Gram – Schmidt Orthogonalization on Encrypted Vectors

Pierluigi Failla and Mauro Barni

Abstract In this paper we address a privacy preserving version of the well known Gram – Schmidt Orthogonalization procedure. Specifically, we propose a building block for secure multiparty computation, that is able to orthogonalize a set of componentwise encrypted vectors. Our setting is the following: Bob needs to compute this orthogonalization on some vectors encrypted with the public key of Alice. Hence our intent is not to propose a stand-alone protocol to solve a specific scenario or a specific application, but rather to develop a sub-protocol to be embedded in more complex algorithms or protocols where the vectors to be orthogonalized can be the result of previous computations. We show that our protocol is secure in the *honest but curious model* and evaluate its computation complexity.

1 Introduction

The classical way to protect sensitive information from misuse is to encrypt it as soon as the information is generated and to store it in an encrypted form. However, when the information needs to be processed, it is necessary to decrypt it, hence creating a weakness in the security of the whole process. The problem with the classical approach is the assumption that the owner of the data and the party in charge of processing it trust each other: the encryption layer is used only to protect the data against third parties. In many cases, however, the owner of the information may not trust the third-parties that are asked to manipulate the sensitive informations. In this scenario the possibility of applying particular cryptographic techniques to process

Pierluigi Failla
Department of Information Engineering - University of Siena - Via Roma, 56 - 53100 Siena - Italy
e-mail: pierluigi.failla@gmail.com

Mauro Barni
Department of Information Engineering - University of Siena - Via Roma, 56 - 53100 Siena - Italy
e-mail: barni@dii.unisi.it

encrypted data has received a considerable attention in the last years. The problem of computing with encrypted data has been intensively studied in the past 30 years [21]. Following that direction, researchers developed many protocols to be applied in applications where the privacy and the security of the inputs are crucial. The proposed applications range from heuristic search in encrypted graphs [12]; ElectroCardioGram (ECG) classification [4]; data mining [1]; face recognition [11]; remote diagnosis [7].

In this paper, we consider a scenario in which two parties are interested in computing a given functionality in a privacy preserving way, but this functionality needs a sub-protocol that computes the Gram – Schmidt Orthogonalization on encrypted vectors. Our intent is to study this particular sub-protocol, giving a detailed description comprehensive of security proof and complexity evaluation.

To the best of our knowledge, this problem has never been addressed so far. Therefore this work focuses on the problem of developing a protocol that realizes the Gram – Schmidt procedure in a privacy preserving fashion. There are a lot of applications in which this kind of sub-protocol could be embedded as a basic privacy preserving primitive, including: QR decomposition [13]; linear least squares problems [6]; face recognition [24]; improving performances of neural networks [19]; wavelets computation [9]; principal component analysis [22] and image compression [18].

2 Signal Processing in the Encrypted Domain

The classical security model is targeted towards protecting the communication between two trusted parties against a third malicious party. In such cases it is sufficient to secure the transmission layer that stays on top of the processing blocks. Most of the applications today, work in this way, for instance when you log in a website, only the transmission is protected and on the other side the website is able to interpret your data in plain form while eventual thirds parties can see only an encrypted communication.

In the last years, the number of applications in which the classical model in not longer adequate has considerably increased since there are several non–trusted parties involved in the process of communication, distribution and processing data. Consider for example a remote diagnosis service (say Bob) where a non–trusted party is asked to process some medical data (owned by Alice) to provide a preliminary diagnosis. It is evident that the security of the users of such a system would be more easily granted if the server was able to carry out the task without getting any knowledge about the data provided by the users (not even the final result). Similarly the service provider may desire to keep the algorithms he is using to process the data secret, since they represent the basis for the service he is providing. Clearly the possibility of such kind of computation would be of invaluable help in situations like those described above.

Without being too specific and avoiding to discuss the details required by a precise definition, we may define *secure signal processing* or *signal processing in the encrypted domain* a collection of techniques that permit to solve the following problem: given the signals x_1 and x_2 signals (or data) belonging to Alice and Bob, compute the output of known function $f(x_1,x_2)$ without that Alice (Bob) gets any information about x_2 (x_1) in addition to that inferable from the output of the computation itself. As generalization it is possible to consider the case in which $f(.)$ is known only to a party and it has to be kept secret as well.

