Secure Adaptive Filtering

Juan Ramón Troncoso-Pastoriza, *Student Member, IEEE*, Fernando Pérez-González, *Senior Member, IEEE*

Abstract

In an increasingly connected world, the protection of digital data when it is processed by other parties has arisen as a major concern for the general public, and an important topic of research. The field of *Signal Processing in the Encrypted Domain* has emerged in order to provide efficient and secure solutions for preserving privacy of signals that are processed by untrusted agents.

In this work, we study the privacy problem of adaptive filtering, one of the most important and ubiquitous blocks in signal processing nowadays. We present several use cases for adaptive signal processing, studying their privacy characteristics, constraints and requirements, that differ in several aspects from those of the already tackled linear filtering and classification problems. We show the impossibility of using a strategy based solely on current homomorphic encryption systems, and we propose several novel secure protocols for a privacy-preserving execution of the LMS (Least Mean Squares) algorithm, combining different SPED techniques, and paying special attention to the error analysis of the finite-precision implementations. We seek the best trade-offs in terms of error, computational complexity and used bandwidth, showing a comparison among the different alternatives in these terms, and we provide the experimental results of a prototype implementation of the presented protocols, as a proof of concept that showcases the viability and efficiency of our novel solutions. The obtained results and the proposed solutions are straightforwardly extensible to other adaptive filtering algorithms, providing a basis and master guidelines for their privacy-preserving implementation.

Index Terms

Privacy, Adaptive Filtering, Iterative Methods, Complexity, Error analysis.

Copyright (c) 2010 IEEE. Personal use of this material is permitted. However, permission to use this material for any other purposes must be obtained from the IEEE by sending a request to pubs-permissions@ieee.org.

J.R. Troncoso-Pastoriza is with the Signal Theory and Communications Department, University of Vigo, Vigo 36310, SPAIN, e-mail: troncoso@gts.uvigo.es

Prof. F. Pérez-González is with the Signal Theory and Communications Department, University of Vigo, Vigo 36310, SPAIN, with the Galician Research and Development Center in Advanced Telecommunications (GRADIANT), Vigo 36310, SPAIN, and with the Dept. of Electrical and Computer Engineering, University of New Mexico, Albuquerque, NM, 87131-0001 USA, e-mail: fperez@gts.uvigo.es

Secure Adaptive Filtering

I. Introduction

In modern society, digital data about individuals that could be considered to be highly personal, can be found relatively easily in the communication networks, especially the Internet. Although most people support the last decades' advances in digital networks, the sensitiveness of these data motivates an increasing concern about the public availability of personal data and the processing performed on them. On the other hand, signal processing researchers have traditionally focused on continuously improving the efficiency and robustness of the applied algorithms, while often leaving aside the crucial aspect of data privacy. Thus, advances in signal processing have not taken into account the trustworthiness of the parties that manage users' signals or the sensitiveness of the information contained within these signals. There are many application scenarios where the need for privacy is clearly present, mainly those in which biological signals (fingerprints, faces, iris, DNA, ECG signals, MRI images,...) are involved, as they hold extremely sensitive information about users or patients, and their privacy is traditionally addressed through written consents that represent the trust that users must put on the party or parties that process their signals.

Signal Processing in the Encrypted Domain (SPED) is an emergent research field that has arisen to effectively tackle the privacy problems involving signal processing. As an interdisciplinary area, it has faced from its birth the challenge of bringing together the views of the cryptographic and the signal processing communities in order to reach the target of efficiently applying privacy preserving techniques to common signal processing operations.

The theoretical grounds of Signal Processing in the Encrypted Domain come from the field of secure function evaluation, that was introduced by Yao in 1982 [1] (Secure two-party computation) through the now widely known *Millionares' problem*, and then generalized to Secure Multiparty Computation [2] (SMC). In the former setting, two millionaires wish to know who is the richest, without disclosing to the other their respective wealth. The solution proposed by Yao was based on the concept of *garbled circuits*. In spite of the generality of the presented solution, the inefficiency of its implementation for many applications has constituted the biggest obstacle for the development of this technology for many years, in such a way that the existence of efficient solutions for the secure execution of a generic function is still an open problem. Nonetheless, many efficient and secure techniques have been developed for specific applications in the past few years, building up a set of tools that foretell the potential of this technology.

Within this set of tools, the most efficient SPED primitives are those that exploit the properties of homomorphic encryption for performing some linear fixed operations, but most of the times Signal

Processing needs to go further, resorting to adaptive filtering algorithms, due to their greater flexibility, higher responsiveness when tracking the changes in the environment, their convergence to the optimal fixed solution when working in a stationary environment, and the fact that they are the optimal solution in settings where the information about the signal characteristics is not complete, offering a much better performance than fixed filters. Hence, a considerable number of practical signal processing applications make use of adaptive filters. As we will show, current homomorphic cryptosystems cannot directly deal with adaptive filters due to cipher blowup after a given number of iterations; on the other hand, full homomorphisms, like Gentry's [3], able of executing any circuit without the need of decryption, are still not practical, due to the huge size needed for the ciphertexts. In fact, the existence of practical fully homomorphic cryptosystems is still an open problem. Even though there are some linear transforms and basic operations that can be directly translated into homomorphic processing, the set is too limited, and when privacy is a concern, the solution cannot impose that these operations be replaced by simpler non-adaptive algorithms, as the negative impact on performance could virtually destroy the usefulness of the algorithm. This is especially true when the involved signals are not stationary, and the filter must track their changes over time.

In this work, we present several secure solutions for privacy-preserving adaptive filtering that involve homomorphic processing, garbled circuits and interactive protocols, in order to overcome the limitations of the three technologies, while profiting from their respective advantages. We take the LMS algorithm as a prototypical example of a relatively simple but powerful and versatile adaptive filter, and compare the privacy solutions for the execution of the algorithm in terms of computation and communication complexity. Furthermore, we also perform a comparison in terms of the effect of fixed-point arithmetic on the error that the algorithm produces. We show the trade-off that the combination of these different technologies establishes between precision, computational load and required bandwidth, and we look for the optimum configuration by proposing novel interactive protocols aimed at efficiently solving the cipher blowup problem, coming to several solutions that reach an optimum balance among the involved performance figures.

A. Notation

We will use indistinctly lowercase letters to represent classes in a ring $(\mathbb{Z}_n, +, \cdot)$ and a representative of that class in the interval $[0, n)$. $\lceil . \rfloor$ will represent the rounding function of a number to the nearest integer. The used vectors will be represented by lower-case boldface letters, whereas matrices will be represented by upper-case boldface letters. The encryption of a number x will be represented by $[\![x]\!]$, and the vector (matrix) formed by the encryptions of the vector $\boldsymbol{x}$ (matrix $\boldsymbol{X}$) will be represented by $[\![\boldsymbol{x}]\!]$ ($[\![\boldsymbol{X}]\!]$). When working with the binary representation of a number x, the encryption of the vector of binary bits of that representation will be denoted as $[\![x]\!]_b$.

The operations performed between encrypted and clear numbers will be indicated as if they were performed in the clear; e.g. $[\![\boldsymbol{X}]\!] \cdot \boldsymbol{b}$ will represent the encryption of $[\![\boldsymbol{X} \cdot \boldsymbol{b}]\!]$. Regarding the complexity calculations, the communication complexity of each protocol will be denoted by C_{cm}, and it will be measured in bits.

The rest of the paper is structured as follows: in Section II, we recall the fundamental algorithms for adaptive filtering whose secure processing versions we provide. Section III presents several exemplifying adaptive filtering scenarios where privacy constraints make necessary the use of a privacy-preserving protocol, together with the trust model in use within those scenarios. In Section IV, some basic concepts about secure computation are introduced. Section V reviews the existing solutions for SPED primitives, and their relationship with the posed problem of secure adaptive signal processing. Section VI presents our solutions for privacy-preserving adaptive filtering. Section VII is devoted to the evaluation of the presented protocols, in terms of bandwidth and computational complexity. A special attention is devoted to finite precision effects and error analysis in Section VII-B, as the private protocols work with fixed-point arithmetic. Finally, Sections VIII and VIII-A describe the practical implementation guidelines of the proposed algorithms, based on the prototypes we have built, and present the obtained results for their complexity evaluation. Section IX gives some conclusions and anticipates future research lines following those initiated in this work.

II. Iterative algorithms for Adaptive Filters

As a brief introduction to the implemented methods, we present a summary of the most representative adaptive filtering family of algorithms, the Stochastic Gradient Algorithms.

Stochastic Gradient Algorithms are characterized by the use of a non-deterministic estimate of the gradient, opposed to other gradient descent methods. The Least Mean Squares (**LMS**) algorithm, developed by Widrow and Hoff in 1960 [4], is the most characteristic algorithm of this family, for being a simple yet powerful and widely used adaptive filtering algorithm. It comprises two processes that jointly form a feedback loop: 1) a transversal filter $\boldsymbol{w}_n$ with N_E coefficients applied to the input sequence u_n, and 2) an update process of the coefficients of the transversal filter, based on the instantaneous estimation error e_n between the output of the filter y_n and a desired response d_n. For real signals, these two processes are expressed as

$$y_n = \boldsymbol{w}_n^T \boldsymbol{u}_n \tag{1}$$

$$\boldsymbol{w}_{n+1} = \boldsymbol{w}_n + \mu \boldsymbol{u}_n \underbrace{(d_n - y_n)}_{e_n}, \tag{2}$$

where μ is the step size and $.^T$ denotes transpose.

One of the variants of the LMS algorithm that does not update the filter coefficients after each output sample, but after a block of N_b samples, is known as Block LMS [5]. It has the advantage of

being computationally more efficient and allowing parallel implementations, at the price of a slightly higher error excess. The update equations of this algorithm are the following

$$\begin{aligned} \boldsymbol{y}_n =& \boldsymbol{\chi}_n \boldsymbol{w}_n \\ \boldsymbol{w}_{n+1} =& \boldsymbol{w}_n + \underbrace{\frac{2\mu'}{L}}_{\mu} \boldsymbol{\phi}_n, \end{aligned} \tag{3}$$

where $\boldsymbol{\chi}_n$ is an $N_b \times N_E$ matrix in which the ith row is the vector $\boldsymbol{u}_{n \cdot N_b+i}^T = [u_{nN_b+i},\, u_{nN_b+i-1},\, \cdots,\, u_{nN_b+i-N_E+1}]$, and $\boldsymbol{\phi}_n = \boldsymbol{\chi}_n^T \boldsymbol{e}_n$ is the vector representing the opposite of the scaled averaged estimate of the error gradient for the N_b samples of the nth block (the scale constant is already embedded into μ). Furthermore, for the same convergence speed, the BLMS algorithm presents, in some cases, better numerical accuracy than the standard LMS. A study on the numerical accuracy for the BLMS algorithm is undertaken in Section VII-B.

There are many other variants of the LMS algorithm, but we will constrain our analysis and designs to only these two forms. For more complex adaptive algorithms, the difficulties of a privacy-preserving implementation are essentially those derived from the cipher blowup problem and, additionally, those derived from the implementation of nonlinear functions. The latter is a problem that does not come specifically from the adaptive filtering scenario and, thus, falls out of the scope of this work. Hence, the chosen forms of LMS are representative enough, as they hold the essential characteristics of adaptive filtering, and at the same time they are practical developments widely used in a vast number of applications, as those sketched in Section III, in the context of a privacy-aware scenario.

III. Privacy Scenario and Trust Model

For all our protocols, we will consider two parties, $\mathcal{A}$ and $\mathcal{B}$, both using an additively homomorphic cryptosystem in an asymmetric scenario, where $\mathcal{B}$ can only encrypt, but $\mathcal{A}$ possesses also the decryption key, and can perform both encryption and decryption.

For the problem of private filtering, the studied scenario represents a problem of private data processing, in which one party possesses the input signal and other party possesses the reference signal or the system model for driving the filtering of the input signal.

Hence, we will assume that one party $\mathcal{B}$ has clear-text access to the to-be-filtered sequence u_n, while the other party $\mathcal{A}$ will provide the desired sequence d_n; both parties' inputs must be concealed from each other. The system parameters can be known by both parties or be provided by one party; in our case, we assume that the update step μ is agreed by both parties. The output of the algorithm (the filtered signal) is provided in encrypted form, in order to be input to a subsequent private protocol.

Regarding the privacy requirements, we will assume that both parties are semi-honest, in the sense that they will adhere to the established protocol, but they can be curious about the information they can get from the interaction. In this scenario, our protocols can be proven private (cf. Section VI-A);

informally, both parties $\mathcal{A}$ and $\mathcal{B}$ can only get the information given by the disclosed output of the system, and no information is leaked from the intermediate steps of the protocols.

