \frac{4}{5}x^2 - \frac{7}{5}x - 2$ in the interval $[-6, 3)$. Applying the procedure described above with different values of $b = b_x = b_y$ and setting different errors ϵ, we obtain the number of segments shown in Table I, where the / symbol denotes that it is not possible to obtain an approximation with the given parameters.

Table I
OF INTERVALS FOR ERROR LIMITS

$\epsilon \backslash b$	4	8	12	16	20	24
0.1	/	25	24	24	24	24
0.01	/	/	18	73	73	73
0.001	/	/	/	219	212	211

IV. STPC IMPLEMENTATIONS

As already outlined in Section II, the main tools for STPC are HE and GC. Moreover, it is also possible to develop hybrid protocols by composing subprotocols, each one implemented with the most suitable solution.

We excluded a priori a purely homomorphic solution since the protocol introduced in Section III needs too many comparisons, for which no efficient homomorphic implementation exists. Hence the choice is between a full-GC solution and a hybrid solution. We start by describing both the protocols and then we analyse them to estimate which is the most efficient.

We assume that both the input $\hat{x}$ and the output $\hat{y}$ are secret and available to one of the two parties: during GC computation they are available to Alice (the evaluator) in the form of garbled secrets, while during HE computation are available to Bob in the form of ciphertexts. This assumption mimics a case in which the function evaluation protocol is embedded inside an outer protocol.

A. Full-GC implementation

To evaluate the function $f()$ by using a full-GC protocol, we need to describe the circuit that implements its piecewise approximation. The circuit can be split into two parts resembling the structure of the algorithm.

1) The sub-circuit of Figure 2 selects the interval c the input $\hat{x}$ belongs and returns the parameters $\hat{x}_c|\hat{y}_c|M_c$ associated to the interval ("|" denotes concatenation). For each segment i, only three input parameters are needed: the coordinates of the left extreme $(\hat{x}_i, \hat{y}_i)$, which are respectively b_x and b_y bit-long, and the parameter M_i -defined in (4)- which is $(b_y + b_v)$ bit-long.

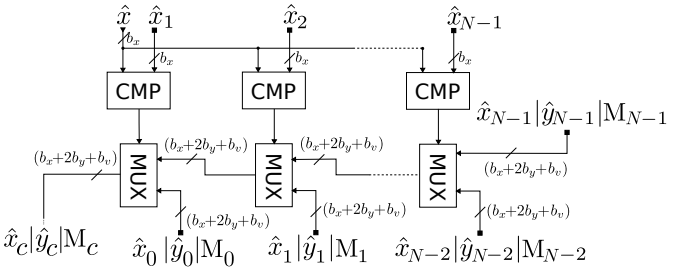

Figure 2. Pure garbled circuit protocol (first part).

The circuit is composed by $N-1$ comparators CMP and $N-1$ multiplexers MUX. The inputs of the i-th CMP, with $i = 1 \dots N-1$, are $\hat{x}$ and the left extreme of the interval $\hat{x}_i$, both of them represented with b_x bits. The output is 1 if $\hat{x} \geq \hat{x}_i$, 0 otherwise. The i-th MUX chooses between the output of the $(i+1)$-th MUX ($\hat{x}_{N-1}|\hat{y}_{N-1}|M_{N-1}$ for the $(N-1)$-th MUX) and $\hat{x}_{i-1}|\hat{y}_{i-1}|M_{i-1}$ according to the output of the i-th CMP, i.e. if the output of CMP is 0, the corresponding MUX propagates $\hat{x}_{i-1}|\hat{y}_{i-1}|M_{i-1}$, otherwise it propagates the output of the previous (right) MUX (or $\hat{x}_{N-1}|\hat{y}_{N-1}|M_{N-1}$ for the $(N-1)$-th MUX). The inputs of the multiplexer are represented with $(b_x + b_y + (b_y + b_v))$ bits.

Because all the parameters are obtained by a well known plain protocol applied to the public function $f()$, they are also considered known to both parties. Hence it is convenient to embed these values within the circuit as constants, thus resulting in a lower number of input bits from the garbler and a reduction in the size of the circuit. The only input to the circuit is then $\hat{x}$ which is b_x bit-long.

This sub-circuit outputs the secrets relative to the three parameters characterizing the segment approximating $f()$ within the interval that contains $\hat{x}$.