The number of possible applications of these techniques is virtually endless. Among the most interesting scenarios investigated so far we mention: private database access [1], in which the Alice accesses a server owned by Bob by means of an encrypted query; private data mining [17], in which two or more parties wish to extract aggregate information from a dataset formed by the union of their private data; secure processing of biometric data [8], in which biometric signals are processed in the encrypted domain to protect the privacy of the owners; watermarking of encrypted signals [16], for digital rights management within buyer-seller protocols; recommender systems [3], in which user's data is analyzed without disclosing it; privacy-preserving processing of medical data [4], in which sensitive medical data is processed by a non-trusted party, for remote medical diagnosis or any other form of home-care system whereby health conditions are monitored remotely.

3 Notation and Preliminaries

In the last few years, new techniques related to homomorphic encryption and multiparty computation showed that it is possible to perform several kinds of computations directly in the encrypted domain in an efficient way and without revealing the information hidden inside the cryptogram [2]. Following this direction, researchers developed many protocols where the protection of the inputs provided by the various parties involved in the computation is a crucial goal. The present work is part of this research streamline.

In the rest of the paper we will use the following notation:

- $\mathbb{V} = \{\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_m\}$ is a set of m vectors $\in \mathbb{R}^n$;
- with $< \cdot, \cdot >$ we indicate the inner product: $< \mathbf{a}, \mathbf{b} >= \sum_{i=1}^{n} a_i b_i$;
- with $[\![a]\!]$ we indicate the Paillier [20] encryption of a; if $\mathbf{a}$ is a vector we always indicate with $[\![\mathbf{a}]\!]$ the componentwise encryption of $\mathbf{a}$;
- s is the cryptosystem security parameter (i.e. for short term security 1024 bit) and ℓ is the bit size of a cryptogram[1], moreover $\mathbb{Z}_N$ is the ring in which the cryptosystem is defined ($s = \lceil \log_2 N \rceil$).

[1] Using the Paillier cryptosystem we have the following equality: $\ell = 2s$.

We recall that the following basic mapping[2] holds for Paillier's cryptosystem: $[\![x]\!][\![y]\!] = [\![x+y]\!]$ and $[\![x]\!]^y = [\![xy]\!]$.

Moreover we recall the Big-$\mathscr{O}$ notation [15] that measures the computational complexity in bit operations. Assuming that the biggest number involved in the computation has ℓ bits we have $add = \mathscr{O}(\ell)$ to compute addition or $mult = \mathscr{O}(\ell^2)$ to compute multiplication and finally $exp = \mathscr{O}(\ell^3)$ to compute exponentiation. In the rest of this paper we often need to compute exponentiation by -1 (or negative numbers), this operation is equivalent to compute the multiplicative inverse in the space of the ciphertexts (namely $\mathbb{Z}^*_{N^2}$), this operation can be computed by using the extended GCD and its computational complexity is equal to compute an exponentiation, so $\mathscr{O}(\ell^3)$. Furthermore we remind that for Paillier cryptosystem $enc \approx dec = \mathscr{O}(\ell^3)$.

4 Basic Building Blocks

In this section we introduce some basic building boxes that we will use to construct our protocol.

4.1 eMul

The first sub-protocol, eMul, allows to compute the product of two Paillier ciphertexts obtaining $[\![xy]\!] = \mathsf{eMul}([\![x]\!], [\![y]\!])$ and is a well-known technique. Let us recall it. Suppose that Bob owns $[\![x]\!]$ and $[\![y]\!]$ encrypted with the public key of Alice, he can obfuscate both cryptograms adding two random numbers due to homomorphic additive properties and obtain $[\![x+r_x]\!]$ and $[\![y+r_y]\!]$. Now he sends these cryptograms to Alice, she decrypts and multiplies them finding: $w = xy + xr_y + yr_x + r_x r_y$, she encrypts it and sends back to Bob that computes:

[2] To be more precise, we have that given an instance of a Paillier cryptosystem and defined as $\mathscr{E}$ and $\mathscr{D}$ the functionalities of encryption and decryption respectively, the following properties hold:

$$\mathscr{D}(\mathscr{E}(x)\mathscr{E}(y)) = \mathscr{D}(\mathscr{E}(x+y))$$

and

$$\mathscr{D}(\mathscr{E}(x)^k) = \mathscr{D}(\mathscr{E}(kx)).$$

$$\begin{aligned}
[\![w]\!][\![x]\!]^{-r_y}[\![y]\!]^{-r_x}[\![r_x r_y]\!]^{-1} = & \\
&= [\![w]\!][\![-xr_y]\!][\![-yr_x]\!][\![-r_x r_y]\!] = \\
&= [\![w - xr_y - yr_x - r_x r_y]\!] = \\
&= [\![\underbrace{xy + xr_y + yr_x + r_x r_y}_{w} - xr_y - yr_x - r_x r_y]\!] = \\
&= [\![xy]\!] \qquad (1)
\end{aligned}$$

obtaining exactly the product of the two encryptions.

Computing eMul requires 2 rounds (one from Bob to send the obfuscated ciphertexts and one from Alice to send back the result) and a bandwidth of 3ℓ (3 ciphertexts are sent) with a computational complexity equal to: 3 *exp* needed to compute $[\![x]\!]^{-r_y}$, $[\![y]\!]^{-r_x}$ and $[\![r_x r_y]\!]^{-1}$; 5 *mult* needed to obfuscate $[\![x]\!]$, $[\![y]\!]$ and to compute the additions to $[\![w]\!]$; 2 *dec* to obtain in plain $x + r_x$ and $y + r_y$ and finally 1 *enc* to encrypt the result, for a total asymptotic number of 6 *exp* operations. Later in this paper we refer to eMul using the following notation: $\mathsf{eMul}([\![x]\!], [\![y]\!]) = [\![x]\!] \bullet [\![y]\!]$.

4.2 eInv

To realize our construction we will use another building block: eInv. This subprotocol works as follow: given an encrypted value $[\![x]\!]$ we have:

$$\mathsf{eInv}([\![x]\!]) = \left[\!\!\left[\frac{1}{x}\right]\!\!\right]. \qquad (2)$$

To reach this goal we use a multiplicative blinding approach [14], in fact assuming T sufficiently bigger than x the multiplicative blinding Tx ca be assumed to be secure[3]. By this, Bob can compute $[\![Tx]\!] = [\![x]\!]^T$ by homomorphic properties and send the result to Alice that is able to decrypt obtaining Tx. Now, she computes $\frac{1}{Tx}$ encrypts it and sends back to Bob $\left[\!\!\left[\frac{1}{Tx}\right]\!\!\right]$. Bob removes the multiplicative blinding due to homomorphic properties: $\left[\!\!\left[\frac{1}{x}\right]\!\!\right] = \left[\!\!\left[\frac{1}{Tx}\right]\!\!\right]^T$ and obtain the desired result.

Computing eInv requires 2 rounds and a bandwidth of 2ℓ because only two cryptograms are sent: 1 from Bob and 1 from Alice. The computational complexity can be measured as: 1 *exp* for the multiplicative blinding, 1 *dec* for decryption, 1 *enc* for encryption and 1 *exp* to remove the blinding; for a total of 4 *exp* bit operations.

[3] Respect to the additive blinding, the multiplicative one requires a larger number of bits to achieve the same security level.

