Secure Direct and Iterative Protocols for Solving Systems of Linear Equations

Juan Ramón Troncoso-Pastoriza, Pedro Comesaña, and Fernando Pérez-González

University of Vigo
Signal Theory and Communications Department
Vigo 36310, SPAIN,
{troncoso,pcomesan,fperez}@gts.tsc.uvigo.es

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 propose novel privacy-preserving protocols for the solution of Linear Systems of Equations, that improve on previous contributions in terms of security; we also propose secure implementations of iterative algorithms, pointing out the difficulties that arise when dealing with iterative operations on encrypted data, and proposing possible solutions to these shortcomings.

Keywords Privacy, System of Linear Equations, Iterative Methods, Complexity.

1 Introduction

In modern society, digital data about individuals can be found relatively easily in the communication networks, especially the Internet. Although people supports the last decades' advances in digital networks, the sensitiveness of these data motivates the raise of an increasing concern about the public availability of personal data, and the processing performed on them. Focusing on the European case, this concern has been reflected in a series of Directives, dealing with the protection of individuals' personal data ([1, 2]). Directive 95/46/EC deals with *the protection of individuals with regard to the processing of personal data and on the free movement of such data*, where *personal data* means *any information relating to an identified or identifiable natural person*. One of the main mottoes of this Directive is that data processing systems must respect the fundamental rights and freedoms, specially in those aspects concerning the right to privacy.

Leaving aside the legal framework, and turning into the technical support to privacy principles, conventional cryptographic protocols deal with the problem of protecting some private information from an unauthorized third party

that otherwise could modify or have access to the information. In the scenario of secure processing, where the privacy must be preserved not only against a third party, but also against the parties that process the data, secure multiparty computation constructions can be used. Nevertheless, typical multiparty computation protocols become too costly in terms of computation and communication complexity for real-world scenarios.

This is the context in which the emerging field of *Signal Processing in the Encrypted Domain* arose. This discipline tries to address the problem of efficiently processing signals in untrusted environments, where not only the communication channel between parties is insecure, but also the parties that perform the computation are not trusted.

In this *privacy preserving computation* framework, several proposals have been recently issued to implement primitives like secure access to encrypted databases [3, 4], transcoding of an encrypted signal without prior decryption [5], or basic problems such as computing scalar products [6] or linear transforms [7].

Up to now, the efficient protocols presented in the field of signal processing in the encrypted domain have been focused in linear operations, like scalar products, and non-iterative algorithms. Nevertheless, there are many basic algorithms needed for most signal processing applications that are iterative and involve not only scalar products with known values, but also products between two a priori unknown sequences. The lack of these algorithms would suppose missing a powerful and irreplaceable tool that enables almost any signal processing application.

Thus, in this work we cope with this problem, and present efficient and provably secure two-party protocols for solving linear systems of equations and inverting matrices, useful for many applications (e.g. least squares minimization), implementing also iterative algorithms, and calculating the needed cipher size to accommodate a given number of iterations. We also perform a full complexity analysis of the presented protocols.

2 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 have size L and will be represented by lower-case boldface letters, whereas matrices will be represented by upper-case boldface letters. $\boldsymbol{A'} = \{a_{i,j}\}_{r,s}^{t,u}$ represents the submatrix of $\boldsymbol{A}$ of size $(t - r + 1) \times (u - s + 1)$, defined by $a'_{i,j} = a_{i+r-1,j+s-1}$.

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 $[\![\boldsymbol{x}]\!]$ ($[\![\boldsymbol{X}]\!]$).

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 $[\![\boldsymbol{X} \cdot \boldsymbol{b}]\!]$. How these operations will be implemented (homomorphically or through an interactive protocol) will be clear by the context.

Regarding the complexity calculations, the complexity of basic modular operations, like additions (A), products (P) and exponentiations (X) will be denoted by $\mathrm{Comp}_A, \mathrm{Comp}_P, \mathrm{Comp}_X$ respectively, prefixing an E (i.e. EA, EP, EX) when they are performed under encryption. The factor $ct < 1$ will denote the ratio between the size of a clear-text value and that of an encrypted value. Finally, the subscript cm will denote communication complexity, measured in number of sent encryptions, while cp will indicate computational complexity, with an indication of the party whose complexity is represented.

The rest of the Paper is structured as follows: in Section 3 some previous concepts are introduced; Section 4 reviews the existing solutions to the problem of secure solving SLEs; in Section 5 we sketch our protocols and give their measures of complexity. Section 6 evaluates the given complexity measures for a specific construction, gives some examples of use and, for the iterative protocols, plots bounds to representable numbers as a function of the performed iterations, focusing on the trade-off among the three considered parameters: complexity, representability and number of iterations. Finally, Section 7 summarizes the obtained results and sketches the future lines.

3 Secure Computation

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

3.1 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], with an additive homomorphism.

In this work, we do not restrict the used cryptosystem for the presented protocols, as far as it presents an additive homomorphism. For the sake of clarification, and for performing the numerical calculations of Section 6, we use the extension of Paillier encryption given by Damgård and Jurik [10], both in its threshold and non-threshold form; a k out of M threshold public key encryption system [11] is a cryptosystem where the private key is distributed among M parties, and at least k of them are needed for decryption.

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 public integer:

$$E_P[x + y] = E_P[x] \cdot E_P[y] \qquad E_P[x \cdot k] = E_P[x]^k.$$

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

The encryption of a message x is done by picking a random $r \in \mathbb{Z}^*_{n^{s+1}}$ and computing the ciphertext $E_P[x]$ as

$$E_P[x] = g^x r^{n^s} \mod n^{s+1}.$$

For the threshold decryption of $c = E_P[x]$, every party calculates a decryption share with his share of the secret key. These decryption shares are distributed among all the parties, and combined to obtain the wanted decryption. In case of malicious parties, they must also generate a zero-knowledge proof [12] for the correctness of the decryption share. For further details, we refer the reader to [10].

We must also draw attention to the fact that currently there is no practical fully homomorphic cryptosystem, that is, there is no secure cryptosystem that allows for the homomorphic computation of additions and products without restrictions. There has been a recent contribution by Gentry [13], that presents a cryptosystem based on ideal lattices with bootstrappable decryption, and it is shown that it achieves a full homomorphism. Nevertheless, the authors argue that making the scheme practical remains an open problem. Thus, we will adhere to using an additively homomorphic cryptosystem and briefly comment the advantages that an efficient and practical fully homomorphic cryptosystem would provide.

3.2 Secret Sharing

Secret sharing is a technique introduced by Adi Shamir [14], by which a given value (the secret) is divided among several parties, such that the cooperation among a number of these parties is needed in order 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 in order 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, it 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).

