Privacy Preserving Processing of Biomedical Signals with Application to Remote Healthcare Systems

Riccardo Lazzeretti

Ph.D Thesis in Information Engineering

University of Siena

UNIVERSITÀ DEGLI STUDI DI SIENA

Facoltà di Ingegneria

Dipartimento di Ingegneria dell'Informazione

Privacy Preserving Processing of Biomedical Signals with Application to Remote Healthcare Systems

Riccardo Lazzeretti

Ph.D Thesis in Information Engineering

Supervisor

Prof. Mauro Barni

Examination Commitee

Prof. Dario Catalano

Prof. Fernando Pérez-González

Dr. Sandro Bartolini

Thesis reviewers

Prof. Dario Catalano

Prof. Fernando Pérez-González

Prof. Dr. Reginald Lagendijk

Siena, November 14, 2012

Contents

Acknowledgements **xv**

1 Introduction **1**
- 1.1 Motivations 1
- 1.2 Contributions 6
- 1.3 Publication List 9
- 1.4 Outline 11

I Cryptographic tools **13**

2 Preliminary Notions **17**
- 2.1 Adversaries 17
- 2.2 Security services 19
  - 2.2.1 Authenticity 19
  - 2.2.2 Access Control 20
  - 2.2.3 Data Confidentiality 20
  - 2.2.4 Data Integrity 20
  - 2.2.5 Non-repudiation 21
- 2.3 Security Models 21
  - 2.3.1 Standard Model 21
  - 2.3.2 Random-Oracle Model 21
- 2.4 Cryptographic Primitives 22

2.4.1 Symmetric Key Cryptography . . . 22
2.4.2 Public Key Encryption . . . 23
2.4.3 Cryptographic Hashes . . . 27
2.4.4 Secure Multi-Party Computation . . . 28
2.5 Additive and multiplicative blinding . . . 29
2.6 Signal representation . . . 32
2.6.1 Approximating real numbers on finite rings . . . 32

3 Homomorphic Encryption 39
3.1 Basis of homomorphic encryption . . . 39
3.2 El-Gamal cryptosystem . . . 41
3.3 Paillier cryptosystem . . . 43
3.4 Basic homomorphic protocols for STPC . . . 44
3.4.1 Product with a plain value . . . 44
3.4.2 Scalar and Matrix product . . . 45
3.4.3 Linear Filtering . . . 46
3.4.4 Packing . . . 47
3.4.5 Non linear function evaluation . . . 49
3.4.6 Product between two encrypted numbers . . . 50
3.4.7 Multiplexer . . . 51
3.4.8 Scalar and Matrix products between encrypted matrices 52
3.4.9 Square of an encrypted number . . . 55
3.4.10 Energy of an encrypted signal . . . 56
3.4.11 Binary representation of an encrypted value . . . 57
3.4.12 Comparison of encrypted numbers . . . 59
3.4.13 Minimum selection . . . 65
3.4.14 Distances . . . 66
3.5 Applications of homomorphic-based STPC . . . 68
3.5.1 Discret Fourier Transform . . . 69
3.5.2 Composite signal representation . . . 71
3.6 Fully homomorphic encryption . . . 74

4 Oblivious Transfer 77
4.1 1-out-of-2 Oblivious Transfer . . . 77
4.1.1 Protocol relying on Random Oracles . . . 77

4.1.2 Protocol not relying on Random Oracles 81
4.2 1-out-of-n Oblivious Transfer 83
4.2.1 Protocol relying on Random Oracles 83
4.2.2 Protocol not relying on Random Oracles 84
4.3 Bandwidth/Computation tradeoff for parallel 1-out-of-2 Oblivious Transfer 86
4.4 Extending *OT* efficiently 88
4.5 Precomputing OT 91

5 Garbled Circuits **93**
5.1 Garbled Circuits description 93
5.1.1 Contructor 94
5.1.2 Data exchange 96
5.1.3 Evaluator 96
5.2 Implementation details 96
5.3 Building blocks for GC: general overview 98
5.3.1 Addition (ADD) 99
5.3.2 Sum of k values 99
5.3.3 Subtraction (SUB) 101
5.3.4 Controlled Addition or Subtraction(ADDSUB) 102
5.3.5 Changing representation 102
5.3.6 Counter (COUNT) 103
5.3.7 Product (MUL) 105
5.3.8 Scalar Product and Matrix Product 105
5.3.9 Linear Filtering 106
5.3.10 Multiplexer (MUX) 107
5.3.11 Square value (SQR) 107
5.3.12 Energy computation 109
5.3.13 Comparison 110
5.3.14 Minimum Value 111
5.3.15 Distances 114
5.3.16 Logarithm (LOG) 115

6 Hybrid Protocols 117
6.1 Comparison between HE and GC solutions 117
6.1.1 Addition 118
6.1.2 Product 118
6.1.3 Multiplexer 120
6.1.4 Comparison and minimum value 121
6.1.5 Distances 122
6.2 Hybrid HE and GC protocols 124
6.2.1 Interface from HE to GC subprotocols 124
6.2.2 Interface from GC to HE subprotocols 126

II Electrocardiogram Classification 129

7 Plain Protocol 133
7.1 Introduction to Electrocardiogram and heart diseases 133
7.2 ECG Classification 138
7.2.1 Overall architecture of the classifier 139
7.2.2 Classification results 145

8 Privacy preserving Classification by using LBP 147
8.1 Linear Branching Programs (LBP) 147
8.1.1 Full-GC implementation 149
8.1.2 Hybrid implementation 151
8.2 Application of LBP to ECG classification 154
8.2.1 Quantization error analysis 156
8.2.2 Analysis 162

9 Privacy preserving Classification by using Neural Network 167
9.1 Neural Network design 167
9.2 Quantized Neural Network classifier 174
9.2.1 Representation vs. Classification Accuracy 177
9.3 Privacy-Preserving NN classifier 179
9.3.1 Comparison with the LBP solution 182

III Quality Evaluation **185**

10 SNR computation **189**
10.1 Noise affecting ECG recording 191
10.2 SNR Evaluation in the Encrypted Domain 193
10.3 Protocol Analysis 199
  10.3.1 Bitsize analysis 199
  10.3.2 Communication complexity 201
10.4 Error Analysis 203

11 Privacy Preserving Quality Evaluation **207**
11.1 Protocol description 209
11.2 Analysis 215
  11.2.1 Data Dimension 215
  11.2.2 Communication Complexity 217
  11.2.3 Classification Performance 219

12 Conclusions **221**
12.1 Tracks for future works 223
12.2 Final considerations 225

Bibliography **227**

List of Figures

2.1 Entities involved in a simple communication. 18
2.2 Typical scenario for symmetric key cryptography. 23

3.1 Packing more values. 48
3.2 eMin protocol extended to more values. 66
3.3 Graphical representation of M-PCR. 72

5.1 General scheme for Garbled Circuits. 94
5.2 Addition Circuit 100
5.3 Reverse tree structure for the computation of n additions. . . . 100
5.4 Subtraction Circuit 102
5.5 Controlled addition or subtraction circuit 103
5.6 Counter Circuit 104
5.7 Product circuit. 105
5.8 Multiplexer circuit 108
5.9 Square circuit. 109
5.10 Equality test circuit ($\overset{?}{=}$). 110
5.11 Comparison Circuit 111
5.12 Minimum Circuit 112
5.13 Euclidean Distance circuit. 114
5.14 Hamming Distance circuit. 115
5.15 Logarithm circuit. 116

6.1 Communication Complexity of the sum. 119
6.2 Communication complexity of product between input operands. 119
6.3 Communication complexity of product between secrets. 120
6.4 Communication Complexity of the multiplexer. 121
6.5 Communication Complexity of the comparison. 122
6.6 Communication Complexity of the Euclidean distance. 123
6.7 Communication Complexity of the Hamming distance. 124

7.1 Positions where to place electrodes. 135
7.2 Waves on ECG trace. 136
7.3 Examples of Arrhythmias 137
7.4 Block Diagram. 139
7.5 Distorted frequency response as derived from a truncated impulse response. 140
7.6 Pan - Tompkins Algorithm. 141
7.7 Decision graph leading to ECG segment classification 145

8.1 Full-GC solution for LBP. 150
8.2 Privacy-preserving ECG diagnosis. 155
8.3 Ring bit length. 160
8.4 Classification accuracy of dataset using 21 and 15 features . . . 161

9.1 A perceptron. 168
9.2 Transfer functions 171
9.3 Classification accuracy with the tansig or hardlims transfer functions 172
9.4 Classification accuracy with the tansig or satlins transfer functions 173
9.5 Classification accuracy with only SATLIN activation function . 173
9.6 NN topology. 174
9.7 Chain blocks to compute the output of a NN. 174
9.8 Quantized NN. In our case we have $n = 4, n_h = 6$ and $n_o = 6$. . 177
9.9 Classification accuracy as a function of ℓ^i, ℓ^h, ℓ^o. 178
9.10 Classification accuracy in function of ℓ^o, with $\ell^i = \ell^h = 13$. . . 178

10.1 Scheme to compute the SNR. 191

10.2 Filter plots 195
10.3 Error incurred by using the integer logarithm 205

11.1 Sequence of steps performed to evaluate the quality of an ECG signal. 209

List of Tables

3.1 Some homomorphic encryption systems. 42
3.2 Product by plain value: complexities 45
3.3 Scalar product with plain vector: complexities 45
3.4 FIR filtering: complexities 47
3.5 Packing: complexities 49
3.6 eMul: complexities 51
3.7 Multiplexer: average complexities 52
3.8 eScalarProd: Complexities 54
3.9 Matrix product between encrypted matrixes: complexities . . . 55
3.10 eSquare: complexities 55
3.11 eEnergy: complexities 57
3.12 toEncBit: average complexities 59
3.13 Equality comparison: average complexities 62
3.14 Minor comparison: average complexities 65
3.15 eMin: average complexities 65
3.16 eEuclideanDistance - complexities 67
3.17 eEuclideanDistance with $\mathbf{x}$ provided by $\mathcal{C}$ - complexities 67
3.18 Hamming Distance - average complexities 68

4.1 1-out-2 *OT* relying on Random Oracle - Complexities 79
4.2 1-out-2 *OT* not relying on Random Oracle - Complexities . . . 83
4.3 1-out-n *OT* relying on Random Oracle - Complexities 84
4.4 1-out-n *OT* not relying on Random Oracle: complexities 86

4.5 Complexity of a single 1-out-2 OT performed in a parallel OT. 88
4.6 Communication complexity of OT_t^n, implemented by an OT_k^k . 91
4.7 Communication complexity of a precomputed OT_t^n. 92

5.1 General average GC complexities 99
5.2 Number of non-XOR gates for COUNT$^\ell$. 104

6.1 Complexity of the HE to GC interface. 126
6.2 Complexity of the HE to GC interface. 128

7.1 Classification pattern 144
7.2 QDF-based classification results 146
7.3 Our QDF-based classification results 146

8.1 Protocols for Secure Evaluation of Private BP/LBP 154
8.2 Numerical results 160
8.3 Estimated Communication Complexity 163
8.4 Performance of protocols for secure ECG classification 163
8.5 Communication complexity of LBP-based ECG classification protocol. 166

9.1 NN Parameters. 179
9.2 Complexity of NN-based ECG classification protocol. 181

10.1 Number of bits necessary to represent the values obtained by a worst case analysis. 201
10.2 Communication complexity of SNR protocol. 203

11.1 Maximum value and number of bits necessary for the magnitude representation of the variables involved in the computation by worst case analysis. Another bit is needed for the sign, except for energies and SNR variance. 218
11.2 Online and offline bandwidth (bits) required by the protocol . . 219
11.3 Performance of the protocol using the linear classifier or a single feature. 220

List of Protocols

1 Function $\phi(x)$ evaluation by using obfuscation with homomorphic encryption scheme. 31
2 Interactive protocol that computes the product between two encrypted values. 50
3 Interactive protocol that select a value depending on a boolean. . . . 52
4 Interactive protocol that computes the scalar product between two encrypted arrays. 53
5 Interactive protocol that computes the square of an encrypted value. 56
6 Interactive protocol that computes the energy of a signal. 57
7 Interactive protocol that returns the encrypted bynary representation of a value. 58
8 Interactive protocol that evaluates if two encrypted values are equal. 60
9 Interactive protocol that evaluates $[x \overset{?}{<} y]$. 63
10 Interactive DGK protocol. 64
11 Interactive protocol that computes the minimum among two values. 65
12 Interactive protocol that computes the euclidean distance between two vectors. 67
13 The Oblivious Transfer protocol using a Random Oracle proposed by Bellare and Micali 78
14 The Oblivious Transfer protocol based on Elliptic curves 80
15 The 1-out-2 Oblivious Transfer protocol not relying on a Random Oracle 82
16 1-out-of-n Oblivious Transfers with the same g^r 84

17 The 1-out-n Oblivious Transfer protocol not relying on a Random Oracle . . . 85
18 Parallel 1-out-2 OT Protocol . . . 87
19 Protocol that reduces OT^n_t to OT^n_k . . . 89
20 Protocol that reduces OT^n_ℓ to OT^k_n (Given the matrix T, we here indicate the j-th column by t^j and the i-th row by t_i, the same for matrix Q). . . . 90
21 Precomputed OT protocol with setup phase . . . 92
22 Interactive interface that returns to $\mathcal{C}$ the secrets relative to x, given the cyphertext $[\![x]\!]$ available to $\mathcal{S}$. . . . 125
23 Interactive interface that returns the cyphertext $[\![x]\!]$ to $\mathcal{S}$, given the secrets $\tilde{w}^{x_{\ell-1}}_{\ell-1}, \ldots, \tilde{w}^{x_0}_0$ relative to x available to $\mathcal{C}$. . . . 127
24 Hybrid LBP protocol. For simplicity we assume that all the y_i values can be packed in a single cyphertext. . . . 152
25 Hybrid SNR computation protocol (packing and bitlengths are omitted for simplicity). . . . 198
26 Hybrid protocol for ECG quality evaluation (packing and bitlengths are omitted for simplicity). . . . 214

Acknowledgements

I'd like to express my gratitude to all those who made it possible to complete this thesis. In particular I am deeply indebted to my supervisor **Prof. Mauro Barni** whose help, stimulating suggestions and encouragement helped me in all the time of research for and writing of this thesis. I am also grateful to Senior Scientist **Jorge Guajardo** who invited me to spend six month in Philips (Eindhoven) and for all the help and friendship he offered me.