2) The second part of the circuit (see Figure 3) implements equation (3). Some simplifications can be applied knowing that the subtractor never outputs values greater than α (which is b_v bit-long) and that the multiplier never outputs values greater than $\hat{y}_{\max}$ which is b_y bit-long. This part of the circuit does not need any additional input from the garbler or from the evaluator. The final output is $\hat{y}$, represented with $(b_y + b_v)$ bits, but the least significant b_v bits can be ignored because they provide extra value precision.

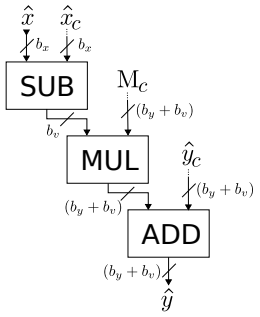

Figure 3. Pure garbled circuit protocol (second part).

The main advantage of the full-GC implementation is the use of only one cryptographic primitive. If the function $f()$ is part of a protocol, where the previous and following functionalities are also implemented by using only GC, the integration of the sub-protocols that approximate $f()$ would be very easy.

B. Hybrid implementation

In this protocol we consider that for the GC part Bob acts as garbler and Alice as evaluator, while in the HE part, Alice owns the private key and distributed the public key to Bob.

Our solution is composed by a GC section, implementing the interval identification, followed by a HE section, implementing approximation. We assume that the hybrid implementation is placed between two homomorphic sub-protocols, hence both the input $\hat{x}$ and the output $\hat{y}$ are available to Bob in encrypted form. For the conversion of the ciphertext $[\![\hat{x}]\!]$ in the corresponding garbled secrets, Bob generates a random number r, having bit-length $b_x + \tau$, where τ is the security obfuscation parameter defined in Section II-C, then he uses r to additively obfuscate $[\![\hat{x}]\!]$ and sends the result $[\![\hat{x} + r]\!]$ to Alice, who can decode it and use the least b_x significant bits of $(\hat{x} + r)$ as input to the garbled circuit of Figure 4, while Bob inputs the random value r together with $(N-1)$ random obfuscation bits $r_1, r_2, \dots r_{N-1}$, used to hide the outputs of the circuit to Alice. The circuit starts by removing the obfuscation r through a subtractor and, similarly to the first part of the full-GC solution, the obtained $\hat{x}$ is used in $N-1$ comparison circuits. The XOR between the output of two adjacent CMP circuits is computed, returning a sequence of bits b_i, where only $b_c = 1$. Finally, the b_i bits are obfuscated by using XOR gates and the array $\{[\![(b_i \oplus r_i)]\!]\}_{i=0}^{N-1}$ is output to Alice.

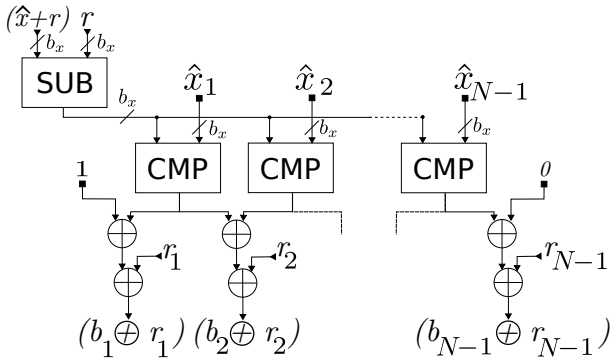

Figure 4. Garbled circuit used inside the hybrid protocol.

In the meanwhile Bob interpolates $[\![\hat{y}_i]\!] = [\![\hat{x}]\!]^{M_i} \cdot [\![-M_i \cdot \hat{x}_i + \hat{Q}_i]\!]$, obfuscates all $[\![\hat{y}_i]\!]$ with new random values s_i and sends them to Alice. Alice decrypts all the $(N-1)$ $[\![\hat{y}_i + s_i]\!]$ and multiplies each

of them for the obfuscated bits computed by the garbled circuit. Then she encrypts and sends them back to Bob together with the encryption of the obfuscated bits, i.e. $[\![b_i \oplus r_i]\!]$. At this point Bob can retrieve all the ciphertexts $[\![\hat{y}_i \cdot b_i]\!]$ with the formulas

$$[\![y_i \cdot b_i]\!] = \begin{cases} [\![(b_i \oplus r_i)(\hat{y}_i \cdot s_i)]\!] \cdot [\![b_i \oplus r_i]\!]^{(-s_i)} & \text{if } r_i = 0, \\ [\![(b_i \oplus r_i)(\hat{y}_i \cdot s_i)]\!]^{(-1)} \cdot [\![\hat{y}_i]\!] \cdot [\![b_i \oplus r_i]\!]^{s_i} & \text{if } r_i = 1. \end{cases}$$

Finally Bob adds all the $[\![\hat{y}_i \cdot b_i]\!]$ by using the homomorphic property. Considering that only the bit b_c associated to the correct interval assumes value 1, while the others are zero, Bob obtains $[\![\hat{y}_c]\!] = \prod_{i=0}^{N-1} [\![\hat{y}_i \cdot b_i]\!]$.