3.3 Secure Multiparty Computation

Secure Multiparty Computation was born in 1982 with Yao's Millionaires' problem [15]. 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 the so called *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 [16]. Thus, there exist efficient protocols for binary comparison [16, 17] or Prefix-OR [16]. Traditionally, the search for efficient solutions has led to proposals for changing between integer and binary representation in order to efficiently implement both arithmetic and binary operations; e.g., there are solutions like BITREP protocol [18], that converts Paillier encrypted integers to Paillier encryptions of their corresponding bit representation.

For our constructions, we will use the two-party version of one existing subprotocol for securely performing multiplication; it is sketched in Appendix A.

4 Prior Art

To the best of our knowledge, there was a previous approach to the problem of privately solving linear systems of equations, by Du and Atallah [19]. In that work, the authors presented the problem of solving a linear system with a matrix and an independent vector partitioned between two parties. They presented a solution based on secret sharing, but the privacy that their solution achieves is not total; as later works have shown (cf. Wright and Yang [20]), their protocol for secure multiplication leaks information about the multiplied matrices, and it also relies on a security parameter that largely increases the needed communication in order to achieve a determined level of concealment on the multiplied values. It is also worth mentioning that Cramer and Damgård proposed in [21] a solution to distributed linear algebra problems, coping with finite fields; on the contrary, the present work gives solutions to problems posed in $\mathbb{R}^n$.

In this work, we improve on previous protocols in terms of achieved privacy, practically limiting the leak of information to the inherent leak that disclosing the solution of a SLE produces.

Regarding prior work in the recent field of Signal Processing in the Encrypted Domain, we are not aware of any current solution for securely executing iterative algorithms, nor any study performed on the impact that an iterative implementation has on the range of representable numbers. Thus, we believe that our solution is the first one to be presented for privacy preserving iterative algorithms.

5 Our constructions

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 $\mathcal{A}$ can only encrypt, but $\mathcal{B}$ possesses also the decryption key, and can perform both encryption and decryption. For the problem of solving an SLE $\boldsymbol{A} \cdot \boldsymbol{x} = \boldsymbol{b}$ [22], we will consider that $\mathcal{A}$ owns an encrypted version of the system matrix $[\![\boldsymbol{A}]\!]$, and of the independent vector $[\![\boldsymbol{b}]\!]$. This scenario can be straightforwardly reached from many initial situations, covering all the possible ways of sharing $\boldsymbol{A}$ and $\boldsymbol{b}$ between both parties. For the sake of brevity, we will focus on this initial situation, and obviate the way of reaching it.

The assumption we make about the system consists in $\boldsymbol{A}$ being either a positive definite matrix or a strictly diagonally dominant matrix, in order to guarantee both a solution to the system and the convergence of the studied methods, as will be detailed later. This assumption is not a severe limitation, as many matrices found in statistics calculations fulfill it [22].

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; informally, both parties $\mathcal{A}$ and $\mathcal{B}$ can only get the information leaked from the solution to the system, and no information is leaked from the intermediate steps of the protocols.

As sparsity of the matrices cannot be used as an advantage under encryption, we will focus only on direct methods for solving linear systems of equations (Section 5.1), and we will not cope with methods based in decompositions of the system matrix (like LU or Cholesky decomposition). Furthermore, we will also provide protocols for iterative methods of SLE solving (Section 5.2) and matrix inversion (Section 5.3).

5.1 Direct method: Gaussian elimination

Firstly, we will implement the method of Gaussian elimination, using the secure multiplication protocol (cf. Appendix A) for implementing the needed multiplications. Due to the lack of a division operation under encryption, the obtained result vector is scaled, but the scale factors are stored in a second vector $\boldsymbol{s}$, so that the solution can be recovered after decryption through a component-wise division. The protocol ends with two vectors $\boldsymbol{x}'$ and $\boldsymbol{s}$, being the solution to the system $x_i = \frac{x_i'}{s_i}, \quad i = 1, \ldots, L$.

Let $\boldsymbol{A} \in \mathcal{M}_{L \times L}(\mathbb{Z})$ be a quantized symmetric positive-definite matrix, or a diagonally dominant matrix, and $\boldsymbol{b} \in \mathbb{Z}^L$ be a quantized column vector. The quantization step Δ is such that the absolute value of every quantized element is upper bounded by a constant T.

In our scenario, we will assume that $\mathcal{B}$ knows the decryption key of an additive homomorphic cryptosystem, and both $\mathcal{A}$ and $\mathcal{B}$ can produce encryptions using this cryptosystem; $\mathcal{A}$ possesses the encrypted matrix $[\![\boldsymbol{A}]\!]$ and the encrypted vector of independent terms $[\![\boldsymbol{b}]\!]$. Both parties will engage in an interactive protocol

in order to obtain the solution $\boldsymbol{x}$ to the linear system $\boldsymbol{A} \cdot \boldsymbol{x} = \boldsymbol{b}$. The protocol is sketched next.

Following the Gaussian elimination algorithm, we will call $\boldsymbol{G}^{(0)} = \boldsymbol{G}$ to the concatenation of $\boldsymbol{G} = [\boldsymbol{A}|\boldsymbol{b}]$. The algorithm is executed in $L-1$ steps. At each step k, the matrix $\boldsymbol{G}$ is modified for obtaining an equivalent system $\boldsymbol{G}^{(k)}$ in which the k-th unknown is not present in the last $L-k$ equations.

For the k-th step of the algorithm, the first $k-1$ elements of the $L-k+1$ last rows of $\boldsymbol{G}^{(k-1)}$ are zero; $\mathcal{A}$ owns an encrypted version of the non-zeroed elements of $\boldsymbol{G}^{(k-1)}$. The secure protocol proceeds as follows

1. $\mathcal{A}$ provides randomized encrypted versions of the submatrix $\boldsymbol{C}^{(k)}$ formed by the last $(L-k+2)$ columns of the last $(L-k+1)$ rows of $\boldsymbol{G}^{(k-1)}$;
2. $\mathcal{B}$, through decryption and reencryption, calculates the (randomized) products of the $(L-k) \times (L-k+1)$ matrices $\boldsymbol{D}^{(k)}$ and $\boldsymbol{E}^{(k)}$, defined as $\left[\!\left[d_{j,m}^{(k)}\right]\!\right] = \left[\!\left[c_{1,m+1}^{(k)} \cdot c_{j+1,1}^{(k)}\right]\!\right]$, and $\left[\!\left[e_{i,j}^{(k)}\right]\!\right] = \left[\!\left[c_{1,1}^{(k)} \cdot c_{i+1,j+1}^{(k)}\right]\!\right]$, and sends the randomized encryptions to $\mathcal{A}$.
3. $\mathcal{A}$ derandomizes the received encryptions and, using homomorphic operations, obtain the next iteration of $\boldsymbol{G}$:

$$\left[\!\left[\boldsymbol{G}^{(k)}\right]\!\right] = \left(\begin{array}{c|c} \{\left[\!\left[g_{i,m}^{(k-1)}\right]\!\right]\}_{(1,1)}^{(k,L+1)} & \\ \hline \boldsymbol{0}_{L-k,k} & \left[\!\left[\boldsymbol{F}^{(k)}\right]\!\right] \end{array}\right),$$

where $\left[\!\left[\boldsymbol{F}^{(k)}\right]\!\right]$ is an $(L-k) \times (L-k+1)$ matrix with elements $\left[\!\left[f_{i,m}^{(k)}\right]\!\right] = \left[\!\left[e_{i,m}^{(k)}\right]\!\right] - \left[\!\left[d_{i,m}^{(k)}\right]\!\right]$.