Adaptive filtering has a considerable number of applications in the field of signal processing. They can be classified in four categories, namely identification, inverse modeling, prediction and interference cancellation. Within these categories, numerous applications are subject to privacy constraints and can benefit from the primitives that we present in this work. In the following paragraphs, as illustrative examples of the applicability of our secure protocols, we briefly introduce some of them, mainly related to *multiuser communications* where the privacy of the users must be protected from each other and, in the cases where it exists, from the central processing server. Further details of the application of our protocols to these scenarios can be found in [6]; we omit them here due to space constraints.

A. Private Adaptive Beamforming

Adaptive beamforming is a spatial application of adaptive filtering where a system composed of an array of antennas changes the directionality of the transmitted/received signal without mechanically moving the antennas. In the most common setting, the system must determine the spatial direction of the interfering signal and/or that of the target signal, and filter the sensed signals in order to cancel the former and extract the latter; it finds use in communications, radar, sonar or speech enhancement. The interfering signal comes usually from another source. The trust model in this scenario deals with, on the one hand, the protection of the transmitted/received target signal, and, on the other hand, the protection of the interfering signal and the spatial position of the interfering source. The two parties involved in the scenario are represented in the beamformer by the adaptive filtering mechanism that cleans the desired signal, and the model and pilot information for the desired signal. Again, this model fits perfectly in our framework, and the protocols that we present can be straightforwardly adapted to this scenario. The private filtering block (Figure 1) provides the adaptive weights applied to the received signals in order to adjust the directivity of the antenna array, without disclosing the contents of the interfering private signal; as in a private interference cancellation scenario, it must be complemented by another private block, denoted private beamforming block, that processes the mixed signals while concealing the private information.

As a specific example of this scenario, we could pose the problem of a cellular smart antenna, property of a mobile operator receiving signals (mixed into a signal $\boldsymbol{u}_n$) from his own users and also from users of a second operator that subcontracts the infrastructure of the former. The latter operator (party $\mathcal{A}$) has decryption capabilities (and reference signals $d_{n,i}$ for each of his users) and wants to perform adaptive beamforming to clean the signals $y_{n,i}$ from the clients without disclosing to the former (party $\mathcal{B}$) their positions or the contents of the cleaned signals, in such a way that the

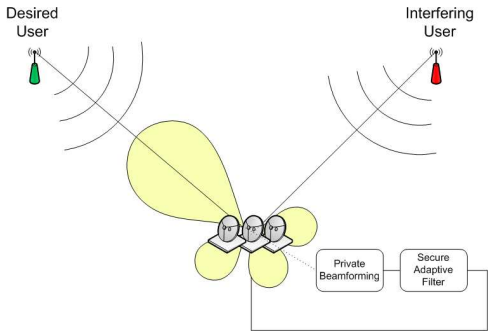


Fig. 1: Private Adaptive Beamforming Scenario.

information of the users of $\mathcal{B}$ is also not disclosed to $\mathcal{A}$.

B. Private Model-Reference Adaptive Control

There are many control applications [7] where the parameters of the controlled system are either not fully known or vary over time. Adaptive control yields a solution for maintaining consistent performance in these cases. It is used in many industrial contexts like, to name a few, robot manipulation, ship steering, aircraft control or metallurgical/chemical process control. Model-Reference Adaptive Control (MRAC) is one approach for constructing adaptive controllers. An MRAC system is composed of four elements:

- A *plant* with a known structure but unknown parameters.
- A *reference model* that specifies the desired output of the control system to the external command. It should match the performance specification while being achievable by the control system.
- A feedback control law (*controller*) with adjustable parameters. It should guarantee tracking convergence and stability.
- An *adaptation mechanism* for updating the adjustable parameters.

The trust model in this scenario can be devised as a two party model (involving privacy of system users at the plant and at the controller), where the plant outputs must be kept secret from the party that runs the controller, and the reference model that the controller applies must also be kept secret for the parties in the plant. In order to adaptively control the plant while keeping the privacy constraints, the same philosophy that we apply to LMS can be used to straightforwardly translate the protocols that we present for their use in this scenario.

As a specific example for this scenario, we could devise a spacecraft control system working with classified information coming from a vehicle in orbit, using an antenna under the control of a non-trusted party; the control information cannot be disclosed for keeping the management of the vehicle behavior secret. In this case, the party that emits the control (reference d_n) signal has decryption capabilities, while the non-trusted party that receives the vehicle's signals (u_n) can only encrypt.

Current privacy-preserving solutions cannot be directly applied to these scenarios due to the cipher blowup problem, that prevents the use of homomorphic computation alone. We will present in the subsequent sections our novel solutions to that problem; they have a direct application in the aforementioned scenarios and present efficient private protocols that overcome cipher blowup, finding an optimal trade-off between precision and complexity.

IV. Secure Computation

In this section, we review some of the concepts of secure computation that are needed for the development of our constructions, namely *Homomorphic Encryption*, *Secret Sharing* and *Secure Multiparty Computation*.

A. Homomorphic Encryption

Some cryptosystems present homomorphisms [8] between the groups of clear-text and cipher-text, that allow for the execution of a given operation directly on encrypted values, without the need of decryption. Examples of homomorphic cryptosystems are RSA, with a multiplicative homomorphism, or Paillier [9] and its variants, with an additive homomorphism.

In this work, we do not restrict the used cryptosystem for the presented protocols, as long as it presents an additive homomorphism. There are many semantically secure cryptosystems with this property, like Paillier [9] or DGK [10], but for the sake of clarification, and for performing the numerical calculations of Section VIII-A, we have chosen the extension of Paillier encryption given by Damgård and Jurik [11], due to its good trade-off between efficiency, encryption size and cipher expansion.

Damgård and Jurik's cryptosystem presents an additive homomorphism that allows computing the addition of two encrypted numbers and the product of an encrypted number and a known integer:

$$[\![x+y]\!] = E_{DJ}[x+y] = E_{DJ}[x] \cdot E_{DJ}[y] \mod n^{s+1}, \quad [\![x \cdot k]\!] = E_{DJ}[x \cdot k] = E_{DJ}[x]^k \mod n^{s+1}.$$

The message space is $\mathbb{Z}_{n^s}$, where n is the product of two safe primes p, q, and the parameter $s \in \mathbb{Z}^+$ is fixed.

The encryption of a message x is obtained by picking a random $r \in \mathbb{Z}^*_{n^{s+1}}$ and computing the ciphertext $E_{DJ}[x]$ as $E_{DJ}[x] = g^x r^{n^s} \mod n^{s+1}$.

We must also draw attention to the fact that currently there is no practical fully homomorphic cryptosystem; i.e., there is no secure cryptosystem that allows for the homomorphic computation of additions and products without restrictions. There have been recent contributions by Gentry [3], that presents a cryptosystem based on ideal lattices with bootstrappable decryption, and shows that it achieves a full homomorphism. Nevertheless, the author argues that making the scheme practical

remains an open problem. There is a research line currently underway, with works like [12], focused on translating Gentry's scheme into a practical fully homomorphic solution, but it is still limited to very small plaintexts and very simple circuits. By now, we will adhere to using an additively homomorphic cryptosystem, always taking into account the advantages that an efficient and practical fully homomorphic cryptosystem would provide.

B. Secret Sharing

Secret sharing is a technique introduced by Adi Shamir [13], by which a given value (the secret) is divided among several parties, such that the cooperation among a number of these parties is needed to recover the secret. None of the parties alone can have access to the secret.

Shamir's scheme is based on polynomials, and the need of k points to completely determine a degree $(k-1)$ polynomial. Secret sharing is a widely used primitive in cryptographic protocols. In this work we focus on two-party protocols; thus, we are only interested in the two-party version of the secret sharing scheme, that is based on linear functions and, consequently, naturally supports the computation of sums and products directly on the shares: let $\mathbb{Z}_n$ be the domain of the secrets. Then, a share of a secret x is defined as two values x_A and x_B, owned by their respective parties, such that $x_A + x_B \equiv x \mod n$. Hereinafter, randomizing an encrypted value x will mean obtaining one share and providing the encryption of the other (through homomorphic addition).

C. Secure Multiparty Computation

Secure Two-Party Computation was born in 1982 with Yao's Millionaires' problem [1], and later generalized to Multiparty Computation by Goldreich *et al* [2]. Yao proposed a solution to the binary comparison of two quantities in possession of their respective owners, who are not willing to disclose to the other party the exact quantity they own. The solution that Yao proposed was based on *garbled circuits*, in which both parties evaluate a given circuit, gate by gate, without knowing the output of each gate. Yao's solution was not efficient, and later, many protocols based on other principles like homomorphic computation or secret sharing were proposed in order to efficiently perform other operations in a secure manner.

Nevertheless, while homomorphic computation and secret sharing are very efficient for implementing arithmetic operations, circuit evaluation is still more efficient when dealing with binary tests [14]. Thus, there exist efficient protocols for binary comparison [14], [15] or Prefix-OR [14] that will be needed, with some modifications, for the implementation of our solutions. Traditionally, the search for efficient solutions has led to proposals for changing between integer and binary representations in order to efficiently implement both arithmetic and binary operations; e.g., there are solutions like

the BITREP protocol [16], that converts Paillier encrypted integers to Paillier encryptions of their corresponding bit representation.

For the garbled circuit constructions, we use the efficient protocols developed in [17], and for the transformation from Paillier representation to a binary representation suitable for usage in a garbled circuit, we employ the protocols in [18].

V. Related Art

Previous work on private linear filtering has been presented as part of the SPEED project [19], dealing with the privacy problem in a two-party setting where one party has an input to a linear filter and another party holds the filter coefficients. Such efficient privacy-preserving solutions are based solely on homomorphic processing, as it fits perfectly the linear filtering operation without imposing any overhead on communication. Within the area of linear filtering, we can point out the works by Bianchi *et al.* [20]–[22], dealing with encrypted DFT and DCT transforms and frequency-domain linear filtering. Additionally, these works discuss also the problem of disclosing data derived from the inputs without any dimensionality reduction, as the original data can be inferred from the disclosed outputs.

There have been also some contributions for more complex operations, involving the combination of garbled circuits and homomorphic processing, most notably those from Kolesnikov *et al.* [18], in which homomorphic processing is used for the linear operations, while garbled circuits deal with non-linear operations.

Regarding the privacy considerations in iterative algorithms, there are some contributions in the area of private collaborative filtering, like those by Canny [23] and Erkin [24]. In the former, Canny developed a privacy-preserving iterative conjugate gradient algorithm for the calculation of the SVD decomposition of a shared preference matrix $\boldsymbol{P}$. The setting in [23] is different from ours in several aspects: a) It involves multiple parties, and the gradient estimate in each iteration is calculated as the sum of the contributions from each of these parties; b) the result of every iteration is decrypted and disclosed before the next iteration; hence, it does not involve successive products of encrypted values, as each party uses only clear-text values for producing the results at every iteration; c) as a drawback, the disclosure of the approximation of the preference matrix and the global gradient calculated at each iteration are publicly known; hence, the security relies on those matrices having a very high dimension and the system having a very high number of users. In the present work, we are dealing with protecting the signals coming from one party during their adaptive filtering by another untrusted party; in this setting, Canny's solution loses its privacy properties, as the value disclosed after each iteration allows each party to calculate the secret input from the other party. Furthermore, we must keep all the intermediate values encrypted in order to effectively preserve the privacy of the involved

users, and this involves repeated products of encrypted numbers that will have direct consequences on the viability of the used privacy-preserving techniques due to the cipher blowup problem.

Other private iterative algorithms involve K-means clustering of a database shared between two parties, like the one proposed by Jagannathan *et al.* [25]; again, in this setting, the results of each iteration (the current classification of the elements) are disclosed before the next, and the security relies on the dimensionality of the databases, unlike the case of private adaptive filtering.

Hence, to the best of our knowledge, there are no specific solutions within the emerging field of Signal Processing in the Encrypted Domain for securely executing iterative or adaptive algorithms besides [26], nor any study performed on the impact that an iterative implementation has on the range of representable numbers when the results of each iteration cannot be disclosed. Thus, our solutions are presented here as the first ones dealing with privacy preserving adaptive filtering algorithms.

VI. Proposed Protocols

In this section, we present different approaches in order to tackle the private implementation of the LMS algorithm, and to overcome the limitations that the sole application of current homomorphic encryption and garbled circuits has in our scenario.