4.3 eDot

Another basic building block our protocol relies on is the the inner product between two encrytped vectors. More formally: given $[\![\mathbf{x}]\!]$ and $[\![\mathbf{y}]\!]$ encrypted with Alice's public key, the protocol $\mathsf{eDot}([\![\mathbf{x}]\!],[\![\mathbf{y}]\!])$ computes $[\![<\mathbf{x},\mathbf{y}>]\!]$. To realize this sub-protocol it is possible to use data obfuscation. Given two vectors of random values $\mathbf{r_x}$ and $\mathbf{r_y}$, generated by Bob, he is able to evaluate the obfuscation of $[\![\mathbf{x}]\!]$ and $[\![\mathbf{y}]\!]$ as the componentwise product: $[\![\mathbf{x}]\!][\![\mathbf{r_x}]\!] = [\![\mathbf{x}+\mathbf{r_x}]\!]$ and $[\![\mathbf{y}]\!][\![\mathbf{r_y}]\!] = [\![\mathbf{y}+\mathbf{r_y}]\!]$. At this point Bob can send to Alice these two vectors of obfuscated values. Alice decrypts them and computes:

$$\begin{aligned} <\mathbf{x}+\mathbf{r_x},\mathbf{y}+\mathbf{r_y}> &= \sum_{i=1}^{n}(x_i+r_{xi})(y_i+r_{y_i}) = \\ &= \sum_{i=1}^{n} x_i y_i + x_i r_{y_i} + y_i r_{xi} + r_{xi} r_{y_i} = \\ &= \sum_{i=1}^{n} x_i y_i + \sum_{i=1}^{n} x_i r_{y_i} + \sum_{i=1}^{n} y_i r_{xi} + \sum_{i=1}^{n} r_{xi} r_{y_i} \end{aligned} \tag{3}$$

encrypts the scalar product obtaining: $[\![<\mathbf{x}+\mathbf{r_x},\mathbf{y}+\mathbf{r_y}>]\!] = [\![\sum_{i=1}^{n} x_i y_i + \sum_{i=1}^{n} x_i r_{y_i} + \sum_{i=1}^{n} y_i r_{xi} + \sum_{i=1}^{n} r_{xi} r_{y_i}]\!]$. Then she sends it back to Bob. Bob has to remove the obfuscation, to do this consider that:

$$\left[\!\!\left[\sum_{i=1}^{n} x_i y_i + \sum_{i=1}^{n} x_i r_{y_i} + \sum_{i=1}^{n} y_i r_{xi} + \sum_{i=1}^{n} r_{xi} r_{y_i}\right]\!\!\right] = \left[\!\!\left[\sum_{i=1}^{n} x_i y_i\right]\!\!\right] \left[\!\!\left[\sum_{i=1}^{n} x_i r_{y_i}\right]\!\!\right] \left[\!\!\left[\sum_{i=1}^{n} y_i r_{xi}\right]\!\!\right] \left[\!\!\left[\sum_{i=1}^{n} r_{xi} r_{y_i}\right]\!\!\right] \tag{4}$$

moreover Bob can compute:

$$\left[\!\!\left[\sum_{i=1}^{n} x_i r_{y_i}\right]\!\!\right] = \prod_{i=1}^{n} [\![x_i]\!]^{r_{y_i}} \tag{5}$$

$$\left[\!\!\left[\sum_{i=1}^{n} y_i r_{xi}\right]\!\!\right] = \prod_{i=1}^{n} [\![y_i]\!]^{r_{xi}} \tag{6}$$

$$\left[\!\!\left[\sum_{i=1}^{n} r_{xi} r_{y_i}\right]\!\!\right] \tag{7}$$

by using the additive property of the cryptosystem and the fact that he knows $\mathbf{r_x}$ and $\mathbf{r_y}$ in plain. Hence Bob can compute:

$$[\![<\mathbf{x},\mathbf{y}>]\!] = [\![<\mathbf{x}+\mathbf{r_x},\mathbf{y}+\mathbf{r_y}>]\!] \left[\!\!\left[\sum_{i=1}^{n} x_i r_{y_i}\right]\!\!\right]^{-1} \left[\!\!\left[\sum_{i=1}^{n} y_i r_{xi}\right]\!\!\right]^{-1} \left[\!\!\left[\sum_{i=1}^{n} r_{xi} r_{y_i}\right]\!\!\right]^{-1}. \tag{8}$$

Computing eDot requires 2 rounds: one to send the obfuscated vectors to Alice and one to send back the result; with a bandwidth of $(2n+1)\ell$, because Bob sends 2 vectors of length n and Alice returns only 1 ciphertext. About the computational complexity we have:

$$\Big(\underbrace{2n}_{\text{compute dot product}} + \underbrace{3}_{\text{remove obfuscation}}\Big)\, mult + 2n\, dec + 1\, enc + \underbrace{2n\, exp}_{\text{obfuscate vectors}} \simeq$$
$$\simeq (4n+2)\, exp. \quad (9)$$

Table 1 shows the three complexities of the sub-protocols described so far.