After $L-1$ iterations, $\mathcal{A}$ has an encrypted upper triangular matrix appended to an encrypted vector, $\left[\!\left[\boldsymbol{G}^{(L-1)}\right]\!\right]$, that constitute a system with the same solution as the original one.

In order to solve the system, both parties initiate the process of back substitution under encryption, consisting in L iterations: in each iteration, an element of the vector $\boldsymbol{x}'$ and the corresponding element of the scale vector $\boldsymbol{s}$ are obtained. As they will be revealed as the output, and they are needed in order to calculate the subsequent elements of $\boldsymbol{x}'$, they can be decrypted before the next iteration in order to lower the complexity by reducing the number of the needed multiplication protocols. For the first step:

1. $\mathcal{A}$ sends $\{\left[\!\left[g_{i,i}^{(L-1)}\right]\!\right]\}_{i=1}^{L}$ and $\left[\!\left[g_{L,L+1}^{(L-1)}\right]\!\right]$.
2. $\mathcal{B}$ obtains, through decryption, the scaling vector $\boldsymbol{s}$, with $s_i = \prod_{l=i}^{L} g_{l,l}^{(L-1)}$, and the value $x'_L = g_{L,L+1}^{(L-1)}$, and sends them back to $\mathcal{A}$.

In each subsequent k-th step, $\mathcal{A}$ calculates, using homomorphic operations:

$$\left[\!\left[x'_{L-k+1}\right]\!\right] = \left[\!\left[g_{L-k+1,L+1}^{(L-1)}\right]\!\right] \cdot s_{L-k+2} - \sum_{l=L-k+2}^{L} \left[\!\left[g_{L-k+1,l}^{(L-1)}\right]\!\right] \cdot x'_l \frac{s_l}{s_{L-k+2}},$$

and sends $[\![x'_{L-k+1}]\!]$ to $\mathcal{B}$ to obtain its decryption.

With the proposed protocol, we are not disclosing any element of the original matrix $\boldsymbol{A}$ nor of the independent terms vector $\boldsymbol{b}$. Furthermore, every step of the protocol can be proven secure with semihonest parties, due to the semantical security of the underlying homomorphic cryptosystem, the security of the used multiplication protocols, and the fact that all the unencrypted values (besides the result and the scaling vector) that each party can access are random and uncorrelated. Nevertheless, the scaling vector reveals the diagonal of the upper triangular matrix of an equivalent system, which gives information about the eigenvalues of the original matrix. Nevertheless, this information affects L scaled elements out of $\frac{L(L+1)}{2}$.

It must be noted that having the values of the principal diagonal of the upper-triangular matrix of the equivalent system yields the possibility of calculating its condition number, or at least, its bound

$$\kappa(\boldsymbol{U}) \geq \frac{\max_i(|u_{ii}|)}{\min_i(|u_{ii}|)}.$$

Thus, this disclosure constitutes a clear advantage in terms of conditioning and efficiency: before executing the back substitution protocol, the rows of $\boldsymbol{G}^{(L)}$ can be multiplied by appropriate factors in order to lower the condition number and minimize error propagation due to working with a fixed point precision. Also, the vector of multiplicative factors s_i can be adequately quantized in the clear to achieve this same goal.

As a last remark, this protocol does not limit the number N of SLEs sharing the same system matrix $\boldsymbol{A}$ and with different independent term vectors $\boldsymbol{b}_i$ that can be solved in parallel; all the vectors $\boldsymbol{b}_i$ can be appended to the system matrix, forming a $L \times (L+N)$ matrix $\boldsymbol{G}_{ext}$ and in each step of the previous protocol, the operations that must be performed on the last column of $\boldsymbol{G}^{(k)}$ will be replicated for the last N columns of $\boldsymbol{G}^{(k)}_{ext}$.

Complexity When solving one system $\boldsymbol{A} \cdot \boldsymbol{x} = \boldsymbol{b}$, the Gaussian Elimination (GE) protocol is performed in $(L-1)$ rounds of communication, with total complexity

$$\begin{aligned}
\text{Comp}_{cmGE} &= (L^3 + L^2 - 2) \\
\text{Comp}_{cpGE,A} &= \frac{1}{3}\left(L^3 + 3L^2 + 2L - 6\right)\text{Comp}_{Encrypt} + \\
&\quad \frac{1}{3}\left(2L^3 + 3L^2 + L - 6\right)\text{Comp}_{EA} \\
\text{Comp}_{cpGE,B} &= \frac{1}{3}\left(L^3 + 3L^2 + 2L - 6\right)\text{Comp}_{Decrypt} + \\
&\quad \frac{2}{3}\left(L^3 - L\right)(\text{Comp}_{Encrypt} + \text{Comp}_P).
\end{aligned}$$

The protocol of Back Substitution (BS) is performed in L rounds of communication, with total complexity

$$\begin{aligned}\text{Comp}_{cmBS} &= 2L \cdot (1 + ct)\\ \text{Comp}_{cpBS,A} &= \frac{1}{2}(L^2 + L - 2)\text{Comp}_{EP} + \frac{1}{2}(L^2 - L)\text{Comp}_{EA}\\ \text{Comp}_{cpBS,B} &= 2L\text{Comp}_{Decrypt}.\end{aligned}$$

Representable numbers We have assumed that the coefficients of the system matrix $\boldsymbol{A}$ are quantized versions of the real-valued coefficients, with a quantization step Δ. Furthermore, the absolute value of the quantized coefficients is bounded by an integer $T > 0$. Then, it is possible to estimate the value of T needed to fit all the performed operations inside a cipher that can represent integers in the range $[0, n)$ without rounding problems.

For the first part of the protocol (the Gaussian elimination), each iteration multiplies two numbers that were obtained in the previous iteration and adds them up, so the previous bound gets squared and doubled:

$$|t_1|, |t_2|, |t_3|, |t_4| < T \Rightarrow |t_1 \cdot t_2 - t_3 \cdot t_4| < 2T^2.$$

Then it is straightforward to conclude that all the elements of the k-th row of the resulting $\boldsymbol{G}^{(L-1)}$ will be bounded by $(2^{2^{k-1}-1})T^{2^k}$, and will constitute the representation of their real-valued equivalents, quantized by $\Delta^{2^{k-1}}$. Thus, the cipher must be such that $n > (2^{2^{k-1}-1})T^{2^k}$ in order to fit all the numbers involved in this protocol. This means that the bit size of the modulus of the cipher must grow exponentially with the dimensionality of the system, what gives a poor scalability.