A. Homomorphic processing

The LMS algorithm, and most of the adaptive filters currently in use, while having an essentially non-linear behavior due to their adaptive nature, comprise only linear operations. Thus, it is foreseeable that homomorphic processing can yield a quite efficient solution. Unfortunately, there are two drawbacks in following this approach:

- There are no practical fully homomorphic cryptosystems; the most promising contribution in this sense is Gentry's poly-time and poly-space fully homomorphic cryptosystem, whose constant factors make it impractical [3]; hence, using only homomorphic processing implies resorting to interactive protocols for performing multiplications between encrypted values, or for any other more complex operation.
- The inputs to the secure protocol must be quantized prior to encryption. Hence, it is necessary to work in fixed point arithmetic, keeping a scale factor that affects all the values under encryption. This factor will increase with each encrypted multiplication, limiting the number of allowed iterations of the adaptive algorithm, until the encrypted numbers cannot fit the cipher, when it is said that the cipher blows up.

There are two approaches for devising a private LMS protocol, depending on whether the output is either disclosed or given in encrypted form. The simplest approach is the one in which the output of the LMS algorithm can be disclosed to both parties; in this case, a secure protocol could be quite efficient,

as the problem of the increased scale factor can be easily solved by requantizing the outputs in the clear after every iteration with no additional overhead, requiring only homomorphic additions and multiplications and interactive multiplication gates. Nevertheless, besides its simplicity, this scenario is of no interest due to the fact that disclosing the output gives both parties all the necessary information for retrieving the other party's private input and rendering the privacy-preserving solution unnecessary and unusable.

The private output scenario is more realistic, and it is the one on which we will focus, as it corresponds to the case where the LMS block can be used as a module of a more complex system whose intermediate signals must not be disclosed to any party. We will adhere from now on to this scenario, and we will begin by presenting a protocol that uses only homomorphic computations (Algorithm 1), in order to have a complexity reference and show its limitations. In Algorithm 1, interactive multiplication protocols are avoided due to the division of the roles of both parties: the party that provides the private input $\boldsymbol{u}$, without decryption capabilities, is the one that will perform the homomorphic operations between the encrypted intermediate values and $\boldsymbol{u}$. In this case, there is a constant scaling factor (`updateFactor`) that is accumulated after every iteration, and that forces to scale the inputs and the intermediate results in order to have the correct output. This accumulated factor limits the maximum number of iterations that the protocol will be able to execute before the cipher blows up:

$$N_{\text{max iter}} = \left\lfloor \frac{n_{\text{cipher}}}{n_x + \log_2(\texttt{updateFactor})} \right\rfloor,$$

where n_x bits are used for representing each input, and n_{cipher} is the bit size of the maximum representable number inside the cipher.

The communication complexity of this protocol, assuming Damgård-Jurik encryptions, is

$$\mathrm{C}_{HPcm} = (2N_{\text{iter}} + N_E - 1)|E|,$$

where N_{iter} is the number of performed iterations, N_E is the length of the filter and $|E|$ represents the number of bits of an encryption.

It is important to note that the iteration limit imposed by this protocol, due to cipher blowup, is a serious drawback and impedes the use of only homomorphic processing (in its current development stage) to perform adaptive filtering. For typical values of the used precision (48-bit numbers, 24 bits for the fractional part) and medium-term security (2048 bits for Paillier modulus), this protocol is limited to approximately 17 iterations, what is insufficient even for reaching the steady-state regime, and prevents its use in any practical application. Therefore, we present it as a reference that sets the minimum of computation and communication complexity that can be achieved for a private LMS. It must be noted that this iteration limit could be improved through the use of a different encoding of the inputs, like the logarithmic encoding presented in [27], but such approach comes at the price of an increased communication and computation complexity even for additions and multiplications.

In the following subsections, we propose several novel alternatives and extensions, through the combination of other privacy-preserving techniques, aimed at overcoming the cipher blowup problem with the minimum overhead in communication and computation complexity, while preserving an acceptable excess error with respect to the infinite precision non-private LMS algorithm.

Algorithm 1 Homomorphic Processing (HP) PrivateLMS Protocol

Inputs: $\mathcal{A}$: $d_n, \boldsymbol{w}_0$; $\mathcal{B}$: $u_n, \boldsymbol{w}_0$

Outputs: $[\![y_n]\!]$.

$\mathcal{A}$	$\mathcal{B}$
Initialize `carriedFactor`$= 2^{n_f}$, `updateFactor`$= 2^{3n_f}$.	
Encrypt inputs and send $[\![d_n]\!]$ to $\mathcal{B}$.	
for $k = 1$ **to** N_{iter}	
	Perform the vector multiplication $[\![y_k]\!] = [\![\boldsymbol{w}_k]\!] \cdot \boldsymbol{u}_k$.
	Scale $[\![d'_k]\!] = [\![d_k]\!]\cdot$`carriedFactor`.
	Obtain $[\![e'_k]\!] = \mu \cdot ([\![d'_k]\!] - [\![y_k]\!])$.
	Perform the scalar multiplication $[\![\boldsymbol{\Delta w}_k]\!] = [\![e'_k]\!] \cdot \boldsymbol{u}_k$.
	Update the coefficients vector $[\![\boldsymbol{w}_{k+1}]\!] = [\![\boldsymbol{w}_k]\!]\cdot$`updateFactor`$+[\![\boldsymbol{\Delta w}_k]\!]$.
Update `carriedFactor`$=$`carriedFactor`$\cdot$`updateFactor`.	
	Output $[\![y_k]\!]$.
endfor	

Security: Regarding the security of this protocol and the ones presented in the following sections, it can be proven, using a simulator argument, that all of them are statistically secure under the random oracle model, assuming semi-honest parties: due to the use of sequentially composed secure subblocks and the semantic security of the underlying cryptosystems, the views that each party gets are statistically indistinguishable from a random sequence, and the parties cannot derive from those views any extra information about the private inputs of the other party. We will not go into details about these proofs, as they are rather straightforward.

B. Garbled Circuits Implementation

This protocol represents the whole LMS algorithm as a binary circuit, in which we include a rounding operation in each multiplication circuit in order to preserve a constant bit-size for the handled numbers. The protocol is sketched as Algorithm 2. It is straightforward to derive the binary circuit implementing Eqs. (1) and (2), so we do not detail its construction in Algorithm 2; as for the garbled implementation, we use the XOR-free version of [17], with the efficient extensions for the Oblivious Transfer (OT) protocol of [28], and an Elliptic Curve version of ElGamal [29], [30] for the encryptions. This implementation uses fixed precision, and rounds the numbers after every

multiplication in order to preserve this precision. Hence, it overcomes the quantization problems that the previous one presents, but it requires working at a bit level, thus being its performance highly dependent on the bit-size of the represented numbers.

Additionally, every transferred bit must be independently encrypted, which also multiplies the communication complexity of the whole protocol by a large factor, resulting in

$$\begin{aligned} \mathrm{C}_{GCcm} =& |E| \left(4n_x^2(N_{\text{iter}} + 2N_E N_{\text{iter}}) + 2n_x(-1 + 10N_{\text{iter}} + 4N_{\text{iter}}n_f + 2N_E(1 + 5N_{\text{iter}} + 4N_{\text{iter}}n_f)) \right. \\ & \left. -4N_{\text{iter}}(5 + n_f(3 + n_f) + N_E(7 + 2n_f(3 + n_f)))\right), \end{aligned}$$

where N_E is the length of the filter, $|E|$ represents the number of bits of an EC-ElGamal encryption, n_x is the total number of bits for representing each number, and n_f is the number of bits used for the fractional part.

The complexity has, as expected, a linear dependence on the product of the number of iterations and the size of the filter, while it has a quadratic dependence on the bit-size of the used numbers and the bit-size of the fractional part, due to the presence of multiplication circuits. The communication complexity is much higher than in Algorithm 1, due to the need of communicating the whole garbled circuit prior to its execution.

A remark worth noting on Algorithm 2 is that inputs get to the circuit once per iteration, even when they could be joined all together (in long enough blocks) and apply OT reduction techniques [28] for lowering the computational complexity of the whole protocol. The reason behind this structure is that we are assuming that the system must work with some real time constraints, and offer the outputs at the same rate as the input, without a significant delay. Hence, the inputs might be packed together for reducing the computation overhead of the OTs in small blocks, whenever the delay is affordable; it must be noted that the communication overhead is not reduced though: the reduction techniques in [28] replace public key encryptions with computationally lighter hash functions; since we are using elliptic curves for the public key encryption, their size is comparable to that of a collision resistant hash for the same security level. The effect of the OT reductions is shown for the hybrid block protocol in Section VIII-A.

C. Hybrid Implementation

In order to overcome the quantization problem in Algorithm 1 and lower the communication complexity of Algorithm 2, we have developed a hybrid algorithm (Algorithm 3) that uses homomorphic processing for the bulk of the algorithm, and a quantization circuit to avoid carrying factors. Conversion protocols from homomorphic encryption to binary representation and vice-versa are used to connect both parts of the protocol.

There are several possible combinations of homomorphic processing and garbled circuits that yield different results in the complexity balance. We can argue that the optimal point for applying

Algorithm 2 Garbled Circuit (GC) PrivateLMS Protocol

Inputs: $\mathcal{A}$: $d_n, \boldsymbol{w}_0$; $\mathcal{B}$: $u_n, \boldsymbol{w}_0$.

Outputs: $[\![y_n]\!]_b$.

$\mathcal{A}$	$\mathcal{B}$
Obtain the bit representation of their respective inputs.	
Execute `generateGC`() for the first $m \leq N_{\text{iter}}$ iterations, and send the garbled circuit and the keys corresponding to her inputs to $\mathcal{B}$; the garbled gates for the remaining iterations can be generated and sent in parallel with the execution of the previous ones.	
for $k = 1$ **to** N_{iter}	
Perform parallel OT protocols so that $\mathcal{B}$ get the input keys to initialize the circuit corresponding to the k^{th} iteration.	
	Execute the circuit, using the received input keys from $\mathcal{A}$. Output $[\![y_k]\!]_b$.
endfor	

quantization in terms of efficiency is at every iteration, when the scaled output of the filter y'_k is obtained (cf. Algorithm 3), using a quantization step of 2^{3n_f} to recover the initial precision of n_f fractional bits. When this strategy is chosen, only one scalar value is input to the quantization circuit at every iteration, which means reaching the minimum of communication complexity for the used garbled circuit. Furthermore, this quantization allows to keep a constant scaling factor for the rest of the handled values, avoiding the rescaling operation that is performed in Algorithm 1 for every input value and for the filter coefficients; hence, the computation complexity also reaches its minimum with this strategy. Lastly, the bounded size of the represented values makes possible the use of a packing strategy for the homomorphic processing, such that more than one input value can be packed into the same encryption. This will be further commented in Section VI-D.

The communication complexity of the protocol is

$$\mathrm{C}_{Hycm} = (2N_{\text{iter}} + N_E - 1)|E_H| + N_{\text{iter}}|E_C|(19n_x + 7n_{\text{sec}} + 24n_f),$$

where N_E is the length of the filter, $|E_H|$ and $|E_C|$ represent the bit-size of a homomorphic and a garbled encryption respectively, n_x is the total number of bits for representing each number, n_f is the number of bits used for the fractional part, and n_{sec} is the number of security bits used for the conversion protocols. As the circuit part involves only rounding operations, and the multiplications are performed homomorphically, the complexity is linear on the bit-length of the inputs and the number of iterations, instead of quadratic, as in the garbled-circuit solution.

In this case, the quantization step used for the filter coefficients is not the same as the one used for the input/output values: filter coefficients are quantized with a finer step, using $3 \cdot n_f$ bits for their fractional part, instead of n_f. This is needed in order to keep the bit-size of the outputs constant and

avoid any further quantization operations. Furthermore, as stated in Section VII-B, the quantization step of the filter coefficients is the one that has the highest impact on the quantization error that is propagated to the outputs, so this measure will make this method have a much better behavior than Algorithm 2 in terms of mean square error (MSE).

Algorithm 3 Hybrid (Hy) PrivateLMS Protocol

Inputs: $\mathcal{A}$: $d_n, \boldsymbol{w}_0$; $\mathcal{B}$: $u_n, \boldsymbol{w}_0$.

Outputs: $[\![y_n]\!]$.