Table 1 Sub-Protocols Complexities.

Sub-Protocol	Rounds	Bandwidth	# of Exponentiations
eMul	2	3ℓ	6 exp
eInv	2	2ℓ	4 exp
eDot	2	$(2n+1)\ell$	$(4n+2)$ exp

5 Gram – Schmidt Orthogonalization on Encrypted Vectors

In the following section we introduce our construction to compute Gram – Schmidt orthogonalization on encrypted vectors. First of all we give a brief description of Gram – Schmidt process in its plain version, than we examine our privacy preserving protocol paying attention to security requirements and complexities.

5.1 Gram – Schmidt Orthogonalization in the Plain Domain

Gram – Schmidt Orthogonalization is a procedure for the orthogonalization of a set of vectors in a Euclidean space [23]. Given a set $\mathbb{V}$ of m vectors, it is possible to show that Algorithm 1 replaces $\mathbb{V}$ with a set of orthogonal vectors.

Algorithm 1 Gram – Schmidt Orthogonalization

1: $s_1 = \frac{1}{<\mathbf{v}_1,\mathbf{v}_1>}$
2: **for all** $i = 2$ to m **do**
3: **for all** $j = 1$ to $i-1$ **do**
4: $\mathbf{v}_i = \mathbf{v}_i - <\mathbf{v}_i,\mathbf{v}_j> s_j \mathbf{v}_j$
5: **end for**
6: $s_i = \frac{1}{<\mathbf{v}_i,\mathbf{v}_i>}$
7: **end for**

Note that the computational complexity is asymptotically equal to $\mathcal{O}(nm^2)$ multiplications [13]. Adding a normalization step at the end of Algorithm 1 it is possible to obtain an orthonormalized set. However techniques for the normalization of encrypted vectors are out the scope of this paper.

5.2 Privacy Preserving Gram – Schmidt Protocol

We consider the case in which Bob owns a set of componentwise encrypted vectors with the public key of Alice and he needs to extract an orthogonalized version of them. For sake of simplicity we assume that $\mathbb{V}$ is a set of linearly independent vectors; this choice avoids the necessity of catching a *division by zero*. Otherwise it is possible considering a variant where Bob asks to Alice to check if $< \mathbf{v}_i, \mathbf{v}_i >$ is equal to zero, if is this the case Alice just sends back the encryption of zero[4]. We already introduced all the basic blocks we will use (See Section 4), so we can translate Algorithm 1 in the following Protocol 2.

Protocol 2 Privacy Preserving Gram – Schmidt Orthogonalization

1: $[\![s_1]\!] = \mathsf{eInv}(\mathsf{eDot}([\![\mathbf{v}_1]\!], [\![\mathbf{v}_1]\!]))$
2: **for all** $i = 2$ to m **do**
3: **for all** $j = 1$ to $i-1$ **do**
4: $[\![\mathbf{v}_i]\!] = [\![\mathbf{v}_i]\!]\mathsf{eDot}([\![\mathbf{v}_i]\!], [\![\mathbf{v}_j]\!]) \bullet [\![s_j]\!]^{-1} \bullet [\![\mathbf{v}_j]\!]$
5: **end for**
6: $[\![s_i]\!] = \mathsf{eInv}(\mathsf{eDot}([\![\mathbf{v}_i]\!], [\![\mathbf{v}_i]\!]))$
7: **end for**

During Step 1 we use sub-protocols eInv and eDot to compute $s_1 = \frac{1}{<\mathbf{v}_1, \mathbf{v}_1>}$ used later in Step 4 to scale the projection. The main part is Step 4 where using sub-protocols eDot and eMul it is possible calculate $\mathbf{v}_i = \mathbf{v}_i - < \mathbf{v}_i, \mathbf{v}_j > s_j \mathbf{v}_j$.