For the second part of the protocol, after the diagonal elements are disclosed, they can be requantized in order to make them relative to the lowest scale and lower the bit-size requirements of the cipher; but in the worst case, without requantizing the scale factors, the largest number present after running the whole protocol will be $2^{2^L-L-1}T^{2^{1+L}-4}$. That will also constrain the size of the cipher.

5.2 Iterative methods: Jacobi's Method

The general form of *stationary iterative methods* for solving SLEs is

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{M} \cdot \boldsymbol{x}^{(k)} + \boldsymbol{c}.$$

Jacobi's method is a particular case of stationary iterative methods, where the system matrix is decomposed into $\boldsymbol{A} = \boldsymbol{D}(\boldsymbol{L} + \boldsymbol{I} + \boldsymbol{U})$, a diagonal matrix $\boldsymbol{D}$, a lower triangular matrix $\boldsymbol{L}$ and an upper triangular matrix $\boldsymbol{U}$, having both $\boldsymbol{L}$ and $\boldsymbol{U}$ zeros in their principal diagonals. Then, $\boldsymbol{M} = -(\boldsymbol{L} + \boldsymbol{U})$ and $\boldsymbol{c} = \boldsymbol{D}^{-1}\boldsymbol{b}$. As divisions are not supported homomorphically, the previous iteration cannot be implemented directly. Thus, the division is simulated by multiplying each row

of $\boldsymbol{A}$ by the diagonal elements of the remaining rows, what results in multiplying the matrix $\boldsymbol{M}$ of Jacobi's method by a scalar factor $\gamma = \left(\prod_{i=1}^{L} a_{ii}\right)$.

$$\boldsymbol{A}' = -\gamma \boldsymbol{D}^{-1} \cdot (\boldsymbol{A} - \boldsymbol{D}) = \gamma \boldsymbol{M}.$$

The factor γ will be propagated at every iteration of the algorithm:

$$\gamma^k \boldsymbol{x}^{(k)} = -\gamma \boldsymbol{D}^{-1} (\boldsymbol{A} - \boldsymbol{D}) \cdot \gamma^{k-1} \boldsymbol{x}^{(k-1)} + \gamma^k \boldsymbol{D}^{-1} \boldsymbol{b}.$$

Let us assume that $\mathcal{B}$ can decrypt and both $\mathcal{A}$ and $\mathcal{B}$ can encrypt with an additive homomorphic scheme, and that $\mathcal{A}$ owns encryptions of $[\![\boldsymbol{A}]\!]$ and $[\![\boldsymbol{b}]\!]$ with this homomorphic system. In order to allow for efficient computation, the following protocol is executed:

1. $\mathcal{A}$ can blind the principal diagonal of $[\![\boldsymbol{A}]\!]$ and send it to $\mathcal{B}$.
2. $\mathcal{B}$ decrypts it, both parties ending up with additive shares of the diagonal elements $\{a_{ii}\}$.
3. With this shares, both parties can securely compute shares of the diagonal matrix $\left(\gamma \boldsymbol{D}^{-1}\right)_{jj} = \prod_{\substack{i=1 \\ i \neq j}}^{L} a_{ii}$, through $\lceil \log_2(L-1) \rceil$ rounds of parallel secure multiplication protocols. They can also calculate the value of γ, and disclose it for use in the following steps of the protocol.
4. $\mathcal{A}$ can then calculate the encryption of $[\![\gamma \boldsymbol{M}]\!] = \left[\!\left[-\gamma \boldsymbol{D}^{-1}\right]\!\right] \cdot [\![\boldsymbol{A} - \boldsymbol{D}]\!]$ and $[\![\gamma \boldsymbol{c}]\!] = \left[\!\left[\gamma \boldsymbol{D}^{-1}\right]\!\right] \cdot [\![\boldsymbol{b}]\!]$, invoking the secure multiplication protocol.
5. Then, $\mathcal{A}$ sends $\mathcal{B}$ a blinded and encrypted version of $[\![\gamma \boldsymbol{M}]\!]$, that $\mathcal{B}$ decrypts for use in the following iterations.

After these initial steps, for the *first iteration* of the secure protocol both parties agree in an initial vector $\boldsymbol{x}^{(0)}$ and $\mathcal{A}$ calculates, through homomorphic additions and multiplications, the encryption of $\left[\!\left[\gamma \boldsymbol{x}^{(1)}\right]\!\right] = [\![\gamma \boldsymbol{M}]\!] \cdot \boldsymbol{x}^{(0)} + [\![\gamma \boldsymbol{c}]\!]$.

For *each subsequent iteration*, $\mathcal{A}$ calculates the encryption of $\gamma \cdot \left[\!\left[\gamma^{k-1} \boldsymbol{c}\right]\!\right]$, and then both parties use the secure multiplication protocol of Appendix A and homomorphic additions in order to obtain the vector for the following step

$$\left[\!\left[\gamma^k \boldsymbol{x}^{(k)}\right]\!\right] = [\![\gamma \boldsymbol{M}]\!] \cdot \left[\!\left[\gamma^{k-1} \boldsymbol{x}^{(k-1)}\right]\!\right] + \gamma \cdot \left[\!\left[\gamma^{k-1} \boldsymbol{c}\right]\!\right].$$

It must be noted that the matrix $[\![\gamma \boldsymbol{M}]\!]$ does not have to be communicated at each iteration, as its blinded version was stored by $\boldsymbol{B}$ at the initial step. Thus, only two vectors per iteration are sent between $\mathcal{A}$ and $\mathcal{B}$.

After each iteration, the factor γ multiplies the result; thus, after a number of steps, the cipher will not be able to accommodate the scaled number, and the protocol will have to stop. This is studied in more depth in Section 6. It must be noted that the accumulated factor is not only γ, but also the quantization step Δ used for the initial quantization of the coefficients of both the system matrix $\boldsymbol{A}$ and the vector $\boldsymbol{b}$ in order to make them integers so that they can be encrypted. This factor must also be taken into account every time γ multiplies

vector $\boldsymbol{c}$, so that the homomorphically added vectors be quantized with the same scaling factor.

Lastly, each step of the protocol can be proven secure with semihonest parties, due to the semantic security of the underlying cryptosystem, the security of the multiplication protocol, and the fact that the unencrypted values that each party sees are random and uncorrelated.