$\mathcal{A}$	$\mathcal{B}$
Encrypt her inputs. Execute `generateGC`() for the rescaling circuit in each of the first $m \leq N_{\text{iter}}$ iterations, and send the garbled circuit to $\mathcal{B}$; the remaining circuits can be generated and sent during the execution of the previous ones.	
for $k = 1$ **to** N_{iter}	
	Perform the vector multiplication $[\![y'_k]\!] = [\![\boldsymbol{w}_k]\!] \cdot \boldsymbol{u}_k$.
Convert $[\![y'_k]\!]$ to its bit-representation using the bit conversion protocol. Perform parallel OT protocols so that $\mathcal{B}$ get the input keys to initialize the circuit corresponding to the k^{th} iteration.	
	Execute the rescaling circuit $[\![y_k]\!]_b = \left[\!\left[\left\lceil \frac{y'_k}{2^{3 \cdot n_f}} \right\rfloor \right]\!\right]_b$, using the received input keys from $\mathcal{A}$.
The shared output of the circuit $[\![y_k]\!]_b$ is converted back to a homomorphic encryption $[\![y_k]\!]$.	
	Obtain $[\![e'_k]\!] = \mu \cdot ([\![d_k]\!] - [\![y_k]\!])$. Perform the scalar multiplication $[\![\boldsymbol{\Delta w}_k]\!] = [\![e'_k]\!] \cdot \boldsymbol{u}_k$. Update the coefficients vector $[\![\boldsymbol{w}_{k+1}]\!] = [\![\boldsymbol{w}_k]\!] + [\![\boldsymbol{\Delta w}_k]\!]$. Output $[\![y_k]\!]$.
endfor	

D. Hybrid Block Protocol and Packing Strategy

As pointed out in the preceding section, the hybrid implementation of the algorithm has the advantage of working always with bounded numbers, and it allows for a parallel block implementation in the form of packed coefficients within a cipher, as introduced in [31].

Typically, the numbers involved in signal processing calculations can be bounded, and their bit-size represents just a very small fraction of the size of a secure cipher modulus; the extra bit size is unused, but it is necessary due to security constraints on the cryptosystem. Nevertheless, this space can be utilized; assuming that every involved calculation result x is bounded at the moment of unpacking such that it occupies at most n_b bits (i.e., $|x| \leq 2^{n_b - 1}$; for the hybrid protocol, $n_b = n_x + 3 \cdot n_f$), every K inputs $\{x_i\}_{i=0}^{K-1}$, with $K \leq \lfloor \frac{n_{\text{cipher}} - n_{\text{sec}}}{n_b} \rfloor$ (being n_{sec} the number of security bits needed for the conversion protocol), can be packed in only one encryption as $[\![\boldsymbol{x}_{\text{packed}}]\!] = [\![\sum_{m=0}^{K-1} (x_m + 2^{n_b - 1}) \cdot$

$2^{m \cdot n_b}]\!]$, being 2^{n_b-1} a shift factor for considering only positive numbers[1]. This packing allows for the computation of vector products and additions with a reduced complexity (it gets divided by the number of packed elements), taking advantage of the unused huge space that the cipher allows.

This strategy was later generalized to an arbitrary base in [22], but due to the use of binary circuits, 2^{n_b} is the most efficient choice, as divisions and multiplications by this factor in the circuit are just implemented for free as bit-shifts in the clear.

While the packing operation improves the efficiency of the homomorphic computations, on the garbled circuit side of the protocol, it has the effect of increasing the size of the used circuits, multiplying it by the number of values packed into the same encryption. Thus, the complexity of the executed garbled circuits is preserved after packing (lowered if OT reduction techniques are used for each packed block), while the conversion protocols also get an increase in performance, as only one conversion is needed for each encryption containing several packed numbers.

Turning to the secure hybrid block protocol, the packed elements must be processed all together, applying the same coefficients to all of them. Hence, the normal LMS algorithm cannot take advantage of packing, as the filter is kept constant for each group of packed samples, and the update equation has to be slightly modified in order to account for the average error of the whole set of packed samples instead of the error of individual samples; this filter is known as the Block LMS algorithm [5], in which the update equation is

$$\boldsymbol{w}_{n+1} = \boldsymbol{w}_n + \mu \sum_{i=0}^{N_b-1} \boldsymbol{u}_{n \cdot N_b+i} \cdot e_{n \cdot N_b+i},$$

where N_b represents the size of the block. The usual choice of N_b for the Block LMS filter is $N_b = N_E$, as it yields the minimum computational complexity.

Since the packing factors 2^{n_b} are chosen to be powers of two, the bit-conversion protocol automatically unpacks the numbers without any extra complexity, and the conversion to homomorphic encryption after the circuit evaluation is performed for each unpacked number in parallel.

The communication complexity of the hybrid block protocol, taking into account that the XOR gates are free of communication for the used implementation, is exactly the same as for the hybrid protocol:

$$\mathrm{C}_{HBcm} = (N_E - 1 + 3N_{\text{iter}} + 5N_E N_{\text{iter}})|E_H| + N_{\text{iter}}|E_C|(19n_x + 7n_{\text{sec}} + 24n_f).$$

This complexity is linear in the number of iterations, the size of the filter and the bit size of the numbers, and it is independent of the number of packed coefficients.

[1]The shift factor fixes the sign convention between the bit representation ($-a \equiv 2^{n_b} - a$) and the modular arithmetic ($-a \equiv n - a$), working always in the range $[0, 2^{n_b})$, and avoiding errors in the conversion between both representations. Hence, it is not an integral part of the packed formulation, and shall only be applied before a conversion protocol.

Algorithm 4 Hybrid Block (HB) PrivateLMS Protocol

Inputs: $\mathcal{A}$: $d_n, \boldsymbol{w}_0$; $\mathcal{B}$: u_n,$\boldsymbol{w}_0$.

Outputs: $[\![y_n]\!]$.

$\mathcal{A}$	$\mathcal{B}$
Encrypt her inputs. $\mathcal{A}$ executes `generateGC`() for the unpacking, parallel rescaling and output circuits in each of the first $m \leq N_{\text{iter}}$ iterations, and sends these garbled circuits to $\mathcal{B}$; the circuits for the remaining iterations can be generated and sent during the execution of the previous ones.	Pack the input vector as $X_j^{(k)} = \sum_{i=0}^{N_b-1} 2^{n_x+3n_f} \cdot u_{k\cdot N_b+i-j}, j = \{0,\ldots,N_E-1\}$.
for $k = 1$ **to** $\lceil N_{\text{iter}}/N_b \rceil$	
	Perform the packed vector multiplication $[\![\boldsymbol{y}_k]\!] = [\![\boldsymbol{w}_k]\!] \cdot \boldsymbol{X}^{(k)}$.
Convert $[\![\boldsymbol{y}_k]\!]$ to its unpacked bit-representation using the bit conversion protocol. Perform parallel OT protocols so that $\mathcal{B}$ get the input keys to initialize the circuit corresponding to the k^{th} iteration.	
	Execute the unpacking and parallel rescaling circuit, using the received input keys from $\mathcal{A}$.
The output of the circuit $[\![y_{k\cdot N_b+i}]\!]_b, i = \{0,\ldots,N_b-1\}$ is converted back to a homomorphic encryption $[\![y_{k\cdot N_b+i}]\!], i = \{0,\ldots,N_b-1\}$.	
	Obtain $[\![e'_{k\cdot N_b+i}]\!] = \mu \cdot ([\![d_{k\cdot N_b+i}]\!] - [\![y_{k\cdot N_b+i}]\!]), i = \{0,\ldots,N_b-1\}$. Perform the scalar multiplication $[\![\boldsymbol{\Delta w}_k]\!] = \sum_{i=k\cdot N_b}^{(k+1)\cdot N_b-1} [\![e_i]\!] \cdot \boldsymbol{u}_{i-N_E+1}$. Update the coefficients vector $[\![\boldsymbol{w}_{k+1}]\!] = [\![\boldsymbol{w}_k]\!] + [\![\boldsymbol{\Delta w}_k]\!]$. Output $[\![y_{k\cdot N_b+i}]\!], i = \{0,\ldots,N_b-1\}$.
endfor	

E. Fast Implementation

The hybrid block protocol is far more efficient than the one based solely on garbled circuits. Nevertheless, the conversion protocols introduce an overhead, and the fact that the input values to the rounding garbled circuits are generated on the fly prevents much of the preprocessing that garbled circuits would need to compensate the complexity of the oblivious transfers. The gap in computational complexity with respect to the solution based on homomorphic processing is too big (cf. Section VIII-A), especially when using a high precision bit representation. Thus, we have come to a much more efficient solution that, in order to tighten that gap, avoids the use of circuits, and substitutes them by an approximate rounding protocol with statistical security. The block implementation can also profit from the use of this solution, with a decrease on the maximum packing efficiency, as now the number of packed coefficients is bounded by $N_b^{(FB)} \leq \lfloor \frac{n_{\text{cipher}}}{n_b+n_{\text{sec}}} \rfloor$, instead of $N_b^{(HB)} \leq \lfloor \frac{n_{\text{cipher}}-n_{\text{sec}}}{n_b} \rfloor$, where $n_b = n_x + 3n_f$ is the maximum number of bits that a coefficient can occupy, and n_{sec}

is the number of security bits required for the protocol. In this case, the approximate rounding protocol also performs the unpacking of the results; it is described in its complete form in the next subsection. The implementation of this fast protocol replicates exactly the implementation of the hybrid protocols, without the generation and use of the garbled circuits, substituted by the much more efficient approximate rounding protocol; thus, for the sake of brevity, we omit its sketch. The disadvantage is that the rounding error rises with this protocol; however, it is compensated by a reduction of the complexity gap with respect to the solely homomorphic solution.

The communication complexity of the fast implementation, in normal and block forms respectively, is

$$\mathrm{C}_{FPcm} = (4N_{\text{iter}} + N_E - 1)|E_H|, \quad \mathrm{C}_{FBcm} = \left(\left(3 + \frac{1}{N_b}\right) N_{\text{iter}} + N_E - 1\right)|E_H|,$$

where N_b is the number of packed coefficients for the block protocol. This complexity is of the same order as that of the protocol that uses only homomorphic processing.

1) Approximate Rounding and Unpacking protocol: We have developed several protocols for quantization under encryption. In Appendix B, we present two versions of them, with unconditional blinding of the used values; one is an exact protocol that produces the same results as the clear-text quantization, and the other is an approximate faster version; both use comparison circuits for performing the quantization operation. We sketch at Algorithm 5 a third version of the secure quantization protocol where a statistical blinding is used instead of an unconditional one, avoiding the need for comparison circuits. The security of the algorithm is controlled by the parameter n_{sec}, chosen such that $2^{-n_{sec}}$ is negligible; then, the distribution of the blinded values is indistinguishable from a random sequence (a distinguisher will succeed with probability $2^{-n_{sec}}$); hence, due to the sequential composition of statistically secure protocols and the semantic security of the encryption system, the protocol can be proven statistically secure under the random oracle model using a simulator argument.

It can be seen that the rounding error that it introduces is higher than that of a linear quantizer, and it is not uniform between $[-\frac{1}{2}, \frac{1}{2})$, but triangular between $[-1, 1)$, thus duplicating the quantization MSE.

The communication complexity of the protocol is

$$\mathrm{C}_{RPcm} = (N_b + 1)|E_H|,$$

where N_b is the number of packed elements in one cipher, and $|E_H|$ is the bit size of a homomorphic encryption. Due to the great benefit in efficiency with respect to the impact on accuracy (cf. Section VII-B), this is the chosen protocol for the fast implementation of the private LMS algorithm.

We must point out that this solution to the cipher blowup problem represents the minimum increase in computation and communication complexity with respect to plain homomorphic processing. We have discarded the possibility of using a different number encoding due to the following reasoning:

our approximate rounding protocol is approximately equivalent to a secure multiplication protocol in terms of bandwidth and total computation (at most, one per iteration in the implementation of the whole LMS); using a different encoding like the one in [27], would introduce the overhead of working with triplets of encryptions for each number, adding two multiplication protocols per encrypted multiplication, and twelve multiplication protocols and two comparison protocols per encrypted addition; hence, our solution is notably more efficient.

Algorithm 5 Approximate Rounding and unpacking Protocol

Inputs: $\mathcal{A}$: Quantization step $\Delta = 2^l$ and a security parameter n_{sec};
$\mathcal{B}$: $[\![x_{\text{pack}}]\!] = [\![\sum_{i=0}^{N_b-1} x_i \cdot 2^{i\cdot(n_b+n_{sec}+1)}]\!]$, $\Delta = 2^l$, n_{sec}
Outputs: $\{[\![Q'_\Delta(x_i)]\!]\}_{i=0}^{N_b-1}$.