I am grateful to thesis reviewers **Prof. Dario Catalano**, **Prof. Fernando Pérez-González** and **Prof. Dr. Reginald Lagendijk** for their suggestions and enthusiastic comments, and to the members of my thesis committee, **Prof. Dario Catalano**, **Prof. Fernando Pérez-González** and **Dr. Sandro Bartolini**, for accepting to be part of the committee.

Special thanks are due to the best research mate I could have, **Pierluigi Failla**, that shared with me a lot of beautiful experiences while we were starting to move our first steps in this interesting topic and that has a so big heart that he doesn't hate me, even if I lost his thesis corrections. Another special thanks is for **Tommaso Pignata**, that is actually working with me.

It was a pleasure to cooperate with **Thomas Schneider**, **Zekeriya Erkin**, **Tommaso Bianchi** and all the other people involved in the SPEED and Priv-WARE projects.

I am also indebted with all my current and past colleagues of the VIPP group (**Andrea**, **Angela**, **Benedetta**, **Giacomo**, **Giulia**, **Guido**, **Luca**, **Marco**, **Pedro**, **Siméon**), **Fabio** and all the other people of the LTT lab for

the funny days I spent with them.

I can't forget in my acknowledgments all the wonderful house-mates (**Tania**, **Jasmin**, **Lilian**, **Pasquale**, **Vincent**, **Felipe**, **Florian** and **Artur**), office-mates (**Arezou** and **Louiza**) and friends (**Eirini**, **Chiara**, **Houria**, **Helen**, **Daphne**, **Maryam**, **Eva**, **Stefano**, **Rocco**, **Diego**, **Davide**, etc.) that made my stay in Eindhoven unforgettable.

I would also acknowledge my best friend **Gabriele** and all the other friends I have in Siena and in Italy.

My special appreciations are for my family: my father **Fosco**, my mother **Rosanna**, my sister **Laura** and my nephew **Daniele**, that always supported me. A big huge to **Fuffola** that stayed next to me for great part of my life, and all the other pets I had.

My final acknowledgment is for my beloved wife **Silvia**, that is actually working on the best gift ever.

Siena
November 14, 2012

Chapter 1

Introduction

1.1 Motivations

Recent technological developments promise to make available in the near future a large variety of new applications ranging from multimedia content production and distribution, to advanced healthcare systems for continuous health monitoring. These advances are raising several important issues concerning the security of the contents themselves, including intellectual property rights management, authenticity, privacy, and conditional access to digital data, that risk slowing the diffusion of new services. It is evident that the development of tools allowing secure manipulation of signals is a pressing need.

Common available technological solutions for “secure manipulation of signals” do not appear to be a valid solution to the above problems, since they simply apply some cryptographic primitives in order to build a secure layer on top of the signal processing modules, to protect them from leakage of critical information. These solutions typically assume that the involved parties or devices trust each other, and thus the cryptographic layer is used only to protect the data against third parties that are not authorized to access the data or for authentication purposes. Sometime, this is not the case, since the owner of the data may not trust the processing devices, or those actors that are in charge of data manipulation.

In the last years an increasing attention has been given to the development of tools for processing encrypted signals [EPK+07], where two or more non-trusted parties wish to collectively process one or more signals to reach a common goal. In the simplest case, the above scenario consists of only two parties. One party, hereafter referred to as the Client ($\mathcal{C}$) owns a signal that has to be processed in some way by the other party, hereafter referred to as the Server ($\mathcal{S}$). Since $\mathcal{C}$ and $\mathcal{S}$ do not trust each other, $\mathcal{S}$ is required to process the signal owned by $\mathcal{C}$ without obtaining any information about it, not even the result of the processing. At the same time, $\mathcal{S}$ wants to protect the information it uses to process the signal provided by $\mathcal{C}$. While the above may seem a formidable, if not impossible, task, a bunch of cryptographic primitives exist, that coupled with a suitable design of the underlying signal processing algorithms, allow to process signals that have been *secured* in some way, e.g. (but not only) by encrypting them. In the recent scientific literature such techniques are usually referred to as s.p.e.d. (standing for Signal Processing in the Encrypted Domain), or SSP (for Secure Signal Processing) techniques.

In the last thirty years[1] the cryptographic community has worked hard to build a set of tools that allow to compute with encrypted data. Available solutions include the use of *Homomorphic Encryption*, whereby some algebraic operations are mapped into simple operations to be applied in the encrypted domain, and *Secure Multi Party Computation - SMPC* where two or more non-trusted parties engage in an interactive protocol to carry out a computation without revealing their own inputs. The special case where only two parties are involved, such as a Client and a Server, is of particular interest and is usually referred to as Secure Two Party Computation (STPC).

Though the possibility of processing encrypted data has been advanced more than thirty years ago, processing encrypted signals poses some new problems due to the peculiarities of this kind of data with respect to the type of data commonly encountered in the cryptographic literature, e.g. alphanumeric strings or bit sequences. The most straightforward difference is that signals are often represented by means of real numbers (and processed by means of floating point arithmetic), while all the available cryptosystems work on integer rings. Moreover the representation of signals in the encrypted domain causes a huge increase in the data size. Despite the above difficulties, some recent studies have shown that the application of non-trivial processing tools to encrypted signals is practically feasible.

The number of possible applications of s.p.e.d. techniques is virtually endless. This thesis focuses on biomedical signal analysis [BPSW07, BFL$^+$11, LGB12]. Another important application is biometric matching [BC08], re-

---

[1] The first mention is in [RAD78b] 1978 by Rivest et al.

cently applied to face recognition [EFG+09, SSW09], fingerprint recognition [BBC+10b, BBC+10a] and iris matching [LSP+12]. Many other application fields can benefit from privacy preserving techniques. In [KLC+08] an efficient buyer-seller protocol embedding an encrypted watermark in a content is proposed, protecting the watermark secrets from the buyer and preventing false infringement accusations by the seller. In [Fai10], a novel technique has been proposed to compute the well-known A^* algorithm[2], on the encrypted weights of a graph. Other applications include data mining on private databases [AS00, LP08]; recommender systems [ABF+08, EBVL11, EVL11, EVTL12], encrypted strings comparison by using Levenshtein distance [RS10], etc.

The use of s.p.e.d. for the processing of medical signals is surely one of the most promising applications among those listed above. As a matter of fact, healthcare industry is rapidly moving toward technologies offering personalized online self-service, medical error reduction, customer data collection and more. Such technologies have the potentiality of revolutionizing the way medical data is stored, processed, delivered and made available in an ubiquitous and seamless way to millions of users throughout the world. In this framework, respecting the privacy of customers is a central problem, since privacy concerns may impede, or at least slow down, the diffusion of new e-health services. This is the case, for example, of on-line repositories of medical data (including signals) managed by a third party [McB08, Bla08]. Would anybody be willing to store his/her medical data in such repositories if his/her privacy rights are not adequately protected? This is the main reason behind the low success of Google Health and the decision to stop the service [GB11, Com11].

In this thesis, we propose some protocols that demonstrate the feasibility of biomedical signal analysis, when non-trusted parties are involved in the computation. Even if the protocols are still far from the efficiency one would need in practical implementation, they are a good starting point for privacy preserving healthcare applications, demonstrating that the privacy of the parties involved can be preserved by using protocols with reasonable communication and computation complexity. The protocols are based on homomorphic encryption and, most of all, on garbled circuits, the latter con-

---

[2] A^* is a *best first* graph search algorithm that uses an heuristic function helping to choose the best candidates during the traversing of common graphs.

stituting a novelty in s.p.e.d. field, wherein the use of GC theory has often been overlooked.

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

In particular, we considered the privacy-preserving classification of ElectroCardioGram (ECG) signals. Classification of ECG signals has long been studied by the signal processing community and several accurate and efficient algorithms exist for this purpose [ASSK07]. Classification and diagnostic programs are very useful tools for automatic data analysis with respect to specific properties. They are deployed for various applications, from spam filters [DCDZ05], remote software fault diagnostics [HRD+07] to medical diagnostic expert systems [RGI05]. However their efficient implementation in a s.p.e.d. framework is not an easy task. Given an ECG signal, we have developed a system that classifies ECG portions corresponding to single heart beats into six possible classes: five possible diseases and one healthy state. To do so, $\mathcal{C}$ encrypts some features extracted from the ECG and starts a classification protocol, interacting with $\mathcal{S}$, who inputs the classification parameters. On one side $\mathcal{C}$ does not want to reveal the ECG features since they are sensitive information that must be kept secret, while on the other side, $\mathcal{S}$ does not want to reveal the classification parameters.

To provide a correct remote service, it is important to guarantee that the recorded signals comply with certain quality measures. Namely, when data remotely measured by patients is used by tele-health services, healthcare providers must trust the information provided by patients. In fact, as a result of the widespread use of remote e-health systems a multitude of vital body signals can be recorded at a remote location (e.g., at home) and transmitted to a (remote) service provider for processing and assessment. The fact that measurements are going to be performed by consumers raises the need for privacy preserving solutions that allow to assess the *quality* of the recorded signals, thus ensuring that the signals are good enough to guarantee correct decisions and handling. In particular, in order to make sound medical decisions based on remotely measured biomedical signals, it has to be ensured that a measurement contains as little noise as possible, the amount of noise being an indication of the quality of the signal. This is very important because if this is not guaranteed there can be critical healthcare decisions based on wrong (or poor quality) data, which in turn, can lead to wrong decisions or treatments. Even if quality evaluation can be embedded in the recording device available to the user, a privacy preserving quality evaluation protocol can be useful in many cases. For example when the service provider is interested in changing the protocol parameters without upgrading the software (or hardware) of the devices. Otherwise if the service provider and the company producing the device are two distinct elements and the former is interested to protect his property also by the company. Alternatively it can be useful when the signal comes from previous elaboration and is available only in its encrypted form. Also device calibration can be considered a major issue, but if performed in presence of trusted stuff, as we here suppose, we can relax the privacy and security constrains related to.

We focus on the problem of measuring the quality of an Electrocardiogram (ECG) signal in the encrypted domain. The server $\mathcal{S}$ computes the quality of an ECG signal having only access to its encrypted version and allowing interaction. At the same time, the client $\mathcal{C}$ is not allowed to know the intermediate results of the computation and the parameters of the algorithms used by $\mathcal{S}$, so that the server is able to protect the details of the quality evaluation algorithm since this may be a proprietary algorithm. In order to propose a simple protocol that can be easily implemented with MPC techniques, we focus on a scheme where the quality of a signal is given by a classifier that, subdivided the signal in segments, uses the Signal to Noise Ratio (SNR). Being a reference clean signal not available, SNR is evaluated between the original segments and a filtered version. It is important to note that measuring signal quality is a difficult problem that has captured the attention of researchers in many fields, as diverse as audio, image, video and medical signal analysis, and our protocol can be adapted to be also applied to these signals.

1.2 Contributions

The main contributions of this thesis are new more efficient modular protocols for the secure evaluation of a class of diagnostics/classification problems. In fact, despite the emerging success of multi-party computation tools before our research, efficients tools were not yet proposed. This thesis demonstrates that s.p.e.d. applications are feasible, also in a promising, but ticklish scenario such as the biomedical one.

The activity has been mainly developed during the European SPEED project[3], where the group of the university of Siena, lead by Prof. Mauro Barni with the participation of Pierluigi Failla and me, acted as coordinator. The goal of SPEED was to foster the advancement of the marriage between Signal Processing and Cryptographic techniques, through the initiation and development of a totally new and unexplored interdisciplinary framework and technologies for signal processing in the encrypted domain. SPEED research activity was carried out both at a theoretical and a practical level, the theoretical part being dedicated to the development of a general framework investigating the fundamental limits and trade-offs of s.p.e.d., and the practical part being devoted to the development of some of the basic s.p.e.d. building blocks and to their application in some selected scenarios: secure biomedical signal analysis, secure biometric matching and secure watermarking. Our research in secure biomedical signal processing continued after the end of the project, also during the period that I spent at Philips Lab, Eindhoven, Netherland (other member of the SPEED project) as an intern, under the guide of the Senior Scientist Jorge Guajardo.

Many new building blocks have been introduced in this thesis. First of all our applications rely on innovative hybrid protocols, i.e. in the composition of many s.p.e.d. subprotocols based on different tools, proposed by Thomas Schneider, Ahmad-Reza Sadeghi (of the Ruhr-Universität of Bochum (RUB) group, one of the partners in the SPEED project) and Vladimir Kolesnikov

---

[3] `www.speedproject.eu`

in [KSS09a] (later extended in [KSS10]) and applied for the first time in our paper [BFK+09a]. It combines the advantages offered by Homomorphic Encryption (HE) [Pai99] and Garbled Circuits (GC) [Yao86]. We also introduced the general concept of Linear Branching Program [BFK+09a], a natural generalization of binary classification trees. LBPs represent an efficient system for privacy-preserving classification of signals. Moreover we proposed an efficient implementation of Neural Networks by using GC in [BFL+11]. This efficient implementation is based on the use of the satlin activation function, instead of the classic tansig activation function. Another important building block is the proposal of a GC protocol that evaluates the integer logarithm in [BGL10], whose main element is an optimized counter circuit. This solution was developed during the period that I spent in Philips. In the same paper the logarithm protocol has been used in a hybrid protocol to compute the SNR of a signal.

The above building blocks has been used to develop the biomedical s.p.e.d. protocols we propose in the thesis. The first application is a diagnostic protocol for the classification of ECG signals, built upon the classification algorithm described in [ASSK07, GSK02]. In particular, the s.p.e.d. classifier relies on a subset of the features proposed in [ASSK07, GSK02] used in a LBP classifier, replacing the Quadratic Discriminant Function (QDF) of the original paper. The protocol has been proposed together with the RUB partner during the SPEED project and was implemented during my week-long stay in Bochum. Relevant is the study carried out to observe how communication, computation complexity and accuracy change by decreasing the number of bits used to represent the data involved in the computation and the number of features used. This is a crucial step where signal processing domain knowledge must be used to ease the s.p.e.d. implementation of the classifier in terms of efficiency and reduced complexity. The protocol is described in the papers [BFK+09a, BFK+09b, BFL+09].

The same protocol has also been implemented by using our secure implementation of a Neural Network (NN) [BFL+11]. In the paper we investigate the relationship between the representation accuracy of the to-be-processed signals (i.e., the number of bits representing the ECG features), the complexity of the proposed protocol and the classification accuracy. Finally we compare the NN protocol with the LBP protocol from a complexity point of view, getting interesting insights about the suitability of LBP-based and NN-based classification for efficient implementation in a SSP framework.