V. Protocols comparison

In this section we are going to compare the protocols described so far. Providing a computation complexity comparison is a difficult task, since the complexity of cryptographic tools depends on the different primitives: Garbled Circuits are based on Hash functions, while Homomorphic Encryption is based on different computation performed on big values (discrete exponentiations in the Paillier cryptosystem). Hence in the following we focus only on communication complexity analysis.

The parameters involved in the protocol are summarized in Table II together with their bit-lengths and the intervals of values we assigned them for the communication complexity analysis. By assuming short term security [21], for homomorphic cryptography we need a standard security parameter of 1024 bits which means that every secret is $2 \cdot 1024 = 2048$ bits long, while for the all garbled circuits we assume a security parameter of 80 bits.

Table II
Protocols parameters. α and number of segments are resulted by tests.

content	name	value(s)
bits for quantization of x axis	b_x	$8-24$
bits for quantization of y axis	b_y	$8-24$
bits for quantization of α	b_v	$4-12$
number of segments	N	$10-210$
homomorphic security parameter	T	1024
garbled circuit security parameter	t	80
obfuscation parameter for HE to GC conversion	τ	80

For the implementation of GC we use precomputation of oblivious transfers, so that an OT_2^1 over t-bit secrets needs the online transmission of $\sim 2t$ bits and for each non-XOR gate $4t$ bits are transferred, while no communication is needed for the XOR gates.

It is important to remember that the parameters $\hat{x}_i|\hat{y}_i|M_i$ are public and hence managed as constants in the circuits. The number of gates in the circuit can be reduced according to their values. For example if a constant bit assuming the value 1 is put at the input of an AND gate, the gate can be erased and the other input bit can be propagated to the following gates. A difficult task is to estimate the number of non free gates that can be eliminated due to constant inputs, especially with large circuits implementing an approximation with many segments.

For the full-GC solution we estimated the impact of the above simplifications by actually constructing several circuits. To evaluate the simplification of the circuit obtained by wiring the constant inputs, we used circuits implementing approximations of the polynomial $y = \frac{1}{5}x^3 + \frac{4}{5}x^2 - \frac{7}{5}x - 2$ in the interval $[-6, 3)$. We took the cases where $b_x = b_y = b$ with $b \in \{8, 12, 16, 20, 24\}$. For each different resolution of input and output three tests were run, changing the desired $\epsilon_{s_{\max}}$ in the set $\{0.1, 0.01, 0.001\}$. In each test we built both a simplified circuit and one without simplification. Then the ratio between the number of non free gates of the first and number of non free gates of the second was computed. The mean value of these tests is 0.56 with small variance, so the transmitted bits related to the number of non free gates becomes $0.56(2(N-1)b_x + 3b_y + b_v + 2b_v b_y) * 4t$ on the average.

The number of non-XOR gates and the bits transmitted for all the sub-circuits are shown in Table III, where we do not take into account the simplification given by the constant inputs, since the number of non free gates eliminated depends heavily from the values of the inputs. As already said, we assume that the $\hat{x}$ input comes from a previous GC computation and the corresponding secrets are already available to the evaluator, hence there is no transmission for the input. Similarly we have no transmission for the output, that is used in further computations.

For the hybrid protocol, we implemented the garbled circuit part similarly to the full-GC construction obtaining the complexities shown in Table IV and Table V. For the homomorphic part we just considered secrets transmission.

In the hybrid protocol wiring constants is possible only inside the comparators where one of the inputs is always constant. We can easy verify that if the least significant constant bit is true we can eliminate one non free gate. If the two least significant constant bits are both true we can eliminate two non free gates and so on. In general the probability of eliminating i gates is equal to 2^{-i}. In a comparator of b_x bits we have an average reduction of $\sum_{i=1}^{b_x} i2^{-i} = 2^{-b_x}(-b_x + 2^{b_x+1} - 2)$. Varying the values b_x and N similarly to the full-GC protocol, we can calculate the average ratio between the number of non-free gates in the circuit with wired constants and the standard circuit obtaining an average of 0.78. Hence the transmitted bits relative to the circuit gates becomes $0.78Nb_x * 4t$.