$\mathcal{A}$	$\mathcal{B}$
	Generate $x_i^{(b)} \in_R \{2^{nb-1}, \ldots, 2^{n_b-1} + 2^{n_b+n_{sec}}\}, i = \{0, \ldots, N_b-1\}$, with which he shifts and additively blinds the packed encryptions: $[\![x_p^{(a)}]\!] = [\![x_{\text{pack}}]\!] + [\![\sum_{i=0}^{N_b-1} x_i^{(b)} \cdot 2^{i\cdot(n_b+n_{sec}+1)}]\!]$, homomorphically. Send $[\![x_p^{(a)}]\!]$ to $\mathcal{A}$.
Decrypt and unpack the received encryptions, obtaining $\{x_i^{(a)}\}_{i=0}^{N_b-1}$.	
Apply a linear quantizer with step $\Delta = 2^l$ to their clear-text vectors component-wise, obtaining $\{Q_\Delta(x_i^{(a)})\}_{i=0}^{N_b-1}$ and $\{Q_\Delta(x_i^{(b)})\}_{i=0}^{N_b-1}$, respectively.	
Encrypt her quantized vector component-wise, and send the encryptions back to $\mathcal{B}$.	
	Unblind the quantized encrypted values obtained from $\mathcal{A}$, obtaining the encrypted quantizations of the original values $\{[\![Q'_\Delta(x_i)]\!]\}_{i=0}^{N_b-1} = \{[\![Q_\Delta(x_i^{(a)})]\!] - Q_\Delta(x_i^{(b)})\}_{i=0}^{N_b-1}$.

VII. Evaluation

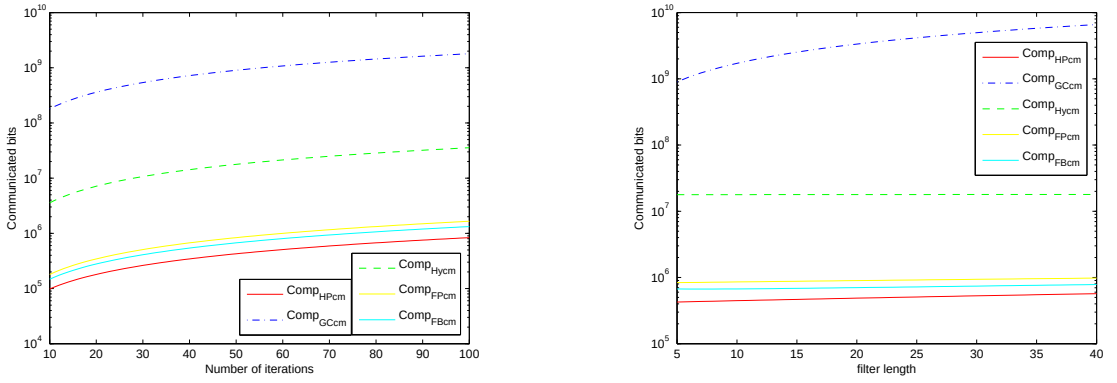
In this section, we perform a comparison of the developed protocols in terms of bandwidth, computational complexity and finite precision effects, providing also an evaluation of the chosen techniques for each of the solutions, and their suitability for the application scenarios. In the next section we also introduce a practical implementation of our protocols, that we have used for measuring actual execution times on real machines.

A. Bandwidth

In terms of communication complexity, the estimated transferred bits for each of the protocols have been given together with their description in the previous section. All the protocols have a communication complexity linear in the number of iterations, the size of the filter and the size of

the encryptions; nevertheless, the constants are not the same and the difference is perceptible and significant for normal values of the LMS parameters. As an exemplifying case, Figures 2a and 2b show the number of communicated bits for each of the protocols for a varying number of iterations and filter length respectively; the length of the encryptions is chosen for mid-term security (2048 bits for Damgård-Jurik modulus, 224 bits for the elliptic curve modulus, and 80 bits for the statistical security parameter used in the conversion protocols).

The obtained results using 32-bit numbers with 16-bit fractional precision are shown for a 5 tap filter in Figure 2a and for 50 iterations in Figure 2b. It can be seen that the bandwidth of the garbled circuit solutions–only garbled circuits (GC) and hybrid protocol (Hy)–is several orders of magnitude higher than that of the solutions including only homomorphic processing (HP). While the HP protocol needs to transfer two encryptions per iteration (8192 bits), the GC protocol communicates around 165 Mb per iteration for the chosen parameters. Hence, the communication complexity for the HP protocol and the fast protocols (FP and block FB) is higher than that of the clear-text protocols, but still practical; on the other hand, the bandwidth needed by the solutions that include garbled circuits make them almost totally infeasible for practical purposes, even when using small encryptions based on Elliptic Curves. The hybrid protocol presents, though, an intermediate complexity, due to the overhead, w.r.t. the HP solution, imposed by the use of conversion subprotocols for changing between bit-representation and homomorphic encryptions. This overhead will be translated in a decrease in computation load for the hybrid block protocol (cf. Section VIII-A).



(a) (b)

Fig. 2: Communication complexity as a function of the number of executed iterations with $N_E = 5$ (a) and the filter length with 50 iterations (b) for $|E_H| = 4096$, $|E_C| = 224$, $n_{\text{sec}} = 80$, $N_b = \min\left(N_E, \lfloor \frac{n_{\text{cipher}}}{n_x + 3 \cdot n_f + n_{\text{sec}}} \rfloor\right)$, $n_x = 32$, $n_f = 16$.

B. Error Analysis and Finite Precision Effects

One of the limitations of the presented protocols, inherent to privacy-preserving techniques that deal with encryption based on finite-fields, is the need of using fixed point arithmetic. This is actually not a severe issue, as current implementations of the traditional insecure algorithms also work with finite precision, but the flexibility of floating point yields a much wider range of representable values, and greatly improves on the quantization error propagated to the outputs of the algorithm. Numerical stability and numerical accuracy of the filters, that determine the resilience to quantization errors, come into play when dealing with fixed-point arithmetic.

While this issue is commonly avoided or mitigated by the use of a sufficiently large plaintext size to accommodate the needed precision, we believe that it is necessary to devote some space to calculating which is the needed precision and plaintext size for keeping the output Mean Square Error (MSE) within a given bound. In this section we review the error analysis of adaptive algorithms working with fixed-point arithmetic and apply it to the specific cases that our protocols involve. We assume that the inputs and outputs are quantized with n_f bits for their fractional part (of the total n_x bits used for coding), and the filter coefficients and some intermediate results are quantized with n_{wf} bits and n_{If} bits for their fractional part respectively. The use of a different quantization level for vector coefficients is explained in Section VI.

Neglecting the overflow effects and assuming stationary d_n and u_n with variances σ_d^2 and σ_u^2, i.i.d.[2] u_n, and uniform and independent quantization errors of the inputs (with variance $\sigma^2 = \frac{2^{-2n_f}}{12}$) and intermediate values (with variance $\sigma_I^2 = \frac{2^{-2n_{If}}}{12}$, and $\sigma_w^2 = \frac{2^{-2n_{wf}}}{12}$ for the filter coefficients), it can be shown that the average power of the error (MSE, or *Mean-Square Error*) at the output in steady-state is [33]

$$\sigma_o^2(c,d) = \sigma_{\min}^2 + \frac{\mu\sigma_{\min}^2 \mathrm{tr}\boldsymbol{R}}{2-\mu\mathrm{tr}\boldsymbol{R}} + \left(||\boldsymbol{w}^*||^2 + \frac{1}{2}\mu\sigma_{\min}^2 N_E\right)\sigma^2 + c\sigma_I^2 + \frac{N_E\sigma_w^2 + d\cdot\mathrm{tr}\boldsymbol{R}\sigma_I^2 + \mu^2\cdot\sigma^2\left(\left(1+c\frac{\sigma_I^2}{\sigma^2}+||\boldsymbol{w}^*||^2\right)\cdot\mathrm{tr}\boldsymbol{R} + \sigma_{\min}^2 N_E\right)}{2\mu-\mu^2\mathrm{tr}\boldsymbol{R}} \tag{4}$$

where the first two terms correspond to the error of the LMS filter with infinite precision, and the rest of the terms stem from quantization. In Eq. (4), $\sigma_{\min}^2 = \sigma_d^2 - \boldsymbol{w}^* E\{d_n \mathbf{u}_n\}$ is the error of the optimum Wiener filter $\boldsymbol{w}^*$, $\mathrm{tr}\boldsymbol{R}$ represents the trace of the input covariance matrix, and c and d are factors that depend on the way quantization is handled in multiplications:

$$c = \begin{cases} 1, \text{ if only the result of } \boldsymbol{w}_n^T\cdot\boldsymbol{u}_n \text{ in (1) is quantized} \\ N_E, \text{ if each intermediate product of } \boldsymbol{w}_n^T\cdot\boldsymbol{u}_n \text{ in (1) is quantized.} \end{cases}$$

$$d = \begin{cases} 1, \text{ if the product } \mu e_n \text{ is quantized before multiplying by } \boldsymbol{u}_n \text{ in (2)} \\ 0, \text{ if there is no intermediate quantization in } \mu e_n\boldsymbol{u}_n \text{ in (2).} \end{cases}$$

[2]The calculations can be generalized to any u_n through the rotated or uncoupled coordinate space [32], but the i.i.d. case is representative enough of the effects of fixed-point precision on the output error.

Equation (4) is not exactly the same as in [33], as we have considered the most general case of having different quantization levels for inputs, filter coefficients, and also for intermediate values.

If only the inputs are quantized, but the intermediate operations do not perform any additional quantization, then the MSE at the output will be, following a derivation analogous to [33],

$$\sigma^2_{o,\mathrm{QI}} = \sigma^2_{\min} + \frac{\mu\sigma^2_{\min}\mathrm{tr}\boldsymbol{R}}{2-\mu\mathrm{tr}\boldsymbol{R}} + \left(||\boldsymbol{w}^*||^2 + \frac{1}{2}\mu\sigma^2_{\min}N_E\right)\sigma^2.$$

Hence, for the studied non-block protocols, the error at the output can be expressed as

$$\sigma^2_{\mathrm{HP}} = \sigma^2_{o,\mathrm{QI}}, \quad \sigma^2_{\mathrm{GC}} = \sigma^2_o(N_E,1), \quad \sigma^2_{\mathrm{Hy}} = \sigma^2_o(1,0).$$

For the fast protocol, the quantization error has a different shape, but the independence assumptions can be applied exactly as in the other protocols, duplicating the power of this quantization error of the intermediate values, that becomes $\sigma^2_I = 2^{-2n_{If}}/6$.

1) Block LMS protocol: Following a similar derivation to that of Caraiscos and Liu [33] for the BLMS algorithm[3], with the same independence assumptions, it is possible to generalize their formula to provide the following approximation to the error in the Block LMS implementation:

$$\begin{aligned}\sigma^2_{o,Bk}(c,d,N_b) =& \sigma^2_{\min} + \frac{\mu\sigma^2_{\min}\mathrm{tr}\boldsymbol{R}}{2-\mu\mathrm{tr}\boldsymbol{R}} + \left(||\boldsymbol{w}^*||^2 + \frac{1}{2}\mu\sigma^2_{\min}N_E\right)\sigma^2 + c\sigma^2_I \\ &+ \frac{\frac{N_E\sigma^2_w}{N_b} + d\cdot\left(N_E\frac{N_b-1}{N_b}\sigma^2_w + \sigma^2_I\mathrm{tr}(\boldsymbol{R})\right) + \mu^2\cdot\sigma^2\left(\left(1+c\frac{\sigma^2_I}{\sigma^2}+||\boldsymbol{w}^*||^2\right)\cdot\mathrm{tr}\boldsymbol{R} + \sigma^2_{\min}N_E\right)}{2\mu-\mu^2N_b\mathrm{tr}\boldsymbol{R}}\end{aligned} \tag{5}$$

where c has the same meaning as in Eq. (4), N_b is the block size, and $d=1$ when each product in $\mu\sum_k e_k\boldsymbol{u}_k$ in (3) is individually quantized, and $d=0$ otherwise.

This result is coherent with the one obtained by Eweda *et al.* [34] for the adaptive system identification problem, but Eq. (5) is more general and takes into account more parameters that allow for a greater flexibility in predicting the error of our implementations. It can be seen that for the same step size μ, both infinite-precision LMS and BLMS have the same misadjustment (first two terms in Eq. (5)) and the same average time constant. For the finite-precision algorithms, Eq. (5) shows that the BLMS reduces the sensitivity to the quantization error in the filter coefficients when $d=0$ (first term of the numerator), but the sensitivity to the quantization of the inputs is not altered (third term in Eq. (5)); quantization of the filter coefficients has a much more critical and noticeable effect than the quantization of the input values when σ^2 and σ^2_w are comparable, what motivates the conclusions in [34] about the better behavior of BLMS; nevertheless, when $\sigma^2 \gg \sigma^2_w$, the averaging performed by

[3]The full derivation is rather direct but lengthy, and it is completely shown in [6].

BLMS has a neglibible impact on quantization error resilience, as shown in Section VII-B3; hence, for the same convergence speed, BLMS presents an MSE similar to that of LMS.