5.2.1 Security Discussion

To discuss the security of our construction we simply recall that in each step the data are encrypted when used by Bob or obfuscated, when used by Alice. Thus the privacy is achieved in the *honest but curious* model [10] due to IND-CPA property of Paillier cryptosystem and the security of obfuscation.

[4] It is simple to note that this reveals to Alice the rank of the set $\mathbb{V}$.

5.2.2 Complexities

We now briefly discuss the complexity of the proposed protocol. We assume that Bob already owns the vectors so in our complexity evaluation we do not consider: the bandwidth, the rounds and the operations needed to manipulate the vectors until the beginning of this protocol. We will examine the two principal steps in the protocol: Step 4 and Step 6 (this is equal to Step 1). During **Step 4** there are $2n+1$ calls to eMul and 1 to eDot, so $4n+3$ rounds are needed with a bandwidth of $(2n+7)\ell$. Finally resulting in a computational complexity of $(16n+8)$ exp. Now, consider that it is necessary to execute Step 4 $\frac{m(m+1)}{2}$ times. Summarizing Step 4 requires: $\frac{m(m+1)(4n+3)}{2}$ rounds; a bandwidth of $\frac{m(m+1)(2n+7)}{2}\ell$ and a computational complexity of: $\frac{m(m+1)(16n+8)}{2}$ exp. Finally **Step 6** requires m executions and for each of them is just 1 call to eInv and 1 to eDot, so we can affirm that: $2m+2m=4m$ rounds are needed with a bandwidth[5] $(2+n+1)m\ell=(3+n)m\ell$ and a computational complexity $(4+4n+2)$ $exp=(6+4n)m$ exp. Summarizing we have:

$$\underbrace{\frac{m(m+1)(4n+3)}{2}}_{\text{Step 4}}+\underbrace{4m}_{\text{Step 6}}\simeq\mathscr{O}(nm^2) \tag{10}$$

rounds, for a bandwidth of:

$$\left(\underbrace{\frac{m(m+1)(2n+7)}{2}}_{\text{Step 4}}+\underbrace{(3+n)m}_{\text{Step 6}}\right)\ell\simeq\mathscr{O}(nm^2\ell) \tag{11}$$

bits, and eventually:

$$\left(\underbrace{\frac{m(m+1)(16n+8)}{2}}_{\text{Step 4}}+\underbrace{(6+4n)m}_{\text{Step 6}}\right)exp=\mathscr{O}(nm^2\ell^3) \tag{12}$$

bit operations.

6 Conclusion

In this paper, we have proposed a secure protocol to compute the Gram – Schmidt Orthogonalization on vectors encrypted with an additive homomorphic cryptosystem like Paillier's. We proved that our construction is secure because nothing is

[5] Consider that eDot is computed on the same vector $\mathbf{v}_i$, so just n encryptions are sent by Bob instead than $2n$.

revealed to the parties. The idea was to propose a building block that could be used *off the shelf* in more complex privacy-preserving systems. To achieve our goal, we proved the protocol to be secure in the *honest but curious* model. Moreover we show all the complexities involved: bandwidth, rounds and bit operations.

In the future, various improvements can be investigated, for instance, it is clear that for a real use of this protocol it is necessary to quantize the vectors because cryptosystems work on integer number representation, so a study of the error introduced by quantization will be really needed for practical implementations. Furthermore, techniques to normalize the encrypted vectors will be useful to generate an orthonormal version of the basis. Finally, it is possible to use techniques like those proposed in [5] to study a packetized version of our construction that could be much more efficient.

Acknowledgements This work is partially sponsored by MIUR under project Priv-Ware (contract n. 2007JXH7ET).