During my internship in Philips, the problem of evaluating the quality of signals (in particular ECG signals) in a privacy-preserving scenario has been addressed. To the best of our knowledge, the problem of measuring the quality of a signal in a s.p.e.d. framework has never been addressed before and also have a small number of solutions in the plain domain. Differently from the diagnostic protocol, where we implemented an existing plain protocol in the encrypted domain, the quality evaluation protocol was studied in order to propose a new and efficient solution tailored for a s.p.e.d. implementation. The new technique analyzes the amount of noise in a biomedical signal and is based on the analysis of the Signal-to-Noise ratio in small windows of the encrypted signal rather than the whole measurement. The analysis is based on the statistics of the raw signal window and a corresponding filtered signal. Similarly to the classification protocol, we provide a careful analysis of the accuracy of the proposed protocol. Finally, the communication complexity of the proposed protocol is analyzed thus proving its efficiency. The results are published in [BGL10, LGB12].

During the PhD, I also worked on the Italian Priv-Ware project supported by the MIUR (Ministero dellUniversità e della Ricerca) under Grant 2007JXH7ET. During the project we worked on privacy protection of the biometric data. Specifically we proposed a novel complete demonstrator based on a distributed biometric system that is capable to protect the privacy of the individuals. The implemented system computes the matching task in the encrypted domain by exploiting homomorphic encryption and using Fingercode templates [BBC+10a, BBC+10b]. The protocol has been implemented and tested in real applicative conditions. Experimental results show that this method is feasible in the cases where the privacy of the data is more important than the accuracy of the system and the obtained computational time is satisfactory. Moreover, in the biometric scenario, a full-GC protocol for iris matching has been proposed in [LSP+12] and we wrote two chapters regarding biometry matching for two books, the first one related to security and privacy in biometrics [Cam13], the second one for Digital Right Management

[HKL13].

Recently we proposed new building blocks, such as a protocol to evaluate the division between secret values [LB11], a protocol to order secret values [LF12] and a protocol that permits to evaluate a general function on secret input via piecewise linear approximation [PLB12].

These applications are not detailed in this thesis for sake of brevity and because out of the main scope of this thesis.

1.3 Publication List

To appear:

[HKL13] M. Barni and **R. Lazzeretti**, and C. Orlandi. Processing Encrypted Signals for DRM Applications. Chapter in *Hartung F., Kalker T., and Lian S. Digital Rights Management: Technology, Standards and Applications. CRC Press.*

[Cam13] **R. Lazzeretti** and P. Failla and M. Barni. Privacy–Aware Processing of Biometric Templates by Means of Secure Two-Party Computation. Chapter in *P. Campisi. Security and Privacy in Biometrics, Springer.*

[PLB12] T. Pignata and **R. Lazzeretti** and M. Barni. General Function Evaluation in a STPC Setting via Piecewise Linear Approximation. In *WIFS'12. International Workshop on Information Forensics and Security*, Tenerife, Spain, December 2–5, 2012.

2012:

[LGB12] **R. Lazzeretti**, J. Guajardo, and M. Barni. Privacy Preserving ECG Quality Evaluation. In *MM&SEC'12, Proceedings of ACM Workshop on Multimedia and security*, Coventry, UK, September 6–7, 2012.

[LSP+12] Y. Luo, S.C. Samson, T. Pignata, **R. Lazzeretti**, and M. Barni. An Efficient Protocol for Private Iris-Code Matching by Means of Garbled Circuits. In *ICIP'12, Special Session on Emerging Topics in Cryptography and Image Processing, International Conference on Image Processing*, Orlando, Florida, U.S.A., September 30 – October 3, 2012.

2011:

[LB11] **R. Lazzeretti** and M. Barni. Division between encrypted integers by means of garbled circuits. In *WIFS'11. International Workshop on Information Forensics and Security*, Foz do Iguacu, Brazil, November 29 – December 2, 2011.

[BFL⁺11] M. Barni, P. Failla, **R. Lazzeretti**, A.R. Sadeghi, and T. Schneider. Privacy-Preserving ECG Classification with Branching Programs and Neural Networks. In *IEEE Transactions on Information Forensics and Security*, Volume 6(2), pages 452 - 468, June 2011.

2010:

[BGL10] M. Barni, J. Guajardo, and **R. Lazzeretti**. Privacy Preserving Evaluation of Signal Quality With Application to ECG Analysis. In *WIFS'10. International Workshop on Information Forensics and Security*, Seattle, U.S.A., December 12–15, 2010.

[BBC⁺10a] M. Barni, T. Bianchi, D. Catalano, R. Di Raimondo, R. Donida Labati, P. Failla, D. Fiore, **R. Lazzeretti**, V. Piuri, A. Piva, and F. Scotti. A Privacy-compliant Fingerprint Recognition System Based on Homomorphic Encryption and Fingercode Templates. In *BTAS'10, IEEE Fourth International Conference on Biometrics: Theory, Applications and Systems*, Washington D.C., U.S.A., September 27–29, 2010.

[BBC⁺10b] M. Barni, T. Bianchi, D. Catalano, R. Di Raimondo, R. Donida Labati, P. Failla, D. Fiore, **R. Lazzeretti**, V. Piuri, A. Piva, and F. Scotti. Privacy-Preserving Fingercode Authentication. In *MM&Sec'10, Proceedings of ACM Workshop on Multimedia and Security*, Rome, Italy, September 9–10, 2010.

2009:

[BFL⁺09] M. Barni, P. Failla, **R. Lazzeretti**, A. Paus, A.-R. Sadeghi, T. Schneider, and V. Kolesnikov. Efficient Privacy-Preserving Classification of ECG Signals. In *WIFS'09. International Workshop on Information Forensics and Security*, London, UK, December 6–9, 2009.

[BFK+09a] M. Barni, P. Failla, V. Kolesnikov, **R. Lazzeretti**, A.R. Sadeghi, and T. Schneider. Secure Evaluation of Private Linear Branching Programs with Medical Applications. In *ESORICS'09, European Symposium on Research in Computer Security*, Saint Malo, France, September 21–25, 2009.

[BFK+09b] M. Barni, P. Failla, V. Kolesnikov, **R. Lazzeretti**, A.R. Sadeghi, and T. Schneider. Combining Signal Processing and Cryptographic Protocol Design for Efficient ECG Classification. In *SPEED'09 Workshop on Signal Processing in the Encrypted Domain, co-located with CHES'09, Workshop on Cryptographic Hardware and Embedded Systems 2009*, Lausanne, Switzerland, September 10, 2009.

2008:

[LB08] **R. Lazzeretti** and M. Barni. Lossless Compression of Encrypted Grey-Level and Color Images. *SPEED'08, Special session on Signal Processing in the Encrypted Domain, co-located with EUSIPCO'08, European Signal Processing Conference*, Lausanne, Switzerland, August 25–29, 2008.

1.4 Outline

This thesis is subdivided in three parts.

The first part introduces the s.p.e.d. setting and the main available tools. A reader that already knows the topic and is interested to its application to the remote biomedical analysis can skip over this part and start reading from the second part. In Chapter 2 we present some general notions related to s.p.e.d. , such as the typology of the attackers and the security models, some cryptographic primitives, blinding and the way data needs to be represented to be correctly used in privacy preserving protocols. Homomorphic Encryption, together with many basic homomorphic protocols, is presented in Chapter 3. Moreover the innovative Fully Homomorphic Encryption recently proposed in [Gen09] is introduced at the end of the chapter. Oblivious transfer protocols are presented in Chapter 4. OTs are fundamental protocols that permit a party to select among two secrets owned by another party. In this thesis oblivious transfer is only used into Garbled Circuits to exchange input secrets. Chapter 5 introduces Garbled Circuits and the main basic circuits that are needed for the implementation of privacy preserving protocols. The circuit for integer approximation of logarithm is particularly relevant, since it represents one of the innovative contributions of this thesis and is further used in the protocol for the evaluation of signal quality. The first part ends with the analysis of pros and cons of the HE and GC approaches to s.p.e.d. and with the introduction of Hybrid Protocols in Chapter 6.

The second part focuses on privacy preserving biomedical analysis. We propose two different implementations of the ECG classification algorithm proposed by Ge et al. [GSK02], presented in Chapter 7 together with an introduction to Electrocardiogram. The first solution, shown in Chapter 8, faithfully replicates Ge's algorithm in the encrypted domain. The classification algorithm can be related to Linear Branching Programs. The second solution, addressed in Chapter 9, replaces the classification algorithm used in Ge's paper with a Neural Network. A comparison of the two solutions, provided at the end of Chapter 9, demonstrates that our efficient implementation of privacy preserving NN provides the same accuracy of the LBP solution with lower complexity.

In the third part of the thesis we describe a protocol to evaluate the quality of an ECG signal in a privacy preserving scenario. After a description of the noise affecting an ECG signal, Chapter 10 proposes the protocol to evaluate the SNR of a no-referenced signal by using s.p.e.d. techniques, together with an analysis of the protocol complexity and the error due to the integer implementation of the logarithm. The SNR protocol is then used to develop another protocol, described in Chapter 11, that uses it to compute the SNR of the segments the signals is subdivided in. The obtained SNRs are then used to extract some features to be used as input to a linear classifier able to assert if the signal quality is sufficient for further computation.

The thesis ends with some conclusions in Chapter 12, where we also provide some hints for future works.

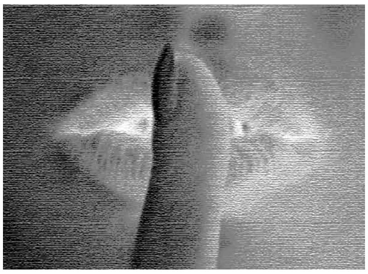
Part I

Cryptographic tools

In a classical distributed processing scenario, two or more entities, such as a service provider and a user, communicate to obtain a common goal. Normally the entities involved trust each other and have no problems to reveal their data, but are interested to prevent that a third party, such as an eavesdropper or an attacker, can observe the communication or interfere with it, changing the transmitted data or denying the service. To protect the communication the involved entities have to negotiate the communication so that a secure channel, where the services are guaranteed, is created. Guaranteeing the security of the data is not a easy job. First of all we have to face different types of adversaries. Luckily a big effort has been applied to develop with symmetric and asymmetric cryptographic primitives in the past.

On the other side, the security provided by classical protection schemes are no more sufficient when even the adversaries are involved in the computation and, for example, the user needs to protect his data also from the service provider. In recent years some Multi-Party Computation (MPC) tools, such us Homomorphic Encryption, Oblivious Transfer, Garbled Circuits, have been proposed to allow the cooperation between parties that do not trust each other. Moreover Hybrid protocols permit to compose subprotocols developed by using different tools so that efficient solutions to real problems can be developed.

Given the possibility to elaborate private data without disclosing it, it is interesting the application of the idea to the processing of private signals. In this part of the thesis, we present the basic notions necessary for the development of protocols relative to Signal Processing in the Encrypted Domain.



"Seest thou not, Rutilio, that he which discloseth a secret unto another, praying him to say nothing because it imports his life, is a fool?"
(*Miguel de Cervantes: The Travels of Persiles and Sigismunda*)

Chapter 2

Preliminary Notions

In this chapter we introduce the notions necessary to fully understand the topic of the thesis. We start in Section 2.1 introducing the typology of attackers. Then, in Section 2.2, the services that a general cryptographic protocol must provide for a secure communication are presented, while in Section 2.3 we show the security models that can be used to demonstrate the security of any privacy preserving application and cryptosystem.

Given the focus of this thesis on privacy protection in signal processing applications, we analyze signal representation in Section 2.6 with the goal of finding a representation usable with the available cryptographic primitives. To conclude, in Section 2.5, we describe blinding techniques, useful to hide signals, or variables, each time that there is the necessity to decrypt them, without revealing their values.

2.1 Adversaries

As shown in Figure 2.1, a simple communication involves many entities: a sender, a receiver and possibly an attacker. A more realistic scenario involves many parties that in turns act as sender and/or receiver and more than one attacker can be present.

The attacker can be considered either passive or active. A *passive attacker* is an eavesdropper monitoring the transmission to extract information, that can be the content of the transmission, or some statistical analysis derived from it, such as the identities of sender and receiver, the quantity of data transmitted, the type of data, etc. A passive attack is difficult to detect, because it does not involve any alteration of the data, but can be prevented by encrypting the transmission. An *active attacker* is an entity interested in interfering with the communication. The principal active attacks are the

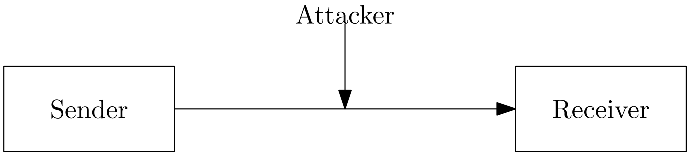


Figure 2.1: Entities involved in a simple communication.

following:

- **Masquerade attack**: the attacker is interested to impersonate one of the players without that the other notes the substitution. When during a communication from Alice to Bob the attacker takes the place of both of them(it acts as Bob for Alice and as Alice for Bob), it is said “Men in the middle”.
- **Replay attack**: the attacker captures a message and subsequently retransmits it to produce an unauthorised effect.
- **Modification of messages**: part of messages, or whole messages, are changed to produce unintended effects.
- **Denial of service**: the attacker prevents the normal use of communication facilities stopping a communication or saturating the channel or the capacity of a server.

Detecting the presence of an active attacker is easy, but a complete prevention of active attacks is quite difficult.

In a privacy preserving scenario, even the parties normally involved, say Alice and Bob (or client and server), can be considered attackers interested to obtain information about the other player. Hence they can be classified according to the following classes:

- **Honest**: an honest party is simply interested to performing the correct computation and obtain the final result, if it is the recipient, or helping the other party to obtain it.
- **Honest but curious**: this kind of party follows the protocol without deviating from it, but in the meanwhile it is interested to obtain any information it can observe about the other party.

- **Malicious**: a malicious party can deviate from the protocol or use particular sequences of inputs to extract information about the secret inputs or outputs of the other party.
- **Covert**: a covert party is willing to act as a malicious party only if it is sure that its malicious actions are not discovered, otherwise it acts as an honest but curious party.

2.2 Security services

In a classical distributed scenario, two or more entities, such as a service provider and a user, communicate to obtain a common goal. Normally the entities involved trust each other and have no problems to reveal their data, instead they are usually interested to prevent that a third party, such as an eavesdropper or an attacker (described in Section 2.1), observes the communication or interfere with it, changing the transmitted data or denying the service. To protect the communication the involved entities have to negotiate the communication so that a secure channel is created, where the services are guaranteed. For completeness, in this section the services that each security system must provide are reviewed, as defined in ITU-T[1] Recommendation X.800. Considering that in this thesis we focus on a scenario involving entities that do not trust each others, in the next chapters we suppose that a secure channel has already been established, without describing the protocols to do that.