Complexity The complexity of the initial part (Jacobi Initial, JI) of the protocol is

$$\begin{aligned}
\mathrm{Comp}_{cmJI} &= 3L^2 + 2L\lceil \log_2(L-1) \rceil - 3L + 5 + ct \\
\mathrm{Comp}_{cpJI,A} &= (5L^2 + 4L\lceil \log_2(L-1) \rceil - 5L + 8)\mathrm{Comp}_{EA} + \\
&\quad (L^2 + L\lceil \log_2(L-1) \rceil - L + 2)2\mathrm{Comp}_{EP} \\
\mathrm{Comp}_{cpJI,B} &= (L^2 + L\lceil \log_2(L-1) \rceil - L + 2)\left(\mathrm{Comp}_{Decrypt} + \mathrm{Comp}_P + \right. \\
&\quad \left. \mathrm{Comp}_{Encrypt}\right) + (L^2 - L + 1)\mathrm{Comp}_{Decrypt}.
\end{aligned}$$

The first iteration ($J1$) does not involve any interaction, and $\mathcal{A}$ incurs in a computational complexity of $\mathrm{Comp}_{cpJ1,A} = L^2(\mathrm{Comp}_{EP} + \mathrm{Comp}_{EA})$.

The complexity of each of the subsequent iterations of this protocol (J) is the following

$$\begin{aligned}
\mathrm{Comp}_{cmJ} &= 2L \\
\mathrm{Comp}_{cpJ,A} &= (3L^2 - 2L)\mathrm{Comp}_{EA} + (2L^2 - L)\mathrm{Comp}_{EP} + L\mathrm{Comp}_{EA} \\
\mathrm{Comp}_{cpJ,B} &= L\left(\mathrm{Comp}_{Decrypt} + \mathrm{Comp}_{Encrypt}\right) + (L^2 - L)\mathrm{Comp}_P + \\
&\quad (L^2 - 2L)\mathrm{Comp}_A.
\end{aligned}$$

After a number of iterations, either the solution can be disclosed, or an error metric can be obtained to determine whether convergence has been achieved. While the choice of this error metric is arbitrary, one possibility is to homomorphically subtract $[\![\boldsymbol{x}^{(k)}]\!] - [\![\boldsymbol{x}^{(k-1)}]\!]$, and either decrypt the result or perform L parallel encrypted comparisons with a predetermined threshold.

Representable numbers As for the direct method, we have assumed that the coefficients of the system matrix $\boldsymbol{A}$ are quantized versions of the real-valued coefficients, with a quantization step Δ, such that their quantized absolute value is bounded by an integer $T > 0$.

For the first part of the protocol, where the factor γ and the matrix $\gamma\boldsymbol{D}^{-1}$ are calculated, γ is the highest number that the system will have to represent, and it is bounded by T^L; the bound for the elements of $\gamma\boldsymbol{D}^{-1}$ is T^{L-1}. Furthermore, as γ is disclosed in the following step, it can constitute a more accurate bound to the encrypted coefficients of $\gamma\boldsymbol{D}^{-1}$. A bound for the absolute value of the coefficients of $\gamma\boldsymbol{M}$ and of $\gamma\boldsymbol{c}$ is $\min(T^L, \gamma T)$.

Lastly, the bound to which the elements of the first vector $\boldsymbol{x}^{(1)}$ are subject is $L \cdot T^{L+1}$.

In each iteration, the previous bound is multiplied by $L \cdot T^L$, meaning that the bound for the elements of the k-th iteration is $L^k \cdot T^{k \cdot L+1}$, i.e., the needed bit-size of the cipher is linear both in the dimension of the system and in the maximum number of iterations that can be performed without errors. Furthermore, the quantization step of the elements of $\boldsymbol{x}^{(k)}$ will be Δ^{kL}.

Convergence of the algorithm When dealing with iterative algorithms like Jacobi's, it is necessary to determine whether the algorithm can converge or not before applying the algorithm. In the general case of *stationary iterative methods*, the necessary and sufficient condition for their convergence with an arbitrary initial vector $\boldsymbol{x}^{(0)}$ is that $\max_i |\lambda_i(\boldsymbol{M})| < 1$, where $\lambda_i(\boldsymbol{M})$ are the eigenvalues of $\boldsymbol{M}$. For Jacobi's method, $\boldsymbol{M} = -\boldsymbol{D}^{-1} \cdot (\boldsymbol{A} - \boldsymbol{D})$. Let us assume that $\boldsymbol{A}$ is a strictly diagonally dominant matrix with bounded coefficients $|a_{ij}| \leq T$. By Ostrowski's theorem [23], the eigenvalues of $\boldsymbol{M}$ are located in the union of L discs

$$\mathcal{L}_1 \triangleq \bigcup_{i=1}^{L} \{z \in \mathbb{C} : |z - m_{ii}| \leq \min\{R_i, C_i\}\},$$

where $m_{ii} = 0$, $m_{ij} = \frac{a_{ij}}{a_{ii}}, i \neq j$, and

$$R_i = \sum_{j=1, j \neq i}^{L} |m_{ij}|,$$

$$C_i = \sum_{j=1, j \neq i}^{L} |m_{ji}|.$$

As $\boldsymbol{A}$ is strictly diagonally dominant, $\sum_{j=1, j \neq i}^{L} |a_{ij}| < |a_{ii}| \Rightarrow R_i < 1$. Thus, it is possible to bound the moduli of the eigenvalues of $\boldsymbol{M}$ as

$$|\lambda_i(\boldsymbol{M})| < 1.$$

Then, Jacobi method always converges for strictly diagonally dominant matrices, and the test of convergence is not needed.

5.3 Matrix inversion through iterative methods

There are cases in which, instead of or additionally to solving a SLE, the inverse of the system matrix is also needed, like the case of regression analysis in statistics. For these applications, the system matrix $\boldsymbol{A}$ must be inverted. As the direct method (through Cramer's rule) is computationally too expensive, we will provide a secure protocol for performing the execution of an iterative method, namely Newton's method. One iteration of this method has the following expression

$$\boldsymbol{X}^{(k)} = \boldsymbol{X}^{(k-1)} \cdot \left(2\boldsymbol{I} - \boldsymbol{A}\boldsymbol{X}^{(k-1)}\right),$$

where $\boldsymbol{X}^{(k)}$ will converge to $\boldsymbol{A}^{-1}$.

The secure protocol for Newton's method will execute an *initial iteration* with an agreed initial value $\boldsymbol{X}^{(0)}$, performed uniquely with homomorphic operations. Then, the *following iterations* make use of the secure multiplication protocol (cf. Appendix A) and homomorphic sums. Each iteration needs two rounds of communication:

1. The first one to calculate $[\![\boldsymbol{Q}^{(k)}]\!] = [\![\boldsymbol{A}]\!] \cdot [\![\boldsymbol{X}^{(k-1)}]\!]$,
2. the second one to calculate $[\![\boldsymbol{X}^{(k)}]\!] = [\![\boldsymbol{X}^{(k-1)}]\!] \cdot (2\boldsymbol{I} - [\![\boldsymbol{Q}^{(k)}]\!])$.

As with any iterative method, the result gets multiplied after each iteration by the quantization step of the used integers, so after a sufficiently high number of iterations, as with Jacobi's method, the cipher will not be able to accommodate the scaled numbers, and the protocol will stop (cf. Section 6).