Alice inputs the obfuscated value to the circuit hence $2tb_x$ bits are transmitted during the OT, while the transmission of the secrets regarding the obfuscation values (r and r_i) known by Bob needs other

Table III
Number of non free gates in garbled circuits (without constant reduction)

Sub circuit	# of non free gates	# of transmitted bits
Comparators	$(N-1)b_x$	$(N-1)b_x * 4t$
Multiplexers	$(N-1)b_x + 2b_y + b_v$	$((N-1)b_x + 2b_y + b_v) * 4t$
Subtractor	b_v	$b_v * 4t$
Adder	$b_v + b_y$	$(b_v + b_y) * 4t$
Multiplicator	$2b_v(b_y - 1)$	$2b_v(b_y - 1) * 4t$
Total	$2(N-1)b_x + 3b_y + b_v + 2b_v b_y$	$(2(N-1)b_x + 3b_y + b_v + 2b_v b_y) * 4t$

Table IV
Number of non free gates in garbled circuits (without constant reduction)

Sub circuit	# of non free gates	# of transmitted bits
Comparator	$(N-1)b_x$	$(N-1)b_x * 4t$
Subtractor	b_x	$b_x * 4t$
Total	Nb_x	$Nb_x * 4t$

Table V
Number of bits transmitted for IO in garbled circuits

		# of IO bits	# of transmitted bits
Evaluator	Input	b_x	$b_x * 2t$
	Output	$(N-1)$	$(N-1)2t$
Garbler	Input	$b_x + (N-1)$	$(b_x + (N-1))t$
	Output	0	0
Total		$2(b_x + (N-1))$	$3(b_x + (N-1))t$

$(b_x+(N-1))t$ bits. Moreover, in the protocol $1+(N-1)$ ciphertexts are transmitted from Alice to Bob and $2(N-1)$ ciphertexts are transmitted from Bob to Alice for a total of $2T(3N-2)$ bits.

In summary, $0.78Nb_x * 4t + 3(b_x + (N-1))t + 2T(3N-2)$ bits are transmitted by the hybrid protocol.

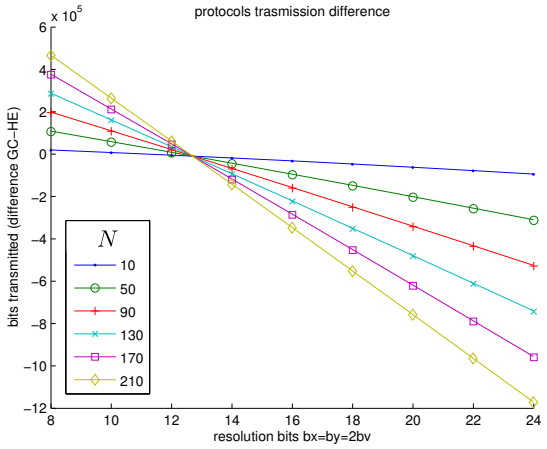

Figure 5. Transmission difference between hybrid protocol and full-GC protocol.

Figure 5 shows the difference between the communication complexity of the hybrid protocol and the one of the pure GC protocol with different number of bits used to represent $\hat{x}$ and $\hat{y}$ (in the x-axis) and with different number N of segment (in the y-axis). When the difference is positive the pure GC protocol is more convenient than the hybrid one from a communication complexity point of view, while when it is negative the hybrid takes less bits to be transmitted. The different values of N are represented with different plots.

During the tests, we considered a case in which the domain and codomain are represented by the same number of bits $b = b_x = b_y$ and $b_v = \frac{b_x}{2}$ (we verified experimentally that this value is sufficient). We let N vary from 10 to 210 with step 40 and b from 8 to 24 with step 2. As we can see the pure garbled circuit protocol is convenient whenever x and y are represented with less than 12 bits.

VI. Conclusion

Given a function $f()$ and an interval belonging to its domain, we considered the problem of approximating $f()$ by means of a piecewise linear function $\widetilde{f}()$ in a STPC setting. We have described two possible protocols implementing the approximation. The first one relies completely on Garbled Circuit theory, while the other one relies on a hybrid solution where GC and Homomorphic Encryption are used together. Considering the presence of constant public inputs to the circuit, the number of non-XOR gates can be reduced in both the implementations. The reduction depends on the particular value that each public input assumes and hence on the function we are going to approximate and the number of bits used to represent the involved values. We used a polynomial function as an example, obtaining for the full-GC circuit an average reduction of 0.56, and 0.78 for the hybrid protocol. In this setting, we have shown that hybrid solution is preferable when $b_x > 12$ bits.