2.2.1 Authenticity

Authentication is concerned with the assurance that communication is between entities that are who they claim to be and is generally achieved by using digital signatures. There are two types of authentication requirements that must be satisfied.

- **Single party authentication:** this is a property of systems where the sender and receiver are well distinct and the receiver has the necessity

---

[1] International Telecommunication Union (ITU) - Telecommunication standardization sector

to verify that the sender is the one it claims to be. This property is also of importance for systems concerning with multimedia and signal data.

- **Mutual authentication:** this kind of authentication is used for protocols where all participants must prove their identity to avoid transmissions from/to unauthorized users.

2.2.2 Access Control

This is the ability to limit and control the access to host systems, applications and data for each individual user or group. To achieve this, each entity trying to gain access must be first identified, or authenticated.

2.2.3 Data Confidentiality

This service prevents the access of data from unauthorised disclosure, introducing the protection from passive attacks, so that information is available only to authorised users.

2.2.4 Data Integrity

This service protects data against active attacks assuring that the data is not changed by an unauthorised entity and, depending on the application, can be grouped into the following two categories:

- **Connection-less:** this service controls the integrity of each data block detecting any modification.

- **Connection oriented:** this is a stronger service concerned with stream of messages and assures that message blocks are received as sent, with no duplication, insertion, modification, deletion or reordering.

Since signals are inherently ordered sequences of values, connection-oriented integrity must be guaranteed, preserving the correct time order of the signal samples and checking that the data received from a server is not forged (coming from a different source, or previously recorded).

2.2.5 Non-repudiation

This service prevents either sender or receiver from denying to have sent a transmitted message. Non-repudiation proofs may regards with both the sender, which can not deny to have sent a certain message, and the receiver, who can not deny that the message has been correctly received.

2.3 Security Models

A security model is a formal description of a security policy and is normally used to prove the security of a protocol. The main security models are the standard model and the random oracle model.

2.3.1 Standard Model

The standard model is a model of computation in which the adversary is only limited by the amount of time and computational power available. Cryptographic schemes are usually based on complexity assumptions, which state that some problems, e.g. factorization, cannot be solved in polynomial time. Schemes which can be proven secure using only complexity assumptions are said to be secure in the standard model. Security proofs are notoriously difficult to achieve in the standard model, so in many proofs, cryptographic primitives are replaced by idealized versions, such as the random oracle model, described later. Other models invoke trusted third parties to perform some tasks without cheating: for example, the public key infrastructure (PKI) model requires a certification authority, which if it were dishonest, could produce fake certificates and use them to forge signatures, or mount a man in the middle attack to read encrypted messages.

2.3.2 Random-Oracle Model

A Random-Oracle is a mathematical function (a theoretical black box) mapping every input to a (truly) random output chosen uniformly from its output domain. A random oracle is a theoretical model of a perfect cryptographic hash function, used in cryptographic proofs; it is typically used when no known implementable function provides the mathematical properties required by the proof. A system that is proven secure using this kind of proofs is said to be secure in the random oracle model, as opposed to security in the standard model. In practice, random oracles are typically used to model cryptographic hash functions in schemes where strong randomness assumptions are needed of the hash function's output. Such proofs generally show that a system or a protocol is secure by showing that an attacker must require impossible behavior from the oracle, or solve some mathematical problem believed to be hard, in order to break the protocol. Impagliazzo and Rudich [IR89] showed the limitation of random oracles - namely that their existence alone is not sufficient for secret-key exchange. Bellare and Rogaway [BR93] advocated their use in cryptographic constructions, where the random oracle produces a bit-string of infinite length which can be truncated to the length desired. When a random oracle is used within a security proof, it is made available to all players, including the adversaries. No real function can implement a true random oracle. In fact, certain artificial signature and encryption schemes are known which are proven secure in the random oracle model, but which are trivially insecure when any real function is substituted for the random oracle [CGH04]. Nonetheless, for many protocols a proof of security in the random oracle model gives very strong evidence that an attack which does not break the other assumptions of the proof, if any (such as the hardness of integer factorisation), must discover some unknown and undesirable property of the hash function used in the protocol to succeed.

2.4 Cryptographic Primitives

To achieve the functionalities described in the previous section, several cryptography blocks can be used. Although these blocks do exist and are used very often in different protocols, their application in a signal processing framework is challenging and they must sometimes be slightly modified.

2.4.1 Symmetric Key Cryptography

Symmetric key cryptography is one of the oldest primitives and provides the confidentiality service. The most important properties of these algorithms are ease of operation and high speed. In these Systems both the sender and the receiver have a common secret key which is used for encryption and decryption of messages. It is assumed that decrypting a message is easy when the key is known and otherwise difficult. Indeed the secret key creates a secure communication channel between sender and receiver as shown in Figure 2.2.

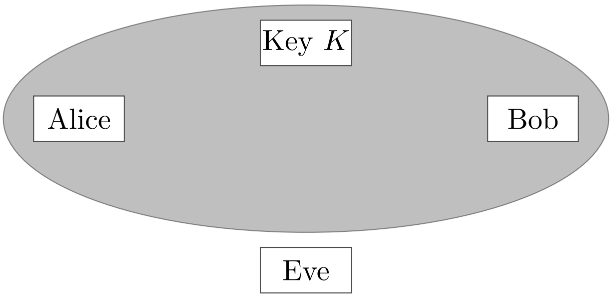


Figure 2.2: The typical scenario for symmetric key cryptography. Alice and Bob are legitimate users of the system, whereas Eve is malicious and wants to eavesdrop the channel. The key K is the common secret key between Alice and Bob and creates a secure channel for communication of legitimate users.

Some important attacks on symmetric key systems are: known plaintext attacks, which use pairs of plaintext and ciphertext to unveil some secret information, and chosen plaintext attacks, in which the user can generate the encryptions of arbitrary plaintexts to break the system.

There are several kinds of symmetric key systems proposed in the literature and used in practice like triple DES and AES (see [DR99] and [Bar04]).

2.4.2 Public Key Encryption

Symmetric key systems are useful and efficient, but their application requires generally special infrastructures to be setup. Some examples are setting up the initial keys or managing keys among several users. A solution is a public key system in which encryption and decryption are performed using two different keys. The first one is published, whereas the second one must be kept secret.

The first efficient public key encryption scheme was the RSA system [RSA78], based on the difficulty of factoring large composite numbers.

An important class of public key cryptosystems are systems based on probabilistic encryption proposed for the first time in [GM84]. In these systems a given plaintext is encrypted to a different message at each new encryption. This is useful when, e.g., encrypting and transmitting single bits: if the used encryption was deterministic, the adversary could easily understand the meaning of encrypted messages by using the public key to encrypt the bits zero and one and comparing the obtained cyphertexts with the one transmitted.

The method proposed in [GM84], also known as Goldwasser-Micali encryption system, is the first probabilistic encryption system which is provably secure under cryptographic assumptions. The security of this system is based on the intractability of the quadratic residuosity problem modulo a composite number $N = pq$, when the factorization of N is unknown. The public key of the system is (x, N), where x is a pseudo-square modulo N, i.e., $x^{(p-1)/2} \equiv -1 \mod p$ and $x^{(q-1)/2} \equiv -1 \mod q$, and the secret key is the pair (p, q), the factorization of N. To encrypt a bit m_0, Alice computes $c = y^2 x^{m_0} \mod N$, for a random y and sends it to Bob. Bob computes the values $c^{(p-1)/2} \mod p$ and $c^{(q-1)/2} \mod q$ and decrypts $m = 1$ if the values are -1 and $m = 0$ if the values are 1. This system is not efficient by itself, since every bit is expanded to several bits, but it has a homomorphic property: if c_0 and c_1 are some encryptions for m_0 and m_1, respectively, then $c_0 c_1 \mod N$ is an encryption of $m_0 \oplus m_1$. We will talk more about homomorphic encryption systems in the next section. The Goldwasser-Micali cryptosystem is sometimes used in larger protocols like in [BCI+07].

One approach to achieve a probabilistic encryption from a deterministic one is the OAEP (Optimal Asymmetric Encryption Padding) introduced by [BR94] using paddings and one-way permutation functions to build a probabilistic version of RSA. The proposed method was claimed to be secure under one-wayness of the underlying permutation, but this claim has been contradicted. The authors of [FOP+01] showed the security of the mechanism in the random oracle model.

The most popular public key cryptosystems are the ElGamal cryptosystem (see Section 3.2), proposed in [ElG85] that is based on the intractability of discrete logarithm in finite fields with large prime number of elements, and the Paillier cryptosystem (see Section 3.3), introduced for the first time in [Pai99] and based on the difficulty of deciding if a number is an n-th power in $\mathbb{Z}_N$, for a large enough N.

Formally a Public Key Encryption Primitive (PKEP) is a triple $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ of algorithms: $\mathcal{G}$ and $\mathcal{E}$ are probabilistic while $\mathcal{D}$ is deterministic. Let p_1 and p_2 be polynomials specified by the PKEP.

- for every n and every $r \in_R \{0,1\}^n$, generator $\mathcal{G}(r)$ outputs a pair of keys (sk, pk);
- for every $m \in \{0,1\}^{p_1(n)}$, each string $r' \in \{0,1\}^{p_2(n)}$ of coin flips of encryption $\mathcal{E}$, and pair (sk, pk) output by $\mathcal{G}$, it holds that decryption $\mathcal{D}(sk, \mathcal{E}(pk, m, r')) = m$;

where $p_1(n)$ and $p_2(n)$ indicate two values related to n, while r' is a random number introduced in each encryption so that different encryptions of the same message result in different cyphertexts.

There are several security notions regarding a PKEP like: *Indistinguishability under chosen-plaintext attack* (IND-CPA), *Indistinguishability under chosen ciphertext attack/adaptive chosen ciphertext attack* (IND-CCA1, IND-CCA2), which we briefly describe below:

- **IND-CPA:** This is generally used for probabilistic encryption schemes and means that an *adversary*, being allowed to do encryptions of his own, sends two (random) messages M_0 and M_1 to a *challenger*. The challenger encrypts one of the messages and sends it back to the adversary. The encryption scheme is indistinguishable under chosen plaintext attack, if the adversary cannot distinguish, better than random, which of the messages has been encrypted by the challenger. The definition can be relaxed to a deterministic scheme if the adversary is not allowed to encrypt M_0 and M_1.
- **IND-CCA1:** This is the property of indistinguishability under adaptive chosen ciphertext attack. This is very similar to IND-CPA with the difference that the adversary has also access to a decryption oracle. Here the adversary is allowed to call the encryption and decryption oracles only until he sends the messages M_0 and M_1 to the challenger, that returns the encryption of one of the messages. The adversary should not be able to decide which message has been encrypted. This means that decryption previously performed can not help the adversary.

- **IND-CCA2:** This is the adaptive version of IND-CCA1 where the adversary is allowed to continue encrypting and decrypting even after receiving the message, with the constraint that he is not allowed to decrypt the received message, even if he can decrypt any other message. Security means that decryptions of other messages provide no information useful to understand what message has been encrypted.

Many public key encryption schemes have homomorphic properties and can be used as a tool in s.p.e.d. applications. An homomorphic property permits to perform an operation $\phi_2(\cdot)$ between encrypted numbers that returns, after decryption, the same result of the operation $\phi_1(\cdot)$ performed on the same values in the plain domain:

$$\phi_1(a, b) = \mathcal{D}(\phi_2(\mathcal{E}(a), \mathcal{E}(b))). \tag{2.1}$$

The most popular homomorphic cryptosystems permit to evaluate the sum among cyphertexts [Pai99], while El-Gamal cryptosystem [ElG85] permits to evaluate the product. Recently, some fully homomorphic cryptosystems (that permit to compute sums and products under encryption) have been proposed ([Gen09] is the first one), but they have not efficiently implementation. Homomorphic schemes are analyzed in detail in Chapter 3.

Digital Signatures

Digital signatures are an asymmetric cryptographic primitive to provide message authentication, and can be seen as the digital equivalent of handwritten signatures. Unlike handwritten signatures however, a digital signature is different for each message that is being signed, because otherwise signatures could easily be copy-pasted from one electronic document to the next.

The signer generates a key pair consisting of a private and a public key. The public key is made available and associated with the signer's identity, e.g. through a public-key infrastructure (PKI). The signer uses his private key to compute the signature on a message; the verifier can later use the corresponding public key to check that the candidate signature is indeed valid for the given message.

The most generally security definition for digital signatures is that of existential unforgeability under chosen-message attack [GMR88]. This means that no adversary who's given a public key as input and can see as many signatures as he likes on messages of his choice, can output a valid forged signature on a message for which he has not yet seen a signature.

Various types of signature schemes exist. The two most common schemes in practice are RSA [RSA78], based on the hardness of factoring large integers, and DSA [DSA00], based on the hardness of discrete logarithms. A number of schemes exist that are provably secure in the random oracle model, including RSA-PSS [BR96] based on the one-wayness of RSA, Schnorr signatures [Sch90, PS00] based on the hardness of discrete logarithms, and BLS signatures [BLS01] based on the computational Diffie-Hellman problem [DH76, Bon98, NP01] in groups with bilinear maps. The latter has the advantage of producing very short signatures. Schemes that are provably secure in the standard model [GHR99, CS99, BB04] exist as well, but are less efficient than their random-oracle counterparts.

2.4.3 Cryptographic Hashes

Hash functions are important cryptography blocks and have several applications. They are generally used to generate short messages from long ones in such a way that the receiver can detect any modifications in the message and thus control its integrity. They can be grouped into two categories: MACs (message authentication codes) and public or unkeyed hashes.

MACs are hash functions which are based on symmetric encryption systems. A user who sends the information produces a secure hash by using a secret key shared with the receiver. The receiver produces the same hash using the same key. If the new hash matches the ones transmitted by the sender, he can control both the authenticity and the integrity of the message.

Unkeyed hash functions, on the other hand, are used to produce short strings from large ones and play an important role in most cryptographic systems. One typical application for such hash functions is in digital signatures where the hash function produces a short representation of the message, and then this short representation is signed. The most important property of a hash function is to be collision-free, meaning that it is computationally infeasible to find two messages with the same hash functions. In the example of digital signatures we see that if a collision occours, then the same signature can be used for both the messages. Some popular examples for hash functions are the (actually insecure) MD5 (Message Digest Version 5) [Riv92] and the SHA family [Rob95], that is periodically improved to guarantee the security.