2) Transient Deviation due to Finite Precision: As shown in the previous sections, the use of fixed-point precision affects the stationary regime of the algorithms, producing a higher level of noise. Actually, the effect of finite precision is also noticeable in the transient period, introducing errors during tracking and altering the adaptation behavior. Following a similar derivation to that in [35], we have extended the theoretical adaptation curve to the BLMS algorithm. The result for the weigth vector misadjustment $\mathcal{M}_n = E\left[\Delta \boldsymbol{w}_n^T \Delta \boldsymbol{w}_n\right]$, for the same assumptions as in previous sections, is

$$\mathcal{M}_n = \mu^2 \cdot N_b \cdot N_E \cdot \left[A \cdot n\gamma^{2(n-1)} + \frac{A}{\gamma - \gamma^2}\left(\gamma^n - \gamma^{2n}\right) + \frac{B}{1-\gamma^2}\left(1-\gamma^{2n}\right)\right] + \frac{N_E \sigma_w^2}{1-\gamma^2}\left(1-\gamma^{2n}\right), \quad (6)$$

with

$$A = 2\sigma^2\sigma_u^2||\boldsymbol{w}^*||^2, \quad B = \sigma_u^2\left[\sigma^2(1+||\boldsymbol{w}^*||^2) + c\sigma_I^2\right] + \sigma^2\sigma_{\text{min}}^2, \quad \gamma = 1 - \mu N_b \sigma_u^2.$$

Eq. (6) gives the evolution of the MSE of the filter coefficients that the finite precision algorithm introduces with respect to the infinite precision LMS during the adaptation period. The notation and parameters are the same as for Eq. (5). This error evolves with a fixed time constant, equal to that of the infinite precision algorithm, until reaching the stationary state for which the output error is given by Eq. (5). This evolution is shown in Figure 3 for the hybrid protocol for different values of the adaptation step and used fractional bits. For a fair comparison, it must be taken into account that the index n refers to successive updates of the vector coefficients, that in BLMS are produced every N_b output samples instead of every sample.

3) Comparison and Evaluation: Figure 4 shows a representative case of the excess MSE (i.e., $E\{e^2\} - \sigma^2_{LMS_\infty}$) with respect to the infinite precision LMS, obtained for each of the proposed protocols for varying bit-size of the fractional part. The theoretical approximations given by Eq (5) are labeled with the subindex th, and the experimental results, with the subindex exp. The Garbled Circuit implementation presents the highest error, mainly due to the use of the same bit size for vector coefficients as for input quantization, and the quantization performed after each multiplication. The hybrid protocol is the most robust against quantization errors, due to the use of a higher resolution for the vector coefficients, and the presence of quantization only in the outputs, and in no other internal calculations. On the other hand, the fast protocol presents a MSE slightly higher than the hybrid protocol, due to the approximate quantization of the outputs. Finally, the MSE produced by the block protocols is virtually the same as the MSE of the corresponding non-block implementations, due to the predominant effect of input quantization over that of filter coefficients quantization. The experimental results are obtained as the average error after running the algorithms for 40968 iterations in steady-state regime, for the system identification setup with $\sigma_u^2 = 0.25$, $\sigma_d^2 = 0.2821$, $\mu = 2^{-8}$,

$\sigma^2_{\text{min}} = 2.5 \cdot 10^{-5}$ and $\sigma^2_{LMS_\infty} = 2.5147 \cdot 10^{-5}$. The homomorphic processing protocol is not shown, as its cipher blows up before reaching the steady-state in practical cases; e.g., a modulus of 2048 bits can only hold 28 iterations using 48 bit numbers with 8 bits for the fractional part. Nevertheless, in theory and with a big enough cipher, it would be the most robust protocol due to the absence of intermediate quantizations. Besides this protocol, the concordance between the theoretical approximation and the experimental results in all the other protocols is remarkable, given the magnitude of the errors with which we are working, assessing the validity of the initial assumptions for obtaining Eq (5).

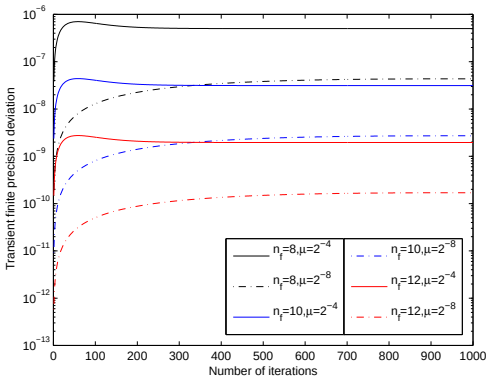


Fig. 3: Excess error during the transient period for the hybrid protocol, with $n_x = 48$, and a 4-tap adaptive filter.

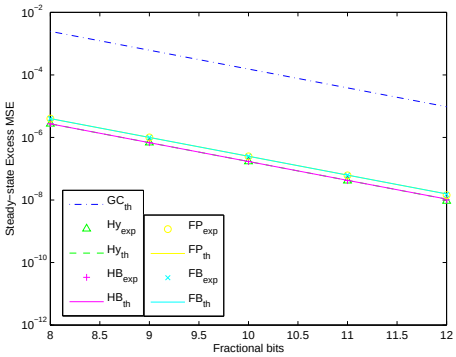


Fig. 4: Steady-state excess error for varying fractional precision, with $n_x = 48$, and 12-tap adaptive filter, packing $N_b = N_E = 12$ coefficients in the block protocols.

There are several effects noticeable in Figure 4 that deserve a comment: on the one hand, the experimental results for the Garbled Circuit protocol are not shown, as for the used bit-sizes the precision used for filter coefficients is too low (equal to that of the inputs and intermediate results), and it suffers from stalling effects, that prevent it from converging; as a consequence, it needs a much higher precision in order to avoid stalling, and even when converging, as shown in the plot, the error that it produces is significantly higher than that of the other protocols. The second observable fact is that the gap of precision in block protocols is almost negligible when $\sigma^2 \gg \sigma^2_w$. This difference is not noticeable in Figure 4, and it would only be significant with very long blocks $N_b \gg 1$ or with $\sigma^2 \approx \sigma^2_w$. The way our protocols are designed avoids this second condition, as they use always a higher precision for the filter coefficients than for the inputs/outputs.

At last, the value of N_b is limited by the maximum plaintext size and the number of bits used for representing each number. Thus, Eq. (5) can be used together with the packing limits for the block protocols $N_b^{(FB)} \leq \lfloor \frac{n_{\text{cipher}}}{n_b + n_{\text{sec}}} \rfloor$, $N_b^{(HB)} \leq \lfloor \frac{n_{\text{cipher}} - n_{\text{sec}}}{n_b} \rfloor$, for finding a trade-off between the committed error due to the used precision, and the complexity of both protocols, dependent on the number of

coefficients that are packed together.

VIII. Practical Implementation

In this section, we present and comment the results of a practical implementation of the proposed protocols. For this purpose, we have chosen the Damgård-Jurik [11] extension of Paillier cryptosystem, due to its flexibility for fitting larger plaintexts with a constant expansion ratio. For the protocols involving garbled circuits, we have chosen the `XOR`-free garbled circuit solution in [17], and the efficient oblivious transfer protocols of [28] with EC-ElGamal encryptions, aiming to the most efficient algorithms currently available for implementing garbled circuits.

For the evaluation of computational complexity, we have implemented the presented protocols and their block versions in C++ using the `crypto++` library [36] for the elliptic curves cryptosystems, and the GNU `GMP` library [37] for multiprecision arithmetic, and we have provided our own implementation of Damgård-Jurik encryptions, with some efficiency improvements in modular exponentiations, detailed in Appendix A. We use these implementations in order to plot the execution times of the three protocols and compare them in terms of CPU usage. We have made the whole software package of our implementation available at [38].

A. Computational Load

We have measured the computational load of the developed algorithms through the total computation time that their efficient implementation yields on a PC with no parallelization, for a fair comparison. Nevertheless, these protocols, and especially their block versions, are easily parallelizable, obtaining a great reduction in execution time when several cores are available. The experiments were performed using our C++ implementation on an Intel Core2Duo processor at 3 GHz with 4GB of RAM running a 64-bit linux distribution. In order to measure only computation times, we have neglected the communication stack, and we have run in the same core the client and the server sequentially, obtaining the aggregated computation times for both parties.

Figure 5 shows the aggregated computation time for the 48 initial iterations of each of the presented protocols, as a function of the filter size. The three protocols involving garbled circuits are the most expensive ones, due to the load that oblivious transfers impose. While this load is normally absorbed through precomputation, with an adaptive algorithm it is not possible to perform the heavy encryption operations a priori, as they involve the results generated in each iteration; hence, no precomputation is applied to any of the performed operations. This has also an impact on their parallelization, as each oblivious transfer round involves only the bits of one input. This is especially critical in the case of the hybrid protocol, as the small OTs in each iteration cannot be joined together into a longer and more efficiently reducible OT. On the other hand, the packing performed in the hybrid block protocol

allows for this reduction, greatly improving computational load as the number of packed coefficients (chosen to equal the size of the filter) increases.

Finally, the execution times of the fast protocols are several orders of magnitude below those of the garbled circuits solutions, and slightly increase the complexity of the homomorphic computation protocol due to the addition of the rounding protocols. This is a remarkable result, taking into account that without this rounding subprotocols, the whole homomorphic computation protocol is completely unusable due to cipher blowup. For the fast protocol, the block-based one does not improve on the computational load, as the fast rounding protocol requires a whole unpacking protocol for each of the packed numbers, and it does not yield the same improvement as in the hybrid block protocol. Hence, the fast protocol is more time-efficient than its block version.

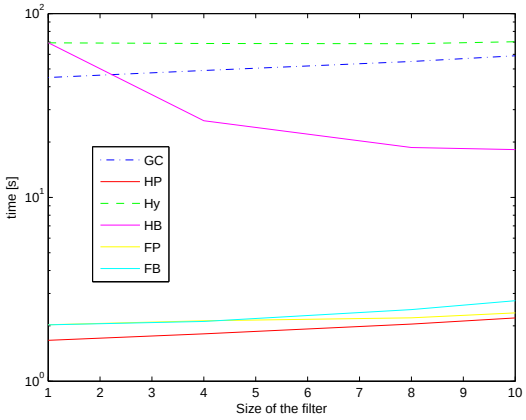


Fig. 5: Aggregated computation time for 2048 bits moduli, $|E_C| = 224$, $n_{\text{sec}} = 80$, $n_x = 32$, $n_f = 16$, 48 iterations and increasing filter size and maximum packed coefficients.

IX. Conclusions and Further Work

Addressing privacy in adaptive filtering applications is an important open issue in the field of Signal Processing in the Encrypted Domain. In this work, we have presented the problem of privacy-preserving adaptive filtering, with several representative scenarios and their trust model and privacy requirements. Due to the impossibility of using a practical full homomorphism, we have proposed several novel solutions employing different techniques, like garbled circuits, additive homomorphisms and interactive protocols, looking for the optimal trade-off in terms of complexity and output error; we have also provided several private quantization algorithms of independent interest to tackle the cipher blowup problem; we have implemented all our novel protocols for the Private LMS algorithm in a working prototype, and we have performed a comparison in terms of bandwidth and computational complexity, concluding that garbled circuits are still far from providing an efficient solution to adaptive

filtering, and interactive approximate protocols with statistical security can yield much more practical solutions.

We have also tackled the issue of the limitation to fixed-point precision when working with encrypted values, resorting to analytical studies on the impact of finite-precision in the output error of the used adaptive filters, during the transient period and in steady-state regime, particularizing the expressions to each of the studied cases. The fast protocols that we have introduced are almost as robust as the original (B)LMS algorithm with respect to quantization errors, while presenting low computational and communication complexity.

This work covers the two main problems of any secure adaptive filtering algorithm, namely cipher blowup and precision limits due to the use of fixed point arithmetic. Further research will aim also at the implementation of more complex nonlinear functions, being this problem not specific of adaptive filtering. Hence, the present work opens the door to further improvements in secure adaptive filtering, setting the basis and a reference implementation for the development of new solutions.

ACKNOWLEDGMENTS

This work was partially funded by Xunta de Galicia under projects “Consolidation of Research Units” 2010/85, SAFECLOUD (ref. 09TIC014CT), SCALLOPS (ref. 10PXIB322231PR) and VISAGE (ref. 10TIC008CT), by the Spanish Ministry of Science and Innovation under project COMONSENS (ref. CSD2008-00010) of the CONSOLIDER-INGENIO 2010 Program, by the Iberdrola Foundation through the Prince of Asturias Endowed Chair in Information Science and Related Technologies, and by the Spanish MEC FPU grant ref. AP2006-02580.