Lastly, the protocol is provably secure with semihonest parties due to the semantic security of the cryptosystem, and the security of the sequentially composed multiplication protocols.

Complexity The first step involves only one round of interaction, and its complexity is given by

$$\begin{aligned}
\mathrm{Comp}_{cmNEWI} &= L^2 \\
\mathrm{Comp}_{cpNEWI,A} &= 2L^3\mathrm{Comp}_{EP} + (2L^3 - 2L^2 + L)\mathrm{Comp}_{EA} \\
\mathrm{Comp}_{cpNEWI,B} &= 0.
\end{aligned}$$

The complexity of each of the subsequent iterations of this protocol is the following

$$\begin{aligned}
\mathrm{Comp}_{cmNEW} &= 2\mathrm{Comp}_{cmMULT}(L, L, L) \\
\mathrm{Comp}_{cpNEW,A} &= 2\mathrm{Comp}_{cpMULT,A}(L, L, L) + L\mathrm{Comp}_{EA} \\
\mathrm{Comp}_{cpNEW,B} &= 2\mathrm{Comp}_{cpMULT,B}(L, L, L).
\end{aligned}$$

Representable numbers Let us assume that the elements of the matrix $\boldsymbol{A}$ are quantized with a quantization step Δ, and their absolute value is bounded by $T > 0$. Then, the elements of matrix resulting from the first iteration of the protocol are bounded by $L^2T^3 + 2T$. For each of the next iterations, the bound $T^{(k-1)}$ is updated as $T^{(k)} = \left(T^{(k-1)}\right)^2 \cdot T \cdot L^2 + 2 \cdot K$. Thus, the order of the bound after m iterations is $O\left(T^{2^{m+1}-1} \cdot L^{2^{m+1}-2}\right)$, i.e. the bit-size of the cipher is exponential in the number of iterations, like for the direct algorithm of Section 5.1.

Convergence of the algorithm The convergence of Newton's method is assured whenever the initial matrix $\boldsymbol{X}^{(0)}$ satisfies $||\boldsymbol{A}\boldsymbol{X}^{(0)} - \boldsymbol{I}|| < 1$. As the initial vector is chosen by both parties, it can be such that this condition is fulfilled,

given the bounds to the elements of $\boldsymbol{A}$ and the bounds to the eigenvalues obtained by the application of Ostrowski's theorem. This way, as for Jacobi's method, it would be unnecessary to check for convergence through an additional interactive protocol.

6 Practical Implementation

In this section, we study a practical implementation of the protocols that we have proposed, and comment on the obtained results. For this purpose, we have chosen Damgård-Jurik [10] extension of Paillier cryptosystem, due to its flexibility for fitting larger plaintexts with a constant expansion ratio. With the complexity calculations shown in the previous section, we will exemplify the figures to which the presented protocols lead, with different parameters.

Firstly, we evaluate the needed size of the cleartext group in order to safely fit all the involved numbers without errors in their representation. Figure 1 shows the bit-size of the plaintext group when coefficients are bounded by 2^{32} and for an SLE with $L = 10$ equations (10×10 system matrix), which are reasonable sizes for common applications. The direct method needs plaintexts of about 2^{15} bits for getting the solution of the system; this is a reasonable figure, taking into account that the output will be directly the solution of the system. On the other hand, the two studied iterative systems have a very different behavior, as anticipated by the calculations of Section 5. While the needed size of the plaintext in the protocol implementing Newton's method grows exponentially with the number of iterations, needing more than 2^{16} bits to fit the represented numbers after 10 iterations, Jacobi's method is far more conservative, needing less than 2^{13} bits for the same number of iterations, and with a linear growth.

Figure 2 shows also the size of the plaintext, varying the dimensionality of the problem; for the iterative algorithms, the number of performed iterations is fixed at 5. This time the direct protocol shows its exponential dependence on the dimensionality of the problem, while the protocol for Jacobi's method is linear, and Newton's algorithm logarithmic. This is a factor that is worth considering when inverting matrices with a high dimensionality.

Regarding the complexity of the developed protocols in terms of communication and computation given a maximum plaintext size, Figure 3 plots the communication complexity measured in bits for the three protocols when solving a system with 5 unknowns, and varying the parameter s of Damgård-Jurik cryptosystem, that gives a plaintext size of $s \cdot m$, where m has been fixed here to 1024 bits. For this system, the minimum s accepted by the direct protocol is $s = 2$. For the iterative protocols, the complexity is calculated for the maximum number of iterations that the size of the cipher can correctly fit. This quantity is indicated in Table 1.

While the communication complexity of the three protocols is approximately linear on s, the protocol for matrix inversion needs much more communication, with a number of iterations limited by the maximum size of the plaintext. On the other hand, the protocol that implements Jacobi's method is much more

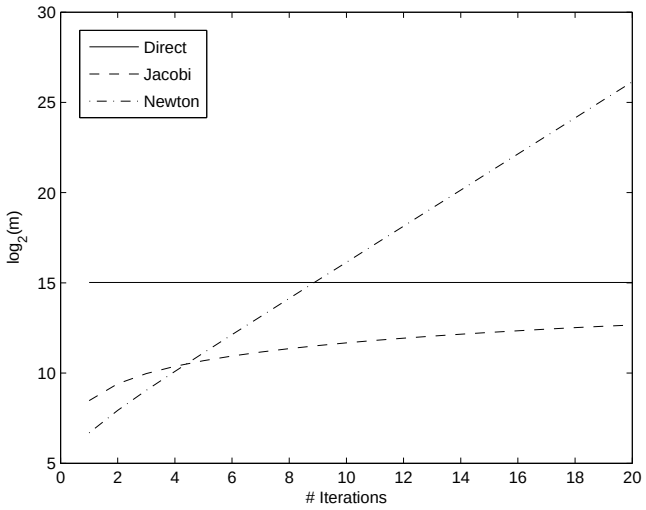

Fig. 1. Logarithm of the bit-size of the plaintext group as a function of the performed iterations for $T = 2^{32}$ and $L = 10$ dimensions.

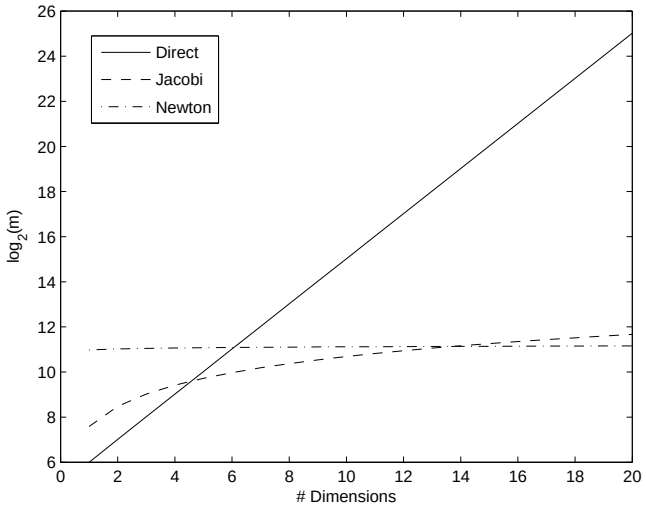

Fig. 2. Logarithm of the bit-size of the plaintext group as a function of the number of dimensions, for $T = 2^{32}$ and 5 iterations.