The above result obviously depends on the to-be-approximated function. In practice, given a function and a desired maximum error, the designer has to vary many parameters, i.e. b_x, b_y and b_v, and calculate for each of them the number of segments. Once described the circuits for the different parameters, a circuit optimization protocol is run and only at this point the best configuration (including the choice between the full-GC and the hybrid protocol) can be chosen.

In the future, we plan to extend our analysis to multivariate functions $f(x_1, \ldots, x_n)$, and to consider a setting in which the function $f()$ has to be kept secret as well. Moreover non-linear approximation can be approached.

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 Journal on Information Security*, vol. 2007, p. 17, 2007.

[2] Z. Erkin, M. Franz, J. Guajardo, S. Katzenbeisser, I. Lagendijk, and T. Toft, "Privacy-preserving face recognition," in *Privacy Enhancing Technologies*. Springer, 2009, pp. 235–253.

[3] T. Veugen, "Encrypted integer division," in *Information Forensics and Security, WIFS 2010, IEEE International Workshop on*, 2010, pp. 1–6.

[4] R. Lazzeretti and M. Barni, "Division between encrypted integers by means of garbled circuits," in *Information Forensics and Security (WIFS), IEEE International Workshop on*, 29 2011-dec. 2 2011, pp. 1 –6.

[5] M. Barni, J. Guajardo, and R. Lazzeretti, "Privacy preserving evaluation of signal quality with application to ECG analysis," in *Information Forensics and Security – WIFS 2010, IEEE International Workshop on*. IEEE, 2010, pp. 1–6.

[6] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proceedings of the 41st annual ACM symposium on Theory of computing*. ACM, 2009, pp. 169–178.

[7] M. Van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, "Fully homomorphic encryption over the integers," *Advances in Cryptology–EUROCRYPT 2010*, pp. 24–43, 2010.

[8] C. Gentry and S. Halevi, "Implementing gentry's fully-homomorphic encryption scheme," *Advances in Cryptology–EUROCRYPT 2011*, pp. 129–148, 2011.

[9] M. Barni, P. Failla, R. Lazzeretti, A. Sadeghi, and T. Schneider, "Privacy-Preserving ECG Classification with Branching Programs and Neural Networks," *Information Forensics and Security – TIFS, IEEE Transactions on*, Jun. 2011.

[10] R. Lazzeretti, J. Guajardo, and M. Barni, "Privacy Preserving ECG Quality Evaluation," in *Multimedia and security (MM&SEC), Proceedings of ACM workshop on*. ACM, 2012.

[11] V. Kolesnikov, A. Sadeghi, and T. Schneider, "From dust to dawn: Practically efficient two-party secure function evaluation protocols and their modular design," Cryptology ePrint Archive, Report 2010/079, 2010. http://eprint. iacr. org, Tech. Rep., 2010.

[12] A. C. Yao, "How to generate and exchange secrets," in *IEEE Symposium on Foundations of Computer Science, FOCS'86*, 1986, pp. 162–167.

[13] Y. Lindell and B. Pinkas, "A proof of Yao's protocol for secure two-party computation," Electronic Colloq. on Comp. Complexity, ECCC Report TR04-063, 2004.

[14] S. Even, O. Goldreich, and A. Lempel, "A randomized protocol for signing contracts," *Communications of the ACM*, vol. 28, no. 6, p. 647, 1985.

[15] D. Malkhi, N. Nisan, B. Pinkas, and Y. Sella, "Fairplay — a secure two-party computation system," in *USENIX*, 2004, http://fairplayproject.net.

[16] V. Kolesnikov and T. Schneider, "Improved garbled circuit: Free XOR gates and applications," in *International Colloquium on Automata, Languages and Programming, ICALP*, vol. 5126, 2008, pp. 486–498.

[17] M. Naor and B. Pinkas, "Efficient oblivious transfer protocols," in *ACM-SIAM Symposium On Discrete Algorithms (SODA'01)*. Society for Industrial and Applied Mathematics, 2001, pp. 448–457.

[18] D. Beaver, "Precomputing oblivious transfer," in *Advances in Cryptology – CRYPTO'95*, ser. LNCS, vol. 963. Springer, 1995, pp. 97–109.

[19] V. Kolesnikov, A. Sadeghi, and T. Schneider, "Improved garbled circuit building blocks and applications to auctions and computing minima," *Cryptology and Network Security*, pp. 1–20, 2009.

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

[21] D. Giry and J. Quisquater, "Cryptographic key length recommendation, march 2009," http://keylength.com.