REFERENCES

[1] A. C. Yao, “Protocols for secure computations,” in *Proceedings of the IEEE Symposium on Foundations of Computer Science*, 1982, pp. 160–164.

[2] O. Goldreich, S. Micali, and A. Widgerson, “How to play any mental game,” in *Proceedings of the nineteenth annual ACM conference on Theory of Computing*. New York, U.S.A.: ACM Press, 1987, pp. 218–229.

[3] C. Gentry, “Fully homomorphic encryption using ideal lattices,” in *Proceedings of the 41st annual ACM symposium on Theory of computing, STOC’09*. Bethesda, MD, USA: ACM Press, May-June 2009, pp. 169–178.

[4] B. Widrow and M. Hoff, “Adaptive Switching Circuits,” in *IRE WESCON Convention Record*. New York: IRE, 1960.

[5] G. Clark, S. Mitra, and S. Parker, “Block implementation of adaptive digital filters,” *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 29, no. 3, pp. 744–752, June 1981.

[6] J. R. Troncoso-Pastoriza and F. Pérez-González, “University of Vigo Technical Report UV/DTC/JRTP/22/12/2010,” University of Vigo, Tech. Rep., September 2010. [Online]. Available: http://www.gts.tsc.uvigo.es/~troncoso/Tech_rep_UV_DTC_JRTP_22_12_2010.pdf

[7] J.-J. E. Slotine and W. Li, *Applied Nonlinear Control*. Prentice Hall, 1991.

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

[9] P. Paillier, “Public-key cryptosystems based on composite degree residuosity classes,” in *Advances in Cryptology - EUROCRYPT 1999*, ser. Lecture Notes in Computer Science, vol. 1592. Springer, 1999, pp. 223–238.

[10] I. Damgård, M. Geisler, and M. Krøigaard, “Efficient and Secure Comparison for On-Line Auctions,” in *Australasian Conference on Information Security and Privacy – ACSIP 2007*, ser. LNCS, vol. 4586. Springer, July 2007, pp. 416–430.

[11] I. Damgård and M. Jurik, “A generalisation, a simplification and some applications of Paillier’s probabilistic public-key system,” in *Public Key Crytography 2001*, ser. Lecture Notes in Computer Science, K. Kim, Ed., vol. 1992. Cheju Island, Korea: Springer, February 2001, pp. 119–136.

[12] N. Smart and F. Vercauteren, “Fully Homomorphic Encryption with Relatively Small Key and Ciphertext Sizes,” in *13th International Conference on Practice and Theory in Public Key Cryptography 2010*, ser. LNCS, vol. 6056, Paris, France, May 2010, pp. 420–443.

[13] A. Shamir, “How to share a secret,” *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[14] I. Damgård, M. Fitzi, E. Kiltz, J. B. Nielsen, and T. Toft, “Unconditionally secure constant-rounds multi-party computation for equality, comparison, bits and exponentiation,” in *Proceedings of the third Theory of Cryptography Conference, TCC 2006*, ser. Lecture Notes in Computer Science, vol. 3876. Springer-Verlag, 2006, pp. 285–304.

[15] T. Nishide and K. Ohta, “Constant-round multiparty computation for interval test, equality test, and comparison,” *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E90-A, no. 5, pp. 960–968, May 2007.

[16] B. Schoenmakers and P. Tulys, “Efficient binary conversion for Paillier encrypted values,” in *Advances in Cryptology - EUROCRYPT 2006*, ser. Lecture Notes in Computer Science, vol. 4004. Springer, 2006, pp. 522–537.

[17] V. Kolesnikov and T. Schneider, “Improved garbled circuit: Free XOR gates and applications,” in *ICALP’08*, ser. LNCS, vol. 5126. Springer, 2008, pp. 486–498.

[18] V. Kolesnikov, A.-R. Sadeghi, and T. Schneider, “How to Combine Homomorphic Encryption and Garbled Circuits – Improved Circuits and Computing the Minimum Distance Efficiently,” in *SPEED Workshop*, Lausanne, Switzerland, September 2009, pp. 100–121.

[19] “Signal Processing in the EncryptEd Domain project (SPEED).” [Online]. Available: http://www.speedproject.eu/

[20] T. Bianchi, A. Piva, and M. Barni, “On the Implementation of the Discrete Fourier Transform in the Encrypted Domain,” *IEEE Transactions on Information Forensics and Security*, vol. 4, no. 1, pp. 86–97, 2009.

[21] ——, “Encrypted Domain DCT based on Homomorphic Cryptosystems,” *EURASIP Journal on Information Security*, vol. 2009, no. Article ID 716357, 2009.

[22] T.Bianchi, A.Piva, and M.Barni, “Composite Signal Representation for Fast and Storage-Efficient Processing of Encrypted Signals,” *IEEE Trans. on Information Forensics and Security*, vol. 5, no. 1, pp. 180–187, March 2010.

[23] J. Canny, “Collaborative filtering with privacy,” in *Security and Privacy, 2002. Proceedings. 2002 IEEE Symposium on*, 2002, pp. 45 – 57.

[24] Z. Erkin, “Secure signal processing: Privacy preserving cryptographic protocols for multimedia,” Ph.D. dissertation, Technische Universiteit Delft, 2010.

[25] G. Jagannathan, K. Pillaipakkamnatt, and D. Umano, “A Secure Clustering Algorithm for Distributed Data Streams,” in *Data Mining Workshops, 2007. ICDM Workshops 2007. Seventh IEEE International Conference on*, 28-31 2007, pp. 705 –710.

[26] J. R. Troncoso-Pastoriza, P. Comesaña, and F. Pérez-González, “Secure direct and iterative protocols for solving systems of linear equations,” in *SPEED Workshop*, Lausanne, Switzerland, September 2009, pp. 122–141.

[27] M. Franz, S. Katzenbeisser, S. Jha, K. Hamacher, H. Schroeder, and B. Deiseroth, “Secure computations on non-integer values,” in *IEEE WIFS’10*. Seattle, USA: IEEE, December 2010.

[28] Y. Ishai, J. Kilian, K. Nissim, and E. Petrank, "Extending Oblivious Transfer Efficiently," in *Advances in Cryptology CRYPTO'03*, vol. 2729, 2003, pp. 145–161.

[29] N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of Computation*, vol. 48, no. 177, pp. 203–209, January 1987.

[30] H. Lipmaa, "Verifiable homomorphic oblivious transfer and private equality test," in *ASIACRYPT'01*, ser. LNCS, vol. 2894. Springer, 2003.

[31] J. R. Troncoso-Pastoriza, S. Katzenbeisser, M. Celik, and A. Lemma, "A secure multidimensional point inclusion protocol," in *9th ACM Workshop on Multimedia and Security (MMSEC'07)*, Dallas, Texas, USA, September 2007, pp. 109–120.

[32] S. T. Alexander, *Adaptive Signal Processing*. Springer-Verlag, 1986.

[33] C. Caraiscos and B. Liu, "A roundoff error analysis of the LMS adaptive algorithm," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 32, no. 1, pp. 34–41, Feb 1984.

[34] E. Eweda, N. Yousef, and S. El-Ramly, "Reducing the Effect of Finite Wordlength on the Performance of an LMS Adaptive Filter," in *EEE International Conference on Signal Communications*, Atlanta, GA, USA, June 1998, pp. 688–692.

[35] S. Alexander, "Transient weight misadjustment properties for the finite precision LMS algorithm," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 35, no. 9, pp. 1250–1258, Sep 1987.

[36] "Crypto++ Library." [Online]. Available: http://www.cryptopp.com/

[37] "GNU MP Bignum Library." [Online]. Available: http://gmplib.org/

[38] J. R. Troncoso-Pastoriza, "PrivateLMS: Prototype Protocols for the Private Execution of the LMS algorithm," August 2010. [Online]. Available: http://www.gts.tsc.uvigo.es/~troncoso/privatelmsimplementation

APPENDIX A
FAST ENCRYPTION AND DECRYPTION FOR DAMGÅRD-JURIK CRYPTOSYSTEM

Encryption and decryption are two of the most costly operations, due to the heavy modular exponentiations that they must perform. For our implementations, we have used a different version of the decryption operation, and for the private encryption of the Paillier cryptosystem (and the Damgård-Jurik extension) that enhance the performance of the original methods. This appendix describes both methods. Modular exponentiations are the most computationally demanding basic operations, whose complexity is linear in the exponent size $|e|$ and quadratic in the modulus size $|n|$ (i.e., $O(|e||n|(|n|-1))$). Thus, reducing the bit size of the involved operands yields important efficiency gains. The presented reductions are based on using the knowledge of the factorization of the public modulus n, enhancing all decryption operations and encryption operations performed by a party with decryption privileges (*private* encryption). Looking at the most common two-party scenarios of homomorphic encryption, the party that owns the data and owns the decryption keys is usually the client, that normally has a processing power lower than the server; hence, it makes sense to optimize the operations that this party must perform, and this is exactly what our modifications do. We will preserve the notation used in Section IV-A.

a) Decryption: Let $L_a(b)$ be defined as $L_a(b) = \frac{b-1}{a}$, for $b \equiv 1 \mod a, 0 < b < a^2$, as in Paillier's work. In [11], it is suggested that the decryption operation, after the exponentiation c^d

$\mod n^{s+1}$, be divided into two parts, using $L'_p(c^d) = L_p(c^d) \cdot q^{-1}$ and $L'_q(c^d) = L_q(c^d) \cdot p^{-1}$ instead of $L_n(c^d)$, and then joined using the Chinese Remainder Theorem (CRT). While this strategy can provide a speed-up in the computations, as each part of the decryption works with half-sized numbers, the initial exponentiation is still the most costly operation. We next show how the knowledge of the factorization of n allows also for breaking up this exponentiation into two parts.

For a message x, its encryption $c = (1+n)^x r^{n^s} \mod n^{s+1}$, can be reduced modulo p^{s+1} and q^{s+1}, obtaining two partial encryptions with half the size of c: $c_p = (1+n)^x r^{n^s} \mod p^{s+1}$ and $c_q = (1+n)^x r^{n^s} \mod q^{s+1}$. By Carmichael's Theorem, the order of the units in the group $\mathbb{Z}_{p^{s+1}}$ (resp. $\mathbb{Z}_{q^{s+1}}$) is a divisor of $p^s(p-1)$ (resp. $q^s(q-1)$). Hence, the minimum exponent that cancels the effect of r^{n^s} is $p-1$ (resp. $q-1$), that is

$$\begin{aligned} L'_p(c_p^{p-1}) =& L_p\left(((1+n)^x)^{p-1} r^{q^s \cdot p^s \cdot (p-1)}\right) \cdot q^{-1} \\ \equiv& \left(1 + x(p-1)n + \begin{pmatrix} x(p-1) \\ 2 \end{pmatrix} n^2 + \ldots + \begin{pmatrix} x(p-1) \\ s \end{pmatrix} n^s\right) \mod p^s, \end{aligned}$$

and analogously for q. Applying the decryption algorithm with p and q for both parts, and multiplying afterwards each of them by the inverses of $p-1$ and $q-1$, the desired result is obtained:

$$d_p = \mathsf{dec}_{p^s}(c_p^{p-1}) \cdot (p-1)^{-1} \equiv x \mod p^s, \quad d_q = \mathsf{dec}_{q^s}(c_q^{q-1}) \cdot (q-1)^{-1} \equiv x \mod q^s.$$

The application of the CRT yields that, if a_p and a_q are two integers such that $a_p \cdot p^s + a_q \cdot q^s = 1$, then $x \equiv d_p \cdot a_q \cdot q^s + d_q \cdot a_p \cdot p^s \mod n^s$.