The underlying property of hash functions is that even modifications of one bit, can result in a completely different hash. It is difficult to find a collision or two different messages having the same hash value. This ensures that a malicious user who modifies the message cannot find a valid hash value.

2.4.4 Secure Multi-Party Computation

Secure multi-party computation (SMPC) [Yao82, GMW91] allows a set of mutually mistrusting parties to compute a function together while keeping their inputs private. Mistrust among parties is usually modelled by assuming the existence of an adversary that is allowed to corrupt some sub set of the parties. By corruption, one usually assumes that the adversary can read (and possibly modify) the internal memory of the corrupted players. The weak reliability of communication is modelled by allowing the adversary to control the communications involving corrupted players. The SMPC paradigm allows many settings and concerns to be modelled and is a strong tool to show that solutions exist to very general cryptographic problems. The power of the framework is that under partial corruption assumption (and various settings and constraints) it is possible to compile any polynomial size function into a protocol that ensures input privacy. Input privacy is assured facing an adversary that is assumed to control the entire state (memory) of corrupted parties (passive adversary) and one that in addition may corrupt the memory arbitrarily (malicious adversary).

The notion of SMPC is very general and allows numerous variants. A basic distinction is between the *computational* setting [GMW91] where all communication is available to the adversary, and the *information-theoretic* setting [BOGW88, CCD88, RBO89], where point-to-point communication links are completely protected, but the adversary is not restricted to probabilistic polynomial time.

Indeed general multiparty computation protocols allow to securely compute any function but the efficiency of the computation is linear in the size of the circuit implementing the function to be computed. This often results in protocols that are simply too inefficient to be used in practice. For this reason, efficient ad-hoc solutions have to be designed to solve specific cryptographic problems, by choosing the best protocol for each functionality involved and by tailoring the bitsize of the data.

A specific case of SMPC that is particularly interesting for this thesis is secure two-party computation (STPC) where only two entities are involved. This entities are normally identified as the server $\mathcal{S}$ and the client $\mathcal{C}$, a user that needs to access to a functionality provided by the server.

2.5 Additive and multiplicative blinding

During a STPC, as we will show in the next chapter, whenever $\mathcal{S}$ can not compute some function in the encrypted domain or the protocol requires to change the data representation, it is necessary to resort to some form of interaction between the server $\mathcal{S}$ and the client $\mathcal{C}$. The simplest way to use interaction between the client and the server to apply a non-linear function to private data is through obfuscation (sometimes referred to as blinding).

Let us consider a scenario in which $\mathcal{S}$, at some point in the protocol, wants to perform a non-linear operation on the encrypted data. A very simple protocol could be devised in the following three steps: 1) $\mathcal{S}$ obfuscates the intermediate result and sends it back to $\mathcal{C}$; 2) $\mathcal{C}$ decrypts it, performs the operation, re-encrypts the result, and sends it to $\mathcal{S}$. 3) finally $\mathcal{S}$ removes the obfuscation. The above protocol requires the following requirements to be met in order to work (at least in the semi-honest scenario).

1. *The obfuscation should be possible on encrypted values*: only linear operations, sums and products with random numbers, can be used to blind a value.

2. *The obfuscation should preserve the meaning of the non-linear operation*: in the case we are interested to evaluate a non-linear operation $\phi(x)$, we say that an obfuscation $y = r_a x + r_b$ preserves the meaning of $\phi(x)$ if there exists a corresponding (possibly non-linear) operation $\psi(y)$ such that
$$\phi(x) = \alpha(r_a, r_b)\psi(y) + \beta(r_a, r_b)x + \gamma(r_a, r_b) \tag{2.2}$$

where α, β, γ can be arbitrary functions of the random variables r_a, r_b. The rationale of the above definition is that $\mathcal{S}$, who knows r_a, r_b and receives the encryption of both x and $\psi(y)$, should be able to compute the encryption of $\phi(x)$ by relying on homomorphic properties.

3. *The obfuscation should be secure, i.e., $\mathcal{C}$ should be able to infer from the obfuscated value neither the true value of the input nor the true result of the non-linear operation*: a possible approach is to consider the security in an information theoretic sense. Let us consider the mutual information between the true value x and the obfuscated value y, given by [CT06]

$$I(X;Y) = \sum_{x,y} p_{X,Y}(x,y) \log \frac{p_{X,Y}(x,y)}{p_X(x) p_Y(y)} \tag{2.3}$$

   where $p_X(x)$ and $p_Y(y)$ are the marginal probability distribution functions of X and Y respectively, while $p_{X,Y}(x,y)$ is the joint probability distribution function of X and Y.

   Perfect or unconditional security is obtained if there exists a choice of r_a, r_b such that $I(X;Y) = 0$ [Sha49]. Perfect security ensures that $\mathcal{C}$ can not discover anything about x by observing either y and $\psi(y)$. Moreover, it also ensures that nothing can be inferred about $\phi(x)$. This can be verified by noting that $\phi(x) \to x \to y \to \psi(y)$ forms a Markov chain and by applying the data processing inequality for mutual information [CT06].

In the setting described above, summarized in Protocol 1, perfect security can not in general be achieved.

However, it would be interesting to quantify the information leakage on both x and $\phi(x)$, so as to define appropriate security measures on the obfuscation of x. In [BAS+09] a theoretical framework to measure the secure leakage of a particular blinding strategy is proposed and applied to evaluate the security of additive and multiplicative blinding.

In [BAS+09] two parties are identified: Alice, that holds some secret signal value $x \in \mathbb{R}$, and obfuscates it by applying an obfuscation function which yields the obfuscated value $y \in \mathbb{R}$, and Bob (the attacker) that, given the

Function evaluation by using obfuscation.

inputs of $\mathcal{C}$: Nothing
inputs of $\mathcal{S}$: $[\![x]\!]$
output for $\mathcal{C}$: Nothing
output for $\mathcal{S}$: $[\![\phi(x)]\!]$

client $\mathcal{C}$ | server $\mathcal{S}$

$$[\![y]\!] = [\![r_a x + r_b]\!] = [\![x]\!]^{r_a}[\![r_b]\!];$$

$$\xleftarrow{\quad [\![y]\!] \quad}$$

decrypts $[\![y]\!]$;
computes $\psi(y)$;
encrypts $\psi(y)$;

$$\xrightarrow{\quad [\![\psi(y)]\!] \quad}$$

$$\begin{aligned} &[\![\phi(x)]\!] = \\ &= [\![\alpha(r_a, r_b)\psi(y) + \beta(r_a, r_b)x + \gamma(r_a, r_b)]\!] = \\ &= [\![\psi(y)]\!]^{\alpha(r_a, r_b)}[\![x]\!]^{\beta(r_a, r_b)}[\![\gamma(r_a, r_b)]\!] \end{aligned}$$

Protocol 1: Function $\phi(x)$ evaluation by using obfuscation with homomorphic encryption scheme.

obfuscated value, tries to discover the true value x by applying an estimation function producing an estimate $\tilde{x}$.

In [BAS+09] the objective is to evaluate how close can Bob's estimate be with respect to the true value x, by using a measure derived by the mutual information. The signal x is assumed to have mean μ and variance σ^2.

The simplest blinding technique is the addition of a random variable to the signal values. The security depends on both the distribution of the signal values and the distribution of the blinding factor. The obfuscation model under a single observation is

$$y = b + x \tag{2.4}$$

where b is a real blinding value statistically independent from x.

Another useful blinding technique consists in multiplying the signal values by a random factor. In the scalar case the model is

$$y = bx \tag{2.5}$$

where b and x are mutually independent.

The analysis performed in [BAS+09] permits to evaluate the security achievable by the two different blinding techniques. The first thing noticed is that multiplicative blinding requires in general a much higher number of bits in order to achieve the same security level. Moreover for additive blinding the security grows exponentially with the number of bits used for the obfuscation values, while for multiplicative obfuscation the security grows only linearly. This means that to increase the security of additive blinding it is sufficient to add a few bits, while to obtain the same result with multiplicative blinding the bitlength of the random value has to be multiplied by a constant factor.

2.6 Signal representation

Before describing the available s.p.e.d. tools in the next chapters, we address the problem of the definition of a proper signal representation model. This includes finding a way to represent signals by using the finite size ring arithmetic made available by all practical cryptosystems.

In this section, some possible ways to represent signals are discussed and the trade-offs that need to be considered when choosing a proper representation introduced. The considered trade-offs include: storage requirements, representation accuracy (linked to the quantization error made when passing from real to integer numbers), computational complexity.

2.6.1 Approximating real numbers on finite rings

Signals are customarily represented by real-valued sequences. Real numbers have dynamic structures such that the length of their representation strings generally increases during the computations. On the other hand, traditional encryption schemes work with strings of limited length, where these lengths are governed by security parameters. This means that if the traditional encryption schemes, which are based on ring of integers or polynomials, are used to perform homomorphic operations on encryption of real numbers, the security parameter should be infinitely large.

The problem of representing real numbers by strings of limited length is not specific to cryptography. Traditional computing systems use truncations of real numbers for storing and representation purposes. In the following we describe several approaches to represent real values in computing systems and analyze their adaption to s.p.e.d. applications.

Floating Point Representation

A wide range of real numbers can be represented by few bytes, when the floating point representation is used. In the natural basis $\beta \in \mathbb{N}$, the floating point representation of the number x is the pair (s_x, e_x), where s_x and e_x are called *significand* and *exponent*, respectively, meaning

$$x \approx s_x \beta^{e_x} \tag{2.6}$$

The significand and exponent are selected in such a way that the approximation error in (2.6) is as small as possible. Floating point representations use bit-strings of lengths ℓ_s and ℓ_e to store the significands and exponents, respectively. We denote such a floating point representation by $\mathcal{FLP}(\ell_s, \ell_e)$. These values are then interpreted as the signed integer representation of e_x, whereas that of s_x is a decimal value such that, per agreement, the decimal point is at the leftmost position. It can be shown that when the value x is represented by (s_x, e_x) in $\mathcal{FLP}(\ell_s, \ell_e)$, then the absolute value of the approximation error in (2.6) is bounded by

$$\beta^{e_x-(\ell_s-1)} \tag{2.7}$$

and hence to decrease the error, it is desirable to make e_x as small as possible. This results in the *normalized* representation of floating point numbers for which the leftmost digit of the significand is always non zero.

Multiplication of floating point numbers is straightforward: multiply the significands, add the exponents, and normalize the result. Addition is, on the other hand, more complicated. To add two numbers we have to find the maximum of the two exponents, adjust the significands in such a way that both of them have the common exponent, add the significands together and finally normalize the result.

To perform these operations using homomorphic encryption, we can use a multiplicative homomorphic encryption for the significands and an additive one for the exponents. In this way the multiplication would be easy to compute, but still requiring truncation of significands during the normalization and checking if the leftmost digit is zero. Addition of two values would need the comparison of two encrypted values and, generally, truncating the results to bring them into the larger exponent and finally normalization.

Given the difficulties of implementing the above operations in the encrypted domain without resorting to interactive protocols, we can conclude that the floating point representation is a not suitable model for s.p.e.d. computation.

Fixed Point Representation

A less space-efficient (but simpler) approach to represent numbers is to use fixed point representation. This method in its original form, is characterised by the scaling factor (quantizer) q. The fixed point representation X of a real number x is obtained by computing

$$X = \lfloor qx \rceil \approx qx, \tag{2.8}$$

i.e. by multiplying the value x by a coefficient q and rounding it to the closest integer number. Given the fixed point representation, if q has been chosen big enough such that qx carries sufficient information about x, an approximation $\tilde{x} = X/q$ of the original real number x can be obtained.

For simplicity q is usually chosen such that $q = \beta^p$ where $\beta \in \mathbb{N}$ (usually $\beta = 2$ or $\beta = 10$) and p is the precision desired.

Let us denote the fixed point representation of x by $\mathcal{FIP}_q(x)$. Given an upper-bound for x, its value is represented by a limited number of digits. Assuming that $|x| < q$, the absolute value of integer representation of numbers in $\mathcal{FIP}_q(x)$ are smaller than q^2, whose representation bitlength we denote by ℓ.

Computation using fixed point representation is easier than in floating point. Addition of values is carried out by adding the corresponding integers and the representation bitlength is $\ell + 1$. Multiplication is still a multiplication between integers followed by truncation. The truncation stage can be explained as below. Let X and Y be the representations of x and y in $\mathcal{FIP}_q(x)$, i.e., $X \approx qx$ and $Y \approx qy$, hence

$$XY \approx q^2 xy \tag{2.9}$$

where the valid representation of xy is an integer Z such that $Z = qxy$. It is not difficult to see that a suitable candidate for Z can be computed by dividing XY by q and taking the quotient. Although even this simple task is not easy to implement in the encrypted domain, it seems that fixed point representation is more suitable than floating point representation for computation in the encrypted domain.

Indeed, one of the strongest arguments against fixed point representation is the large number of bits needed to achieve the same accuracy of floating point representation. We argue here that this argument cannot be used in s.p.e.d. applications, since here, we already need long representations to achieve a high security level. As one of the most famous examples for additive homomorphic encryptions, we mention the Paillier encryption scheme [Pai99]. One of the most important parameters in this system is the integer N, where encrypted messages belong to the ring $\mathbb{Z}_{N^2}$ and the plaintext X is assumed to be smaller than N. N is generally called the *modulo number* in Paillier encryption. On the other hand, according to (2.9), we see that even for permitting a single multiplication, the length of N must be larger than twice the representation length, i.e., the length of q^2, hence $\mathcal{FIP}_q(x)$ can be used together with Paillier encryption with *modulo number* N as long as

$$q^4 < N \tag{2.10}$$

To correctly represent the result of a product. Assuming $q^4 = N$ and the fact that the maximum error of $\mathcal{FIP}_q(x)$ is q^{-1} we see that using a fixed point representation and the Paillier scheme with modulo number N, the error can be reduced to $N^{-1/4}$.

Fixed point representation without truncation

The fixed point representation provides a reasonable method to approximate arithmetic operations on real numbers, but after each multiplication a division by the quantizer must be performed. We are not aware of any method to efficiently perform this operation on encrypted numbers. One possibility is to select a *modulo number* large enough to perform all operations without any truncation. The final result is then truncated after being decrypted. This could require enormous values of N which are not practical.

To solve this problem we observe that the accuracy of a floating point representation varies over the range of represented numbers, as can be observed from (2.7), whereas accuracy of fixed point representation is fixed over the whole range of definition.