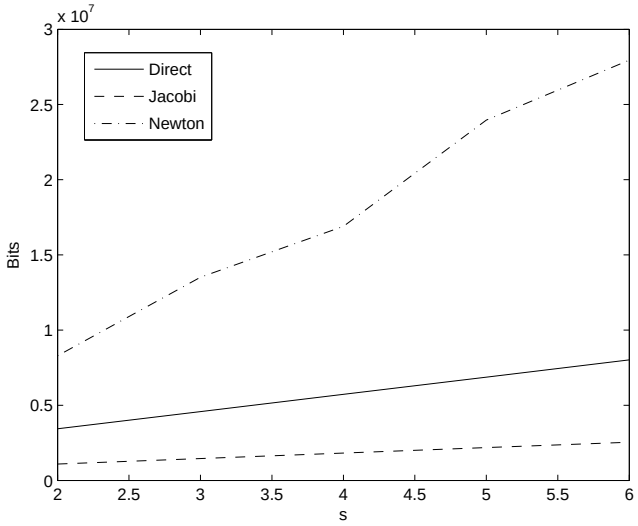

Fig. 3. Communication complexity of the presented protocols as a function of s, with $T = 2^{32}$ and $L = 5$ dimensions.

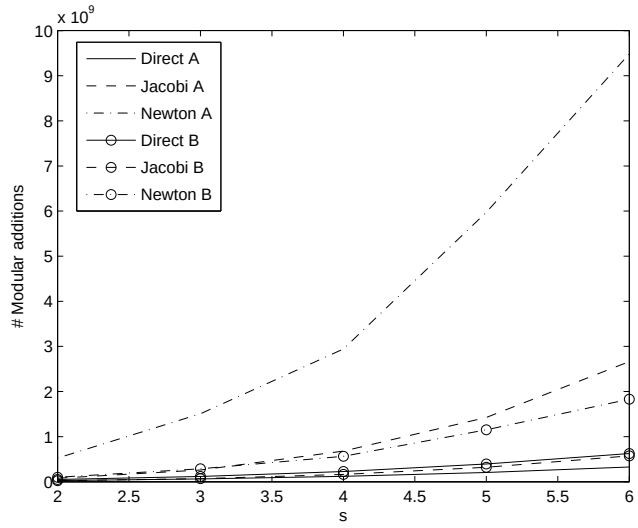

Fig. 4. Computational complexity of the presented protocols as a function of s, with $T = 2^{32}$ and $L = 5$ dimensions.

Table 1. Number of allowed iterations as a function of s, with $T = 2^{32}$ and $L = 5$ dimensions.

s	2	3	4	5	6
Jacobi[iters]	13	19	26	32	38
Newton[iters]	4	5	5	6	6

efficient, as it can perform a much larger number of iterations within the same plaintext size, while incurring in a lower complexity.

The same behavior can be observed in terms of computational complexity (Figure 4), that is approximately quadratic on s for the three protocols, but the multiplicative constants are much larger for Newton's protocol than for the other two. As a function of the number of dimensions, the protocol implementing Jacobi's method is also much better behaved than the other two methods, but still needs a large plaintext size when a high number of iterations must be performed. We have not included more plots illustrating this behavior due to space limitations.

Summarizing, the needed bit size for the three protocols is relatively high, and for the case of the protocol for Newton's method it grows exponentially with the number of performed iterations. Jacobi's is far more efficient, as it can accommodate a much larger number of iterations using the same maximum plaintext size. The complexity of the protocol for Jacobi's method is also lower than the other two methods for a sufficiently high number of dimensions.

Nevertheless, in order to perform an arbitrary number of iterations, and to lower the complexity of the three protocols, it would be desirable to have a means for renewing the cipher with a lower scale factor. It must be noted that, even when having a full homomorphic cryptosystem, this problem cannot be avoided. The full homomorphism would allow for performing all the operations without interaction, considerably lowering the communication complexity, as well as the computational complexity (depending on how the homomorphic operations must be performed). Nevertheless, with a fully homomorphic cryptosystem the growth of the ciphered numbers would be also unavoidable, and the method for requantization would also be needed.

7 Conclusions and Further Work

In this work we have proposed new privacy-preserving protocols for solving systems of linear equations (SLEs), making use of homomorphic computation and secret sharing. We have implemented a direct method (Gaussian elimination), as well as iterative methods for solving SLEs (Jacobi's method) and matrix inversion (Newton's method). These protocols are secure with semi-honest parties, and, to the best of our knowledge, they are the first iterative methods under encryption proposed up to date.

There are some difficulties in the implementation of an iterative method that have been pointed out in the present work, namely the growth of the ciphered

numbers and their change in quantization scale. As a continuation of this work, we are working in a protocol for tackling these problems, that will be presented in a future contribution.

8 ACKNOWLEDGMENTS

This work was partially funded by Xunta de Galicia under Projects 07TIC012322PR (FACTICA), 2007/149 (REGACOM), 2006/150 ("Consolidation of Research Units"), by the Spanish Ministry of Science and Innovation under projects COMONSENS (ref. CSD2008-00010) of the CONSOLIDER-INGENIO 2010 Program, and SPROACTIVE (ref. TEC2007-68094-C02-01/TCM) and the FPU grant Ref. AP2006-02580.

References

1. : Directive 95/46/EC of the European Parliament and of the Council. Official Journal L 281, 23/11/1995 P. 0031 - 0050 (October 1995)
2. : Directive 2002/58/EC of the European Parliament and of the Council. Official Journal L 201, 31/07/2002 P. 0037 - 0047 (July 2002)
3. Brinkman, R., Doumen, J.M., Jonker, W.: Using secret sharing for searching in encrypted data. In: Workshop on Secure Data Management in a Connected World (SDM 2004). Volume 3178 of Lecture Notes in Computer Science., Springer-Verlag (2004) 18–27
4. Song, D.X., Wagner, D., Perrig, A.: Practical techniques for searches on encrypted data. In: Proceedings of the 2000 IEEE symposium on Security and Privacy. (2000) 44–55
5. Johnson, M., Ishwar, P., Prabhakaran, V., Schonberg, D., Ramchandran, K.: On compressing encrypted data. IEEE Transactions on Signal Processing **52**(10) (October 2004) 2992–3006
6. Goethals, B., Laur, S., Lipmaa, H., Mielikäinen, T.: On private scalar product computation for privacy-preserving data mining. In Park, C., Chee, S., eds.: 7th Annual International Conference in Information Security and Cryptology (ICISC 2004). Volume 3506 of Lecture Notes in Computer Science., Seoul, Korea, Springer (December 2004) 104–120
7. Bianchi, T., Piva, A., Barni, M.: On the implementation of the discrete fourier transform in the encrypted domain. IEEE Transactions on Information Forensics and Security **4**(1) (2009) 86–97
8. Rivest, R., Adleman, L., Dertouzos, M.: On data banks and privacy homomorphisms. In: Foundations of Secure Computation, Academic Press (1978) 169–177
9. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Advances in Cryptology - EUROCRYPT 1999. Volume 1592 of Lecture Notes in Computer Science., Springer (1999) 223–238
10. Damgård, I., Jurik, M.: A generalisation, a simplification and some applications of paillier's probabilistic public-key system. In Kim, K., ed.: Public Key Crytography 2001. Volume 1992 of Lecture Notes in Computer Science., Cheju Island, Korea, Springer (February 2001) 119–136
11. Desmedt, Y., Frankel, Y.: Threshold cryptosystems. In: Proceedings of Crypto 1989, Santa Barbara, California, USA (1989) 307–315