Finally, as the values of $(p-1)^{-1} \mod p^s$, $(q-1)^{-1} \mod q^s$, $a_q \cdot q^s \mod n^s$ and $a_p \cdot p^s \mod n^s$ can be precalculated, and the L' functions can be executed once for the highest power of p and q and subsequently modularized for the rest of the iterations of the algorithm (as $L_b(a \mod b^{j+1}) \equiv L_b(a \mod b^{s+1}) \mod b^j$), neglecting the complexity of a modularization and the addition/subtraction of a unit, the total decryption complexity is reduced to

$$2\left(X_{\frac{(s+1)|n|}{2},\frac{|n|}{2}} + D_{\frac{(s+1)|n|}{2}} + P_{s|n|} + \sum_{k=2}^{s}\left((k-1)(P_{\frac{k|n|}{2}} + A_{\frac{k|n|}{2}})\right)\right) + A_{s|n|},$$

where $X_{a,b}$ is the computational complexity of an exponentiation with modulus size a and exponent size b, A_b and P_b are the complexity of a modular addition and product with modulus size b respectively, and D_a is the complexity of an integer division with dividend's size a. This results can be compared to the complexity of a regular decryption, performed as stated in [11],

$$X_{(s+1)|n|,|n|} + D_{(s+1)|n|} + \sum_{k=2}^{s}\left((k-1)(P_{k|n|} + A_{k|n|})\right).$$

The reduction factor in complexity due to splitting the exponentiation is almost four.

b) Encryption: For regular encryption there is no additional gain to the one pointed out in Paillier's original work, by virtue of which taking $g = 1 + n$ reduces the exponentiation $g^x \mod n^2$ to a product $g^x \equiv (1+x\cdot n) \mod n^2$, generalized in [11] to n^{s+1} as a sum of s chained products; the exponentiation r^{n^s} is, in principle, unavoidable. Nevertheless, when the encryption is performed by a party with decryption capabilities ("private" encryption), the knowledge of the private key allows for further improvements on efficiency, applying the same rationale as for fast decryption. In this case, the reduction seeks partitioning the exponentiation r^{n^s} into two exponentiations with half-sized base and exponent.

Given $a_{p^{s+1}}$ and $a_{q^{s+1}}$ such that $a_{p^{s+1}} \cdot p^{s+1} + a_{q^{s+1}} \cdot q^{s+1} = 1$, $r^{n^s} \mod n^{s+1}$ can be calculated as

$$r_p \equiv r^{p^s (q^s \mod (p-1))} \mod p^{s+1}, \quad r_q \equiv r^{q^s (p^s \mod (q-1))} \mod q^{s+1},$$
$$r^{n^s} \equiv r_p \cdot a_{q^{s+1}} \cdot q^{s+1} + r_q \cdot a_{p^{s+1}} \cdot p^{s+1} \mod n^{s+1}.$$

Precalculating the values of $a_{q^{s+1}} \cdot q^{s+1} \mod n^{s+1}$ and $a_{p^{s+1}} \cdot p^{s+1} \mod n^{s+1}$, the complexity of each encryption is reduced to

$$2X_{\frac{(s+1)|n|}{2},\frac{(s+1)|n|}{2}} + 2(s+1)P_{(s+1)|n|} + 2s \cdot A_{(s+1)|n|},$$

compared to $X_{(s+1)|n|,(s+1)|n|} + 2s \cdot P_{(s+1)|n|} + (2s-1)A_{(s+1)|n|}$ of a normal encryption, which yields a complexity reduction almost by a factor of four.

APPENDIX B
CIPHER RENEWAL: QUANTIZATION UNDER ENCRYPTION

In order to renew the cipher and eliminate part of the excess of precision accumulated by the lack of a division operation, it is necessary to quantize the encrypted values. For this purpose, and to preserve perfect secrecy, we have developed interactive protocols of independent interest for performing quantization:

Let $[x] \in \mathbb{Z}_n$ be a class in $\mathbb{Z}_n$, and x its positive representative in the interval $x \in [0, n)$. $\mathcal{A}$ and $\mathcal{B}$ possess their respective shares x_A, x_B of x (i.e. $x_A + x_B \equiv x \mod n$). Both $\mathcal{A}$ and $\mathcal{B}$ want to requantize x with a step $\Delta \in (2, \lceil n/2 \rfloor)$, with a maximum quantization error of Δ. Let us assume that $\mathcal{A}$ knows the decryption key of an additive homomorphic cryptosystem, and both $\mathcal{A}$ and $\mathcal{B}$ can produce encryptions using this cryptosystem. The scenario can be plotted also with a threshold homomorphic cryptosystem, with straightforward modifications.

If $\mathcal{B}$ owns an encryption of $[\![x]\!]$, then he generates a random $x_B \in \mathbb{Z}_n$, blinds with it the encryption of $[\![x]\!]$, and sends the result $[\![x + x_B \mod n]\!]$ to $\mathcal{A}$, who decrypts $x_A = x + x_B \mod n$. Then, both parties start with a share of x.

Each party quantizes his/her share $x_{AQ} = \left\lfloor \frac{x_A}{\Delta/2} \right\rfloor$, $x_{BQ} = \left\lfloor \frac{x_B}{\Delta/2} \right\rfloor$; with these values, both parties can obtain the bit representation of their respective quantities and run a binary comparison protocol (cf. [6]) $x_{BQ} > \left\lceil \frac{n}{\Delta/2} \right\rceil - x_{AQ}$, ending up with an encryption of the binary comparison.

Then, $\mathcal{A}$ can obtain $[\![Q_R(x)]\!] = [\![x_{AQ}]\!] + [\![x_{BQ}]\!] - \left\lceil \frac{n}{\Delta/2} \right\rceil \cdot \left[\!\left[x_{BQ} \geq \left\lceil \frac{n}{\Delta/2} \right\rceil - x_{AQ} \right]\!\right]$. We denote the result $Q_R(x)$ because it does not coincide exactly with the quantization $Q(x)$ when performed in the clear, because $Q_R(x)$ is quantized with a precision of $\Delta/2$, but the split in two shares introduces an error of ± 1 in the quantization of x. Thus, even when the obtained precision is Δ, the resulting encrypted number must be scaled by $\Delta/2$ after decryption in order to obtain the true quantized value.

The previous protocol could be thought of as a *fast* version of the quantization protocol, that has the drawback of introducing some noise due to the independent quantization of both shares. When the quantization must yield exactly the same results as in the clear, we can use an *exact* version of the previous protocol, that provides a perfect quantization, with the same result as if performed in the clear, at the cost of an increased computation and communication complexity. We now describe this *exact* solution.

After splitting x in two shares x_A and x_B, each party quantizes his share with step Δ, obtaining respectively $x_{AQ} = \lceil \frac{x_A}{\Delta} \rfloor$, $x_{Ar} = x_A \mod \Delta$, and $x_{BQ} = \lceil \frac{x_B}{\Delta} \rfloor$, $x_{Br} = x_B \mod \Delta$; both have the quantity $n_\Delta = n \mod \Delta$ in the clear. The quantization of x as a function of the previous four values can be expressed as

$$
\begin{aligned}
[\![Q(x)]\!] = [\![x_{AQ}]\!] + [\![x_{BQ}]\!] + \left(1 - 2\left[x_{Br} \geq \left\lceil \frac{\Delta}{2} \right\rceil\right]\right) \cdot \left(1 - xor\left(\left[\!\left[x_{Ar} \geq \left\lceil \frac{\Delta}{2} \right\rceil\right]\!\right], \left[x_{Br} \geq \left\lceil \frac{\Delta}{2} \right\rceil\right]\right)\right) \\
\cdot \left(\left[\!\left[x_{Ar} + x_{Br} \in \left[\left\lceil \frac{\Delta}{2} \right\rceil, \left\lceil \frac{3\Delta}{2} \right\rceil\right)\right]\!\right] + \left([\![x_{Ar} + x_{Br} \in \mathcal{I}_{n_\Delta}]\!] - \left[\!\left[x_{Ar} + x_{Br} \in \left[\left\lceil \frac{\Delta}{2} \right\rceil, \left\lceil \frac{3\Delta}{2} \right\rceil\right)\right]\!\right]\right) \\
\cdot \left[\!\left[x_{BQ} \geq \left\lceil \frac{n}{\Delta} \right\rceil - x_{AQ}\right]\!\right]\right) - \left\lceil \frac{n}{\Delta} \right\rceil \cdot \left[\!\left[x_{BQ} \geq \left\lceil \frac{n}{\Delta} \right\rfloor - x_{AQ}\right]\!\right].
\end{aligned}
$$

As the only needed binary operation is the exclusive-OR, for efficiency reasons we avoid the use of garbled circuits and implement it homomorphically as $xor(a,b) = a + b - 2a \cdot b$ in $\mathbb{Z}_n$. The set $\mathcal{I}_{n_\Delta}$ represents an interval reduced modulo 2Δ:

$$
\mathcal{I}_{n_\Delta} = \begin{cases} \left[\left\lceil \frac{3\Delta}{2} \right\rceil + n_\Delta, \left\lceil \frac{\Delta}{2} \right\rceil + n_\Delta\right)_{2\Delta}, & \text{if } n_\Delta \geq \left\lceil \frac{\Delta}{2} \right\rceil \\ \left[\left\lceil \frac{\Delta}{2} \right\rceil + n_\Delta, \left\lceil \frac{3\Delta}{2} \right\rceil + n_\Delta\right)_{2\Delta}, & \text{if } n_\Delta < \left\lceil \frac{\Delta}{2} \right\rceil, \end{cases}
$$

being $[,)_{2\Delta}$ the modular reduction of the interval with modulus 2Δ.

The binary comparisons $x_{Ab} = [x_{Ar} \geq \lceil \frac{\Delta}{2} \rceil]$ and $x_{Bb} = [x_{Br} \geq \lceil \frac{\Delta}{2} \rceil]$ are performed by each party independently. $\mathcal{A}$ can encrypt $[\![x_{Ab}]\!]$ and send it to $\mathcal{B}$, who can perform $(1 - 2[x_{Bb}]) \cdot (1 - xor([\![x_{Ab}]\!], [x_{Bb}]))$ using only homomorphic operations. Each of the two needed interval checks can be performed through two comparison circuits and a homomorphic sum ($[\![x \in [a,b)]\!] = [\![x \geq a]\!] - [\![x \geq b]\!]$). After obtaining these values, the whole expression can be evaluated with 5 homomorphic sums and 3 invocations of the secure multiplication protocol.

The total complexity calculated for the *exact* protocol, for a modulus bit-size $|n| = l$, is

$$\begin{aligned}
\mathrm{C}_{EQcm}(n,\Delta) =& |E| + 3\mathrm{C}_{MULTcm} + 4\mathrm{C}_{COMPcm}\left(\lceil \log_2 \Delta \rceil + 1\right) + \mathrm{C}_{COMPcm}\left(\left\lceil \log_2 \frac{n}{\Delta} \right\rceil\right), \\
\mathrm{C}_{EQcp,\mathcal{A}}(n,\Delta) =& \mathrm{C}_{EncBit} + 3\mathrm{C}_{MULTcp,\mathcal{A}} + 4\mathrm{C}_{COMPcp,\mathcal{A}}\left(\lceil \log_2 \Delta \rceil + 1\right) + \mathrm{C}_{COMPcp,\mathcal{A}}\left(\left\lceil \log_2 \frac{n}{\Delta} \right\rceil\right), \\
\mathrm{C}_{EQcp,\mathcal{B}}(n,\Delta) =& \mathrm{C}_{Encrypt} + 2\mathrm{C}_{EP} + 10\mathrm{C}_{EA} + 3\mathrm{C}_{MULTcp,\mathcal{B}} + 4\mathrm{C}_{COMPcp,\mathcal{B}}\left(\lceil \log_2 \Delta \rceil + 1\right) + \mathrm{C}_{COMPcp,\mathcal{B}}\left(\left\lceil \log_2 \frac{n}{\Delta} \right\rceil\right),
\end{aligned} \tag{7}$$

where $|E|$ represents the number of bits of an encryption (or share). The subindex cm stands for communication complexity, and cp for computational complexity for party $\mathcal{A}$ or $\mathcal{B}$, being C_{MULTxx} the corresponding complexity of the interactive multiplication protocol; C_{EA}, C_{EP} respectively denote the computational complexity of a homomorphic addition and product (by a known scalar) for the used cryptosystem (or secret sharing scheme), $\mathrm{C}_{Encrypt}$ and C_{EncBit} represent the computational complexity for encrypting (sharing) an integer in $\mathcal{Z}_n$ or a bit respectively, and $\mathrm{C}_{COMPxx}(l)$ is defined in [6].

The *fast* protocol has complexity

$$\begin{aligned}
\mathrm{C}_{EQfcm}(n,\Delta) =& |E| + \mathrm{C}_{COMPcm}\left(\left\lceil \log_2 \frac{n}{\Delta} \right\rceil + 1\right), \\
\mathrm{C}_{EQfcp,\mathcal{A}}(n,\Delta) =& \mathrm{C}_{COMPcp,\mathcal{A}}\left(\left\lceil \log_2 \frac{n}{\Delta} \right\rceil + 1\right), \\
\mathrm{C}_{EQfcp,\mathcal{B}}(n,\Delta) =& \mathrm{C}_{Encrypt} + \mathrm{C}_{EP} + 2\mathrm{C}_{EA} + \mathrm{C}_{COMPcp,\mathcal{B}}\left(\left\lceil \log_2 \frac{n}{\Delta} \right\rceil + 1\right).
\end{aligned}$$