This means that the values of N can be made smaller for the same arithmetic accuracy. This smaller values of N are appropriate for special problems and must be selected according to the exact protocol which is used to solve the problem.

Other representations

Other approaches to represent the signals have been proposed.

A method to represent real numbers is to approximate them by means of rational numbers, whose numerator and denominator, in turn, are represented by integers. Such an approach is described in [FSW02] where the homomorphic properties of the Paillier cryptosystem are exploited. This method has a better accuracy than fixed point representation, the only problem is that in this method even the addition of two values can cause the denominators to grow quickly and hence before any further computation truncation is required, involving a highly non-linear operation that can not be performed without interaction.

An alternative representation for signals in the encrypted domain could be obtained by approximating a real or complex value over a ring of algebraic integers [GAM85]. Algebraic integers offer some advantages with respect to both integer and rational representations: for example, they are dense over $\mathbb{C}$, meaning that the quantization does not require any scaling factor, and the dynamic range of the quantized values is dramatically reduced for a given error tolerance. Let $\omega = e^{2\pi j/R}$, $R = 2^{\mu}$, $\mu \geq 2$ be a primitive Rth root of unity. The subring of the field of complex numbers $\mathbb{C}$ generated by ω over the integers $\mathbb{Z}$ is defined as

$$\mathbb{Z}[\omega] = \left\{ \sum_{i=0}^{R/2-1} \alpha_i \omega^i \,\middle|\, \alpha_i \in \mathbb{Z} \right\}. \tag{2.11}$$

The set $\mathbb{Z}[\omega]$ is usually referred to as the ring of *algebraic integers*. Both addition and multiplication can be defined over $\mathbb{Z}[\omega]$ as combinations of the integer coefficients α_i. If $\mu = 2$, then $\mathbb{Z}[\omega] \equiv \mathbb{Z}[j]$ is the ring of *Gaussian integers*, i.e., it contains numbers in the form $a + jb$, $a, b \in \mathbb{Z}$. Hence, approximating over $\mathbb{Z}[j]$ is equivalent to rounding both the real and imaginary part to the nearest integer. If $\mu \geq 3$, then $\mathbb{Z}[\omega]$ is *dense* in $\mathbb{C}$, meaning that any complex number can be arbitrarily approximated by a number in $\mathbb{Z}[\omega]$ without requiring the multiplication by a scaling factor. In practical applications, we are interested only in those elements of $\mathbb{Z}[\omega]$ which can be represented by bounded integer coefficients $\alpha_i \leq K$. As a drawback, such a representation requires to encrypt $R/2$ values for each complex sample. Since the number of bits of the encrypted representation is fixed due to security requirements, this means that the algebraic integer representation would require $R/4$ times the bits of an integer representation when translated into the encrypted domain.

Recently another representation for real values have been proposed. The main idea [FDH+10] is to represent each real number x with a triple $(\rho_x, \sigma_x, \tau_x)$, where $\rho_x \in \{0, 1\}$ is a flag indicating that x is not equal to 0, $\sigma_x = \text{sign}(x) \in \{-1, +1\}$ is the sign of the value and

$$\tau_x = \left\lceil -S \log_B \frac{|x|}{C} \right\rfloor.$$

Hence $x \approx \rho_x \cdot \sigma_x \cdot C \cdot B^{-\tau_x/S}$. This representation permits to have the same relative mean error for big and small values. The product between two encrypted numbers is relatively easy to compute, but addition requires a complex protocol involving a huge look up table.

Remarks

Given the above discussion, it is evident that fixed point representation without truncation and rescaling is the most suitable representation model if homomorphic encryption has to be used. For this reason in the following we will always assume that signals are represented as sequences whose values are represented by fixed point integers. Once the signal representation model has been fixed, it is necessary to choose its working parameters, namely the number of bits used to represent a signal sample and the quantization step used to pass from a real to an integer representation.

The actual value of a sample should always be recoverable from its finite field representation. For example, if we are working on $\mathbb{Z}_N$, the set of integers modulo N, for each sample x we should have $|x| \leq (N-1)/2$; otherwise, its magnitude will be lost due to the modulo N operations. Hence, the size of N imposes a trade-off between the accuracy of encrypted domain computations and the number of operations that can be performed without resorting to interactive protocols.

Chapter 3

Homomorphic Encryption

In this chapter we are going to introduce homomorphic encryption (HE). Asymmetric encryption schemes owning to this category permit to perform operations on encrypted data, without knowing the private (decryption) key. The operations that can be performed depend on the specific property of each encryption scheme. The most famous and used homomorphic cryptosystems allow to compute the sum between encrypted numbers by performing another operation. In the last years a new homomorphic protocol [Gen09], allowing the computation of both additions and products, has been proposed, arousing the interest of the scientific community, but its low performance impede its practical employment.

We start this chapter with an overview of some cryptographic schemes in Section 3.1 and then we analyse the El-Gamal and the Paillier cryptosystems respectively in Section 3.2 and Section 3.3. A list of basic (non-interactive and interactive) protocols, useful to construct more complex applications, is presented and accurately analysed in Section 3.4. We continue illustrating in Section 3.5 the implementation of the Discret Fourier Transform in the encrypted domain and a composite data representation that permits to reduce the size of an encrypted signal still allowing to process it. Finally in Section 3.6 fully (not yet efficient) homomorphic cryptosystems are discussed.

3.1 Basis of homomorphic encryption

Formally an encryption $\mathcal{E} : G_1 \rightarrow G_2$ and its decryption $\mathcal{D} : G_2 \rightarrow G_1$ are homomorphic if there are two homomorphisms $\phi_1 : G_1 \times G_1 \rightarrow G_1$ and $\phi_2 : G_2 \times G_2 \rightarrow G_2$ such that for any $a, b \in G_1$ we have:

$$\phi_1(a, b) = \mathcal{D}(\phi_2(\mathcal{E}(a), \mathcal{E}(b))). \tag{3.1}$$

The homomorphic properties of some cryptosystems depend on a the following property: the *malleability*. More in detail:

Definition 1 (Malleability)**.** *A cryptosystem is malleable if given an encryption of a plain message m, it is possible to generate another cyphertext which decrypts to $f(m)$, for a known function $f(\cdot)$, without necessarily having information about m.*

Even if malleability can be considered a security weakness, since it allows to modify cyphertexts by non authorized parties, in the s.p.e.d. context it is a very useful property. In fact homomorphic cryptosystems have several applications like secure voting systems [Ben88], private information retrieval [OS07], and searchable encryption [BDCOP04].

The first homomorphic encryptions have been proposed (named as *homomorphic privacies*) in [RAD78a]. These methods were based on the difficulty of factoring large integers and have been broken later (by [BY88]). One privacy homomorphism which can be used for both addition and multiplication has been proposed in [DF96] and is used for private data-analysis with rational numbers. The same author has also proposed a provably secure (in their proposal) multiplicative and additive homomorphic encryption in [DF02]. In the proposed methods the length of ciphertexts grows exponentially with operations on encrypted values. These systems have been broken by [Wag03] and [CKN06b].

Homomorphic encryption systems have also been analyzed in generic settings. It has been shown in [ALN87] that any non probabilistic encryption scheme which performs addition in the encrypted domain is insecure. In 1996 Boneh and Lipton [BL96] showed that any deterministic homomorphic encryption scheme can be broken in sub-exponential time and hence any such cryptosystem which works on a small set must be probabilistic.

Most popular homomorphic encryption systems are either multiplicative or additive (ϕ_2 is $*$ or $+$ respectively in Equation 3.1) and are based on two assumptions: the difficulty of solving discrete logarithm problem and different residue systems. The most famous example of multiplicative homomorphic encryption is the El-Gamal cryptosystem [ElG85], described in Section 3.2.

The other group of cryptosystems consists of encryption schemes whose security is based on the difficulty of finding quadratic and other residue classes in $\mathbb{Z}_N$, where N is a large composite number whose factorization is unknown. The first example of this group is the Goldwasser and Micali encryption scheme [GM84] which assumes the difficulty of deciding if a number a is a quadratic residue modulo a composite integer N (i.e. determine if b exists such that $a = b^2 \mod N$). This is a probabilistic encryption for which plaintext messages are single bits each of which is mapped in to an element of $\mathbb{Z}_N$. The encryption of XOR combination of bits is the product of encryptions. This scheme is known as the first probabilistic encryption scheme. The next method in this group is the Benaloh encryption scheme presented in [Ben88]. This system is very similar to the previous system, in which the power of 2, for the quadratic residuosity, is replaced by another number r and in this way the cardinality of plaintext set is increased. This method works only for small values of r since decryption must be done by exhaustive search or by Baby-step giant-step in $O(\sqrt{r})$ operations, as shown in [Ben88]. Another scheme is the Naccache-Stern cryptosystem [NS97] which is again very similar to the Benaloh-cryptosystem. Its security is based on the difficulty of distinguishing x^p in $\mathbb{Z}_N$, where p is a prime factor of $\phi(N)$, like the previous system, while the decryption is improved with respect to it. The Okamoto-Uchiyama cryptosystem ([OU98]) with a modulo number of the form $N = p^2q$ is the first one which has a large bandwidth and makes the decryption simpler than former solutions. This system has been later improved by Paillier [Pai99], as described in Section 3.3.

Some of the above homomorphic encryption systems, their homomorphisms ϕ_1 and ϕ_2, and corresponding literature references are given in Table 3.1.

3.2 El-Gamal cryptosystem

We now examine the El-Gamal encryption, the most popular multiplicative homomorphic cryptosystem. ElGamal encryption system is an asymmetric key encryption algorithm for public-key cryptography which is based on the DiffieHellman key exchange [DH76, Hel78]. ElGamal encryption is used in the free GNU Privacy Guard software, a recent versions of PGP, and other cryptosystems. Its security depends upon the difficulty to computing discrete

Encryption	ϕ_1	ϕ_2	Probabilistic	Reference
RSA without padding	multiplication	multiplication	No	[RSA78]
El-Gamal	multiplication	multiplication	Yes	[ElG85]
Goldwasser-Micali	XOR	multiplication	Yes	[GM84]
Benaloh	addition	multiplication	Yes	[Ben88]
Naccache-Stern	addition	multiplication	Yes	[NS97]
Okamoto-Uchiyama	addition	multiplication	Yes	[OU98]
Paillier	addition	multiplication	Yes	[Pai99]
Damgård-Jurik	addition	multiplication	Yes	[DJ01]

Table 3.1: Some homomorphic encryption systems, their homomorphisms ϕ_1 and ϕ_2, and corresponding literature references

logarithms in a cyclic group G. If the decisional DiffieHellman assumption (DDH) [DH76, Bon98, NP01] holds in G, then ElGamal achieves semantic security, providing indistinguishability under Chosen Plaintext Attack (IND-CPA), even if not under Chosen Ciphertext Attack (IND-CCA).

In El-Gamal cryptosystem a cyclic group G of order q and a generator g of G are given. The encryption and decryption are described below:

- Key generation: the key owner (receiver) chooses a secret element $x \in \mathbb{Z}_q$. The public key is composed by g and $h = g^x$, while the private (decryption) key is x.
- Encryption: the encryption of the message $m \in \mathbb{Z}_q$ is c_1, c_2 with:

  $$c_1 = g^y$$
  $$c_2 = mh^y$$

  where $y \in_R \mathbb{Z}_q$ is randomly chosen by the sender ($\in_R$ indicates that the value is randomly chosen among all the values of the set).
- Decryption: given the encrypted message c_1, c_2 the message m can be obtained as:

  $$m = c_2(c_1^x)^{-1}.$$

When the encryptions $c_{1,m_1} = g^{y_1}$, $c_{2,m_1} = m_1 h^{y_1}$, $c_{1,m_2} = g^{y_2}$, and $c_{2,m_2} = m_2 h^{y_2}$ are given, $c_{1,m_1} c_{1,m_2} = g^{y_1+y_2}$ and $c_{2,m_1} c_{2,m_2} = m_1 m_2 h^{y_1+y_2}$ is a valid encryption of the message $m_1 m_2$.

3.3 Paillier cryptosystem

With easy implementable encryption and decryption, Paillier cryptosystem is the most used additive homomorphic cryptosystem, it is also used in this thesis for the description of every homomorphic protocol.

In the Paillier cryptosystem the message space is $\mathbb{Z}_N$, while the ciphertext belongs to $\mathbb{Z}_{N^2}$ and requires $2T$ bits, where $T = \log_2(N)$. The flexibility of this system has been later improved in [DJ01] in which the plaintext and ciphertext sets belong to to $\mathbb{Z}_{N^s}$, and $\mathbb{Z}_{N^{s+1}}$, respectively, where $s > 1$. The encryption and decryption algorithms for the Paillier system are shown below:

- Key generation: the key owner computes $N = pq$ and $\lambda = \mathrm{lcm}(p-1, q-1)$ and selects a random integer $g \in \mathbb{Z}^*_{N^2}$ such that $N|\mathrm{ord}(g)$ (N divides the order of g). The public (encryption) key is (N, g) and the private (decryption) key is λ.

- Encryption: the encryption of the message $m \in \mathbb{Z}_N$ is c with

  $$c = g^m r^N \mod N^2,$$

  where $r \in_R \mathbb{Z}_N$.

- Decryption: given the encryption $c \in \mathbb{Z}_{N^2}$, the original message m can be obtained as:

  $$m = L(c^\lambda \mod N^2)\mu \mod N,$$

where $\mu = (L(g^\lambda \mod N^2))^{-1} \mod N$, and L is an integer function defined by $L(u) = \lfloor (u-1)/N \rfloor$.

In [Pai99] it has been shown that the security of this system is based on the difficulty of distinguishing elements which are Nth powers from those which are not, in $\mathbb{Z}_N$, when N is a composite number. This assumption is called *decisional composite residuosity assumption*, or DCRA for short. Regarding the homomorphic property of this encryption scheme, we can see that $\mathcal{E}(m_1 + m_2) = g^{m_1+m_2} r^N \mod N^2$, whereas $\mathcal{E}(m_1)\mathcal{E}(m_2) = g^{m_1+m_2}(r_1 r_2)^N \mod N^2$. Since r, r_1, r_2 are arbitrary values, $\mathcal{D}(\mathcal{E}(m_1)\mathcal{E}(m_2)) = m_1 + m_2$.

The efficiency of the Paillier system, together with its homomorphic properties, determined its use in several applications, such as private data retrieval on streaming data [OS07].