References

1. R. Agrawal and R. Srikant. Privacy-preserving data mining. *ACM Sigmod Record*, 29(2):439–450, 2000.
2. N. Ahituv, Y. Lapid, and S. Neumann. Processing encrypted data. *Communications of the ACM*, 30(9):780, 1987.
3. E. Aimeur, G. Brassard, J.M. Fernandez, F.S.M. Onana, and Z. Rakowski. Experimental Demonstration of a Hybrid Privacy-Preserving Recommender System. In *The Third International Conference on Availability, Reliability and Security*, pages 161–170. IEEE, 2008.
4. M. Barni, P Failla, V. Kolensikov, R. Lazzeretti, A. Paus, A. Sadeghi, and T. Schneider. Efficient Privacy-Preserving Classification of ECG Signals. In *Workshop on Information Forensics and Security, WIFS 2009*, 2009.
5. T. Bianchi, A. Piva, and M. Barni. Efficient pointwise and blockwise encrypted operations. In *Proceedings of the 10th ACM workshop on Multimedia and security*, pages 85–90. ACM, 2008.
6. A. Bjorck. Solving linear least squares problems by Gram-Schmidt orthogonalization. *BIT Numerical Mathematics*, 7(1):1–21, 1967.
7. J. Brickell, D.E. Porter, V. Shmatikov, and E. Witchel. Privacy-preserving remote diagnostics. In *Proceedings of the 14th ACM conference on Computer and communications security*, page 507. ACM, 2007.
8. J. Bringer and H. Chabanne. An authentication protocol with encrypted biometric data. In *Proceedings of the Cryptology in Africa 1st international conference on Progress in cryptology*, pages 109–124. Springer-Verlag, 2008.
9. C.K. Chui and E. Quak. Wavelets on a bounded interval. *Numerical methods of approximation theory*, 9(1):53–57, 1992.
10. R. Cramer. Introduction to secure computation. *Lectures on Data Security*, pages 16–62.
11. Z. Erkin, M. Franz, J. Guajardo, S. Katzenbeisser, I. Lagendijk, and T. Toft. Privacy-preserving face recognition. In *Privacy Enhancing Technologies*, pages 235–253. Springer.
12. P. Failla. Heuristic Search in Encrypted Graphs. *Accepted at IARIA International Conference on Emerging Security Information, Systems and Technologies, SECURWARE 2010*, 2010.
13. G.H. Golub and C.F. Van Loan. *Matrix computations*. Johns Hopkins Univ Pr, 1996.
14. F. Kerschbaum. Practical privacy-preserving benchmarking. In *Proceedings of the 23rd IFIP International Information Security Conference*, pages 17–31. Springer, 2008.

15. N. Koblitz. *A course in number theory and cryptography*. Springer, 1994.
16. A. Lemma, M. Van Der Veen, P. Tulys, and A. Kalker. Homomorphic Encryption for Secure Watermarking, 2006. WO Patent WO/2006/129,293.
17. Y. Lindell and B. Pinkas. Privacy preserving data mining. *Journal of cryptology*, 15(3):177–206, 2008.
18. Y.D. Ma, C.L. Qi, Z.B. Qian, F. Shi, and Z.F. Zhang. A novel image compression coding algorithm based on pulse-coupled neural network and Gram-Schmidt orthogonal base. *Dianzi Xuebao(Acta Electronica Sinica)*, 34(7):1255–1259, 2006.
19. S.J. Orfanidis. Gram-Schmidt neural nets. *Neural Computation*, 2(1):116–126, 1990.
20. P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. *Advances in Cryptology EUROCRYPT 1999*, pages 223 – 238, 1999.
21. R.L. Rivest, L. Adleman, and M.L. Dertouzos. On data banks and privacy homomorphisms. *Foundations of Secure Computation*, pages 169–178, 1978.
22. A. Sharma and K.K. Paliwal. Fast principal component analysis using fixed-point algorithm. *Pattern Recognition Letters*, 28(10):1151–1155, 2007.
23. L.N. Trefethen and D. Bau. *Numerical linear algebra*. Society for Industrial Mathematics, 1997.
24. W. Zheng, C. Zou, and L. Zhao. Real-time face recognition using Gram-Schmidt orthogonalization for LDA. *Pattern Recognition*, 2:403–406, 2004.