12. Shoup, V.: Practical threshold signatures. In: Advances in cryptology - EUROCRYPT 2000. Volume 1807 of Lecture Notes in Computer Science., Springer (2000) 207–220
13. Gentry, C.: 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) 169–178
14. Shamir, A.: How to share a secret. Communications of the ACM **22**(11) (1979) 612–613
15. Yao, A.C.: Protocols for secure computations. In: Proceedings of the IEEE Symposium on Foundations of Computer Science. (1982) 160–164
16. Damgård, I., Fitzi, M., Kiltz, E., Nielsen, J.B., Toft, T.: Unconditionally secure constant-rounds multi-party computation for equality, comparison, bits and exponentiation. In: Proceedings of the third Theory of Cryptography Conference, TCC 2006. Volume 3876 of Lecture Notes in Computer Science., Springer-Verlag (2006) 285–304
17. Nishide, T., Ohta, K.: Constant-round multiparty computation for interval test, equality test, and comparison. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences **E90-A**(5) (May 2007) 960–968
18. Schoenmakers, B., Tulys, P.: Efficient binary conversion for paillier encrypted values. In: Advances in Cryptology - EUROCRYPT 2006. Volume 4004 of Lecture Notes in Computer Science., Springer (2006) 522–537
19. Du, W., Atallah, M.J.: Privacy-preserving cooperative scientific computations. In: Proceedings of the 14th IEEE Computer Security Foundations Workshop, Nova Scotia, Canada (June 2001) 273–282
20. Wright, R., Yang, Z.: Privacy preserving bayesian network structure computation on distributed heterogeneous data. In: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, Seattle, WA, USA, ACM Press (2004) 713–718
21. Cramer, R., Damgård, I.: Secure distributed linear algebra in a constant number of rounds. In: 21st Annual International Cryptology Conference on Advances in Cryptology. Volume 2139 of Lecture Notes in Computer Science., Springer (2001) 119–136
22. Dahlquist, G., Björck, Å.: Numerical methods. Dover Publications (2003)
23. Horn, R.A., Johnson, C.R.: Matrix Analysis. Cambridge University Press (1985)
24. Cramer, R., Damgård, I., Nielsen, J.: Multiparty computation from threshold homomorphic encryption. In: Advances in Cryptology EUROCRYPT'01. Volume 2045 of LNCS., Springer-Verlag (October 2001) 280–300

A Secure Multiplication Protocol

In order to multiply two encrypted matrices, as there is no multiplication operation in an additively homomorphic cryptosystem, it is necessary to execute an interactive protocol in order to perform each product. The generic protocol for secure multiplication gates has been known since [24]. In this work, we use a variant for non threshold encryption, that is included in this appendix for clarification and completeness. Let us assume that there exists an additively homomorphic cryptosystem with plaintext in $\mathbb{Z}_n$ such that $\mathcal{B}$ can decrypt and both $\mathcal{A}$ and $\mathcal{B}$ can encrypt. $\mathcal{A}$ owns two encrypted scalars $[\![x_1]\!]$ and $[\![x_2]\!]$ and wants to

multiply them under encryption. In order to do that, $\mathcal{A}$ generates two random values $r_1, r_2 \in \mathbb{Z}_n$, and uses them to blind both numbers, using homomorphic modulo-n sum obtaining $[\![z_1]\!] = [\![x_1]\!] + r_1 \mod n$, and $[\![z_2]\!] = [\![x_2]\!] + r_2 \mod n$, and sends them to $\mathcal{B}$.

Due to his decryption capabilities, $\mathcal{B}$ can obtain z_1 and z_2 in the clear, multiply them, and reencrypt the result $[\![z_1 \cdot z_2]\!]$. $\mathcal{B}$ sends this encrypted product to $\mathcal{A}$, who, through homomorphic sums, can obtain the desired result, as

$$[\![x_1 \cdot x_2]\!] = [\![z_1 \cdot z_2]\!] - r_1 [\![x_2]\!] - r_2 [\![x_1]\!] - r_1 r_2.$$

In the scenario of a threshold homomorphic cryptosystem, the procedure is analogous, with the exception that the random values must be generated by both parties [16].

For the case of the product of an $L \times M$ matrix and an scalar, the protocol is exactly the same as the scalar-scalar case, with $L \times M$ scalar products in parallel.

For the case of matrix-matrix product, the extension is also straightforward, as all the scalar products are performed using the scalar-scalar product protocol in parallel, with only one randomization per matrix coefficient, and the remaining operations are sums, that can be performed homomorphically. Obviously, in order to minimize the computation and communication complexity, $\mathcal{A}$ may let $\mathcal{B}$ perform all the partial additions that $\mathcal{B}$ can do in the clear and $\mathcal{A}$ would need to do homomorphically.

Neglecting the complexity of the random number generation algorithms, the complexity of the whole protocol, when multiplying an $L \times M$ matrix and an $M \times N$ matrix is

$$\begin{aligned}
\mathrm{Comp}_{cmMULT}(L, M, N) &= M \cdot (L + N) + L \cdot N \\
\mathrm{Comp}_{cpMULT,A}(L, M, N) &= L \cdot N \cdot M \cdot (3\mathrm{Comp}_{EA} + 2\mathrm{Comp}_{EP}) \\
\mathrm{Comp}_{cpMULT,B}(L, M, N) &= M \cdot (L + N)\mathrm{Comp}_{Decrypt} + M \cdot L \cdot N\mathrm{Comp}_{P} + \\
&\quad L \cdot N \cdot ((M - 1)\mathrm{Comp}_{A} + \mathrm{Comp}_{Encrypt}).
\end{aligned}$$

When the previous expressions are used in this work without the parameters L, M, N it will be assumed that the product is performed between two scalars ($L = M = N = 1$).