In the following, we will refer to a message encrypted by using an additive cryptosystem (generally Paillier) with the notation $[\![\cdot]\!]$. Moreover when we assert that $[\![x]\!] = [\![y]\!]$ we will mean that $\mathcal{D}(\mathcal{E}(x)) = \mathcal{D}(\mathcal{E}(y))$, where we avoid to use the decryption $\mathcal{D}(\cdot)$ and encryption $\mathcal{E}(\cdot)$ operators for simplicity.

3.4 Basic homomorphic protocols for STPC

In this section we present a list of protocols based on homomorphic encryption useful for several STPC applications. In these protocols two parties are involved: a client $\mathcal{C}$, owning the private key, and a server $\mathcal{S}$ interested to process data encrypted with $\mathcal{C}$'s public key, without knowing the private key. We start with some trivial protocols that can be easily computed by $\mathcal{S}$ and then we describe a list of protocols that require interaction with $\mathcal{C}$. If not explicitly specified, protocols are well-known and simple applications of homomorphic encryption, here optimized according to the authors experience.

The complexity of each protocol is analysed. The communication complexity is related to the number of rounds, i.e. the number of times the protocol requires the transmission of data and the total number of bits transmitted by the protocol. Considering that each protocol can be part of a more complex application, the starting input transmission from $\mathcal{C}$ to $\mathcal{S}$ and the final result transmission from $\mathcal{S}$ to $\mathcal{C}$ are omitted from the analysis. The computational complexity is expressed as the number of operations required. Considering that encryption (`enc`) and decryption (`dec`) operations have complexity comparable to an exponentiation (`exp`) and that multiplication (`mult`) has negligible complexity with respect to an `exp`, the computational complexity will be given as a function of the number of `exp` operations.

3.4.1 Product with a plain value

We suppose that $\mathcal{S}$ needs to compute the product between a value x owned by $\mathcal{C}$ and a value a owned by $\mathcal{S}$: $y = a \cdot x$. Obviously $\mathcal{S}$ has only the encrypted version $[\![x]\!]$ available and the result will be in encrypted form. If $a = 2$, $\mathcal{S}$ can compute $[\![y]\!] = [\![2x]\!] = [\![x+x]\!] = [\![x]\!][\![x]\!] = [\![x]\!]^2$. In general $\mathcal{S}$ can compute $[\![y]\!] = [\![ax]\!]$ as $[\![y]\!] = [\![x]\!]^a$ for any integer a.

The product does not require interaction, hence has null communication complexity. The computation complexity is 1 `exp`. Table 3.2 summarizes the complexities involved in the product.

Rounds	**Bandwidth**	**#** `exp`
0	0	1

Table 3.2: Product by plain value: complexities

3.4.2 Scalar and Matrix product

Let us consider the case where $\mathcal{S}$ has to compute the scalar product $y = \mathbf{a} \cdot \mathbf{x}^\top$ between a vector $\mathbf{x} = [x_1, \dots, x_n]$ owned by $\mathcal{C}$ and a vector $\mathbf{a} = [a_1, \dots, a_n]$ owned by $\mathcal{S}$ [EPK+07]. For the computation we need that $\mathcal{C}$ encrypts each element of the vector separately. Given the encryption $[\![\mathbf{x}]\!] = [[\![x_1]\!], \dots, [\![x_n]\!]]$ of the vector, $\mathcal{S}$ can compute $[\![y]\!] = [\![\mathbf{a} \cdot \mathbf{x}^T]\!] = [\![\sum_{i=1}^n a_i x_i]\!] = \prod_{i=1}^n [\![a_i x_i]\!] = \prod_{i=1}^n [\![x_i]\!]^{a_i}$. The scalar product has null communication complexity, while n `exp` and $n-1$ `mult` are required. Table 3.3 summarizes the complexities involved in the scalar product.

Rounds	**Bandwidth**	**#** `exp`
0	0	n

Table 3.3: Scalar product with plain vector: complexities

Given the implementation of the scalar product, the protocol that computes a matrix product is trivial. Given the (l, m) matrix

$$[\![\mathbf{x}]\!] = \begin{bmatrix} [\![x_{1,1}]\!] & \cdots & [\![x_{1,m}]\!] \\ \vdots & \ddots & \vdots \\ [\![x_{l,1}]\!] & \cdots & [\![x_{l,m}]\!] \end{bmatrix}$$

and the (m,n) matrix

$$\mathbf{a} = \begin{bmatrix} a_{1,1} & \cdots & a_{1,n} \\ \vdots & \ddots & \vdots \\ a_{m,1} & \cdots & a_{m,n} \end{bmatrix},$$

$\mathcal{S}$ can compute the (l,n) matrix $[\![\mathbf{y}]\!] = [\![\mathbf{xa}]\!]$ where the generic i,j element is $[\![y_{i,j}]\!] = \prod_{k=1}^{m} [\![x_{i,k}]\!]^{a_{k,j}}$. The complexity of the matrix product is equal to the complexity of $l \times n$ scalar products between arrays having m elements.

3.4.3 Linear Filtering

Implementing a FIR (Finite Impulse Response) filter in the encrypted domain is easy [EPK+07]. We suppose that $\mathcal{S}$ has to filter the sampled signal $[\![\mathbf{x}]\!] = \{[\![x_0]\!], \ldots, [\![x_n]\!]\}$ by applying a filter having integer coefficients $c_0, \ldots, c_k$ ($k \ll n$). If the coefficients are real they have to be quantized. An encrypted filtered sample can be computed as

$$[\![y_i]\!] = [\![\sum_{j=0}^{k} x_{i-j} c_j]\!] = \prod_{j=0}^{k} [\![x_{i-j}]\!]^{c_j}. \tag{3.2}$$

Obviously, as in plain domain filtering, we have to pay attention to the indices, by limiting the computation to $[\![y_k]\!], \ldots, [\![y_n]\!]$ or by choosing a value for $[\![x_{i-j}]\!]$ ($[\![x_{i-j}]\!] = [\![x_0]\!]$ or $x_{i-j} = 0$) when $i - j < 0$. Considering that a filter having even real magnitude response, has symmetric coefficients, i.e. $c_i = c_{k-i}$, and that it is possible that there are more integer coefficients having the same value, it is convenient to group the x_i values that have to be multiplied by the same coefficient and add them together. If we consider the set of different coefficients c_a of the filter and the sets of indices $\{j_{i,a}\}$ (the indices of all the samples that have to be multiplied by c_a to compute y_i) we can rewrite (3.2) as:

$$[\![y_i]\!] = [\![\sum_{\{a\}} c_a \sum_{\{j_{i,a}\}} x_{j_{i,a}}]\!] = \prod_{\{a\}} \Big(\prod_{\{j_{i,a}\}} [\![x_{j_{i,a}}]\!] \Big)^{c_a}. \tag{3.3}$$

By considering that the protocol does not require interaction and we have to compute as many exponentiations as the number of different coefficients, we can summarize the complexity as in table 3.4.

Rounds	Bandwidth	# `exp`
0	0	$\#[c_a]$

Table 3.4: FIR filtering: complexities ($\#[c_a]$ denotes the number of different coefficients in the filter)

IIR (Infinite Impulse Response) filters do not have an efficient implementation in the encrypted domain. Their principal characteristic is the reduced number of coefficients required, but any filtered sample is computed as a function of the original samples and the previously filtered samples:

$$y_i = \sum_{j=0}^{k_1} x_{i-j}a_j + \sum_{j=1}^{k_2} y_{i-j}b_j. \tag{3.4}$$

Observing (3.4) we can see that the implementation in the encrypted domain is easy, but we have problems with the output representation. The quantisation introduced in a FIR filter produces a filtered signal where all the samples are amplified by the same factor. In the IIR filters the amplification grows for each sample because they are computed by multiplying the previous filtered sample with a quantised coefficient, hence interaction with the private key owner is cyclically required to remove the amplification factor from the filtered samples (i.e. by truncating the output of the filter) and go on with the computation.

3.4.4 Packing

Whenever during the computation (or at the end of the protocol) $\mathcal{S}$ has to send back the intermediate values (or the final result) to $\mathcal{C}$ and the values can be represented by an array of elements, as first proposed in [TPKCL07], instead of transmitting the encryption of each element $\mathcal{S}$ can pack more values together to optimize the bandwidth required for the transmission. To pack more values in a single cyphertext it is necessary that the following conditions are satisfied:

- each value $[\![y_i]\!]$ is bounded, i.e. $y_i \in [y_{i,min}, y_{i,max})$;
- the values $y_{i,min}, y_{i,max}, \; \forall i$ are known to both $\mathcal{C}$ and $\mathcal{S}$.

For simplicity we suppose that each $y_i \in [0, 2^\ell)$. Extension to different ℓ_i is trivial. We can pack up to k values together (where $k = \lfloor N/2^\ell \rfloor$ and N is the length of the ring) as in Figure 3.1 by computing

$$\llbracket \widehat{y}_h \rrbracket = \llbracket \sum_{i=0}^{k-1} y_{hk+i} \cdot 2^{i\ell} \rrbracket = \prod_{i=0}^{k-1} \llbracket y_{hk+i} \rrbracket^{2^{i\ell}}, \tag{3.5}$$

where $h \geq 0$ is the index of the pack, when more than one pack are required.

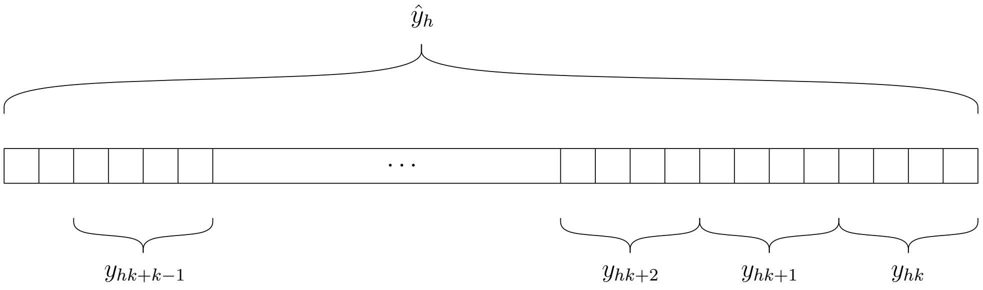


Figure 3.1: Packing more values.

If the interval is different from $[0, 2^\ell)$ we can translate each value $y_i \in [y_{i,min}, y_{i,max})$ in a value $\overline{y}_i = y_i - y_{i,min} \in [0, y_{i,max} - y_{i,min}) \subseteq [0, 2^{\ell_i})$. Considering the values in the interval $[0, 2^{\ell_i})$ instead of $[0, y_{i,max} - y_{i,min})$ is sometime less efficient from a bandwidth point of view, but permits faster unpacking on $\mathcal{C}$'s side, after decryption. More generally, when numbers can assume negative values, i.e. $-2^{\ell-1} < y < 2^{\ell-1}$, it is preferable to change their representation by mapping them into the interval $(0, 2^\ell)$:

$$\llbracket \widehat{y}_h \rrbracket = \llbracket \sum_{i=0}^{k-1} (y_{hk+i} + 2^{\ell-1}) \cdot 2^{i\ell} \rrbracket = \prod_{i=0}^{k-1} (\llbracket y_{hk+i} \rrbracket \llbracket 2^{\ell-1} \rrbracket)^{2^{i\ell}}. \tag{3.6}$$

If the original values have to be provided to $\mathcal{C}$, he unpacks the values in $\widehat{y}_h$ and removes the translation, if present. Otherwise if the values contained in a pack need to be unpacked and used in further computation, an interactive protocol is necessary. Considering a single pack, a simple protocol is based on obfuscation: $\mathcal{S}$ chooses k random values r_i of size $\ell - 1$, computes $r = \sum_{i=0}^{k-1} r_i 2^i$ and blinds the pack with r: $\llbracket \hat{y}_{h,bl} \rrbracket = \llbracket \hat{y}_h + r \rrbracket = \llbracket \hat{y}_h \rrbracket \llbracket r \rrbracket$. Considering that for security we need that $r_i \gg y_i$ a new constraint on ℓ and y_i must be satisfied, namely $-2^{\ell-\tau-1} < y_i < 2^{\ell-\tau-1}$, where τ is a security blinding parameter and the 1 subtracted in the outers' exponents is introduced to avoid that the sum carry is added to y_{i+1}. At this point $\mathcal{S}$ sends the encrypted pack to $\mathcal{C}$ that decrypts it, unpacks the values, encrypts them again and sends them to $\mathcal{S}$, that will finally remove the obfuscation by using the additive homomorphic property of the encryption scheme.

Packing has null communication complexity. To evaluate the computational complexity we have to consider a case in which we must pack n values in more packs. Each packing requires $k-1$ `exp` (the last value of each pack do not need the exponentiation). The number of packs is equal to $p = \lfloor n/k \rfloor$. The total computation complexity is $n-p$ `exp`. Table 3.5 summarizes the complexities involved in the packing.

Rounds	**Bandwidth**	# `exp`
0	0	$n-p$

Table 3.5: Packing: complexities

3.4.5 Non linear function evaluation

Whenever $\mathcal{S}$ needs to evaluate a non linear function $\phi(\cdot)$ on variables $x_1, \ldots, x_n$ available to $\mathcal{S}$ only in encrypted form $[\![x_1]\!], \ldots, [\![x_n]\!]$, it has to perform an interactive protocol together with $\mathcal{C}$. As already said in Section 2.5 and demonstrated in [BAS+09], additive obfuscation is more secure then multiplicative obfuscation, hence here we consider only additive obfuscation. To evaluate a non linear function, $\mathcal{S}$ has to obfuscate the encrypted values $[\![x_i]\!]$, where x_i can be represented by using ℓ bits, by using random values that have to be chosen in $\{0,1\}^{\ell+\kappa}$ (κ is a security obfuscation parameter, generally $\kappa = 80$).

At this point $\mathcal{S}$ transmits the blinded values to $\mathcal{C}$ (to optimize the bandwidth during the transmission, packing can be used) that decrypts the values and performs some computation. Finally $\mathcal{C}$ sends the encrypted results to $\mathcal{S}$ that must recover $\phi(x_1, \ldots, x_n)$ by using the encrypted result, the encryption of the x_i values and the random values he used for the obfuscation. This general scheme will be specifically analysed in the following interactive schemes.

3.4.6 Product between two encrypted numbers

Computing a product between two encrypted numbers is not possible without interaction. Let assume that $\mathcal{S}$ owns $\llbracket x \rrbracket$ and $\llbracket y \rrbracket$, encrypted with the public key of $\mathcal{C}$, and is interested in computing $\llbracket z \rrbracket = \llbracket xy \rrbracket$ without revealing x, y and z to $\mathcal{C}$. $\mathcal{S}$ can use eMul protocol, shown in Protocol 2 that returns $\mathsf{eMul}(\llbracket x \rrbracket, \llbracket y \rrbracket) = \llbracket xy \rrbracket$. In Protocol 2 and in the following protocols, packing is overlooked for simplicity.

eMul FUNCTION

inputs of $\mathcal{C}$: Nothing
inputs of $\mathcal{S}$: $\llbracket x \rrbracket, \llbracket y \rrbracket$
output for $\mathcal{C}$: Nothing
output for $\mathcal{S}$: $\llbracket z \rrbracket = \mathsf{eMul}(\llbracket x \rrbracket, \llbracket y \rrbracket) = \llbracket xy \rrbracket$

client $\mathcal{C}$ server $\mathcal{S}$

server: chooses $r_x, r_y \in_R \{0,1\}^{\ell+\kappa}$;
$\llbracket x_r \rrbracket = \llbracket x + r_x \rrbracket = \llbracket x \rrbracket \llbracket r_x \rrbracket$;
$\llbracket y_r \rrbracket = \llbracket y + r_y \rrbracket = \llbracket y \rrbracket \llbracket r_y \rrbracket$;

$$\xleftarrow{\quad \llbracket x_r \rrbracket, \llbracket y_r \rrbracket \quad}$$

client: decrypts $\llbracket x_r \rrbracket, \llbracket y_r \rrbracket$;
$w = x_r y_r = (x + r_x)(y + r_y) =$
$= xy + x r_y + y r_x + r_x r_y$;
encrypts w;

$$\xrightarrow{\quad \llbracket w \rrbracket \quad}$$

server: $\llbracket z \rrbracket = \llbracket w \rrbracket \llbracket x \rrbracket^{-r_y} \llbracket y \rrbracket^{-r_x} \llbracket -r_x r_y \rrbracket =$
$= \llbracket w - x r_y - y r_x - r_x r_y \rrbracket = \llbracket xy \rrbracket$.

Protocol 2: Interactive protocol that computes the product between two encrypted values.

Computing eMul requires 2 rounds (one from $\mathcal{S}$ to send the obfuscated ciphertexts and one from $\mathcal{C}$ to send back the result) and a bandwidth of $6T$ bits (3 ciphertexts, each $2T$ bits long, are sent). If x_r and y_r can be packed together, only $4T$ bits have to be transmitted. The computational complexity is equal to: 2 `enc` to obtain the encryption of $\llbracket r_x \rrbracket$ and $\llbracket r_x \rrbracket$ necessary in the obfuscation phase; 2 `mult` needed to obfuscate $\llbracket x \rrbracket$ and $\llbracket y \rrbracket$; 1 `exp` and 1 `mult` to pack the values; 1 `dec` to obtain x_r and y_r in plain; 1 `enc` to encrypt the result of the plain product; 2 `exp` needed to compute $\llbracket x \rrbracket^{-r_y}$ and $\llbracket y \rrbracket^{-r_x}$; 1 `enc` to compute the value $[\![-r_x r_y]\!]$ necessary to remove the obfuscation and finally 3 `mult` needed to compute the additions to $[\![w]\!]$, obtaining a total number of 8 `exp` operations. Table 3.6 summarizes the complexities involved in the eMul protocol.

Rounds	**Bandwidth**	**#** `exp`
2	$4T$	8

Table 3.6: eMul: complexities

3.4.7 Multiplexer

We now consider the Multiplexer function that implements the following condition in the encrypted domain:

```
if $b = 1$ then
    $z = x$
else
    $z = y$
end
```

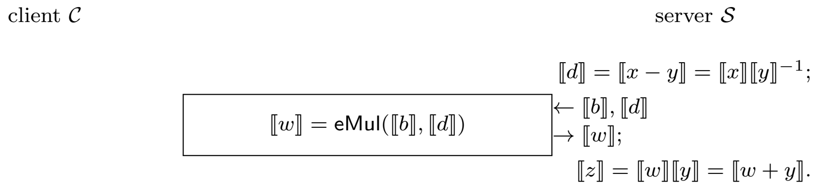
where b is a boolean input. The function can be expressed as $z = b(x - y) + y$ and can be implemented in the encrypted domain as described in Protocol 3. Considering that b is a single bit, we can perfectly blind it by a xor operation with a random bit r_b by computing $[\![b \oplus r_b]\!] = [\![b]\!]$ when $r_b = 0$ and by computing $[\![b \oplus r_b]\!] = [\![1 - b]\!] = [\![1]\!][\![b]\!]^{-1}$ when $r_b = 1$, and modifying the eMul function, so that once $\mathcal{C}$ returns the result, $\mathcal{S}$ can remove the obfuscation. To evaluate the obfuscation introduced we can consider that $[\![(b \oplus r_b)(d + r)]\!] = [\![b(d + r)]\!] = [\![bd + br]\!]$ when $r_b = 0$ (where $r \in_R \{0, 1\}^{\ell+\kappa}$ is the value used to obfuscate d) and $[\![(b \oplus r_b)(d + r)]\!] = [\![(1 - b)(d + r)]\!] = [\![d + r - bd - br]\!]$ when $r_b = 1$. Hence $\mathcal{S}$ has to compute

$$[\![w]\!] = \begin{cases} [\![(b \oplus r_b)(d + r)]\!][\![b]\!]^{-r} & \textbf{if } r_b = 0 \\ [\![(b \oplus r_b)(d + r)]\!]^{-1}[\![b]\!]^{-r}[\![d + r]\!] & \textbf{if } r_b = 1 \end{cases} . \tag{3.7}$$

The protocol requires two rounds and by packing together $[\![b \oplus r_b]\!]$ and $[\![d + r]\!]$ in the first transmission, only two cyphertexts are transmitted. 1 `exp`

Multiplexer PROTOCOL

inputs of $\mathcal{C}$: Nothing
inputs of $\mathcal{S}$: $[\![b]\!], [\![x]\!], [\![y]\!]$
output for $\mathcal{C}$: Nothing
output for $\mathcal{S}$: $[\![z]\!] = [\![b(x-y)+y]\!]$


client $\mathcal{C}$
server $\mathcal{S}$
$[\![d]\!] = [\![x-y]\!] = [\![x]\!][\![y]\!]^{-1}$;
$\leftarrow [\![b]\!], [\![d]\!]$
$[\![w]\!] = \mathsf{eMul}([\![b]\!], [\![d]\!])$
$\rightarrow [\![w]\!]$;
$[\![z]\!] = [\![w]\!][\![y]\!] = [\![w+y]\!]$.

Protocol 3: Interactive protocol that select a value depending on a boolean.

is needed to compute $[\![d]\!]$. To obtain $[\![b \oplus r_b]\!]$ we need $1/2$ `enc` and $1/2$ `exp` on average, and 1 `exp` is necessary to pack the values. $\mathcal{C}$ performs 1 `dec` and 1 `enc`. At the end of the eMul function $1+1/2$ `exp` are required on the average to remove the obfuscation. The protocol complexity is summarized in Table 3.7.

Rounds	Bandwidth	# `exp`
2	$4T$	$6+1/2$

Table 3.7: Multiplexer: average complexities

3.4.8 Scalar and Matrix products between encrypted matrices

Let us consider the case where $\mathcal{S}$ has to compute the scalar product $y = \mathbf{a} \cdot \mathbf{b}^\top$ between the encrypted vectors $[\![\mathbf{a}]\!] = [[\![a_1]\!], \ldots, [\![a_n]\!]]$ and $[\![\mathbf{b}]\!] = [[\![b_1]\!], \ldots, [\![b_n]\!]]$. The scalar product needs interaction and can be obtained by extending the eMul protocol, as shown in Protocol 4, where packing is not used for simplicity.

In eScalarProd, $\mathcal{S}$ obfuscates each element of the vector and sends them to $\mathcal{C}$, that computes the scalar product between the obfuscated arrays and sends back the encrypted result. Finally $\mathcal{S}$ removes the total obfuscation ob_{tot} that eScalarProd FUNCTION

inputs of $\mathcal{C}$: Nothing
inputs of $\mathcal{S}$: $[\![\mathbf{a}]\!], [\![\mathbf{b}^\top]\!]$
output for $\mathcal{C}$: Nothing
output for $\mathcal{S}$: $[\![y]\!] = \mathsf{eScalarProd}([\![\mathbf{a}]\!], [\![\mathbf{b}^\top]\!]) = [\![\mathbf{a} \cdot \mathbf{b}^\top]\!]$

client $\mathcal{C}$		server $\mathcal{S}$
		$\forall i = 1 \ldots n$: chooses $r_{a,i}, r_{b,i} \in_R \{0,1\}^{\ell+\kappa}$; $[\![a_i^r]\!] = [\![a_i + r_{a,i}]\!] = [\![a_i]\!][\![r_{a_i}]\!]$; $[\![b_i^r]\!] = [\![b_i + r_{b,i}]\!] = [\![b_i]\!][\![r_{b_i}]\!]$;
	$\xleftarrow{[\![a_i^r]\!], [\![b_i^r]\!]\ \forall i}$	
$\forall i$ decrypts $[\![a_i^r]\!], [\![b_i^r]\!]$; $w = \mathbf{a^r} \cdot \mathbf{b^r} = \sum_{i=1}^n (a_i^r b_i^r)$; encrypts w;		$[\![-ob_{tot}]\!] =$ $= [\![-\sum_{i=1}^n (a_i r_{b,i}) - \sum_{i=1}^n (b_i r_{a,i}) - \sum_{i=1}^n (r_{a,i} r_{b,i})]\!] =$ $= \prod_{i=1}^n [\![a_i]\!]^{-r_{b,i}} \cdot \prod_{i=1}^n [\![b_i]\!]^{-r_{a,i}} \cdot [\![-\sum_{i=1}^n (r_{a,i} r_{b,i})]\!]$;
	$\xrightarrow{[\![w]\!]}$	
		$[\![y]\!] = [\![w]\!][\![-ob_{tot}]\!]$.

Protocol 4: Interactive protocol that computes the scalar product between two encrypted arrays.

can be obtained by considering that

$$\begin{aligned}
\mathbf{a^r} \cdot \mathbf{b^r}^\top &= \sum_{i=1}^n (a_i^r b_i^r) \\
&= \sum_{i=1}^n [(a_i + r_{a,i})(b_i + r_{b,i}) \\
&= \sum_{i=1}^n (a_i b_i + a_i r_{b,i} + b_i r_{a,i} + r_{a,i} r_{b,i}) \\
&= \sum_{i=1}^n (a_i b_i) + \sum_{i=1}^n (a_i r_{b,i}) + \sum_{i=1}^n (b_i r_{a,i}) + \sum_{i=1}^n (r_{a,i} r_{b,i}) \\
&= \mathbf{a} \cdot \mathbf{b}^\top + ob_{tot}.
\end{aligned} \tag{3.8}$$

Analysing the protocol complexity we observe that 2 rounds are required. If we suppose that each array element can be represented by using ℓ bits we can pack the elements into $p = \lceil 2n/k \rceil$ packs that have to be transmitted in the first round (where $k = \lfloor T/(\ell+\kappa) \rfloor$ is the maximum number of elements in each pack). The protocol needs $2n$ `enc` for the encryption of the obfuscation parameters and $2n$ `mult` for blinding on $\mathcal{S}$ side; $2n - p$ `exp` and $2n - p$ `mult` for packing; p `dec` and 1 `enc` on $\mathcal{C}$ side; $2n$ `exp`, $2n$ `mult` and 1 `enc` to compute $[\![-ob_{tot}]\!]$ and finally 1 `mult` to remove it. Note that if packing is not used the computation complexity is similar because the $2n - p$ `exp` for packing and p `dec` are replaced by $2n$ `dec`. The complexities involved in the scalar product are summarized in Table 3.8.

Rounds	**Bandwidth**	**#** `exp`
2	$(p+1)2T$	$6n+2$

Table 3.8: eScalarProd: Complexities

Given the eScalarProd implementation, the protocol that computes a product between encrypted matrixes is trivial. Given the (l, m) matrix

$$[\![\mathbf{a}]\!] = \begin{bmatrix} [\![a_{1,1}]\!] & \cdots & [\![a_{1,m}]\!] \\ \vdots & \ddots & \vdots \\ [\![a_{l,1}]\!] & \cdots & [\![a_{l,m}]\!] \end{bmatrix}$$

and the (m, n) matrix

$$[\![\mathbf{b}]\!] = \begin{bmatrix} [\![b_{1,1}]\!] & \cdots & [\![b_{1,n}]\!] \\ \vdots & \ddots & \vdots \\ [\![b_{m,1}]\!] & \cdots & [\![b_{m,n}]\!] \end{bmatrix},$$

$\mathcal{S}$ can compute the (l, n) matrix $[\![\mathbf{y}]\!]$ where the generic i, j element is $[\![y_{i,j}]\!] =$ eScalarProd$([\![\mathbf{a_i}]\!], [\![\mathbf{b_j}]\!])$, where $[\![\mathbf{a_i}]\!]$ is the row vector $[[\![a_{i,1}]\!], \ldots, [\![a_{i,m}]\!]]$ and $[\![\mathbf{b_j}]\!]$ is the column vector $[[\![b_{1,j}]\!], \ldots, [\![b_{m,j}]\!]]$.

The complexity of the matrix product can be related to the complexity of $l \times n$ scalar products between arrays having m elements. It is possible to run the eScalarProd algorithms in parallel, hence the number of rounds is the same. The protocol starts with $\mathcal{S}$ that obfuscates $lm + mn$ values ($lm + mn$ `exp`) and packs them into $p = \lfloor (lm + mn)/k \rfloor$ packs ($lm + mn - p$ `exp`). During the first round p packs are transmitted. While $\mathcal{C}$ decrypts p values (p `exp`), computes ln scalar products in the plain domain and encrypts the results (ln `exp`), $\mathcal{S}$ computes the total obfuscation matrix $[\![\mathbf{ob_{tot}}]\!]$ having l rows and n columns ($(2m+1)ln$ `exp`). During the second round the $[\![\mathbf{w}]\!]$ matrix is transmitted (ln values). Finally $\mathcal{S}$ removes the obfuscation $[\![\mathbf{ob_{tot}}]\!]$ from $[\![\mathbf{w}]\!]$ to obtain $[\![\mathbf{y}]\!]$ (0 `exp`). The complexity are summarized in Table 3.9.

Rounds	**Bandwidth**	# `exp`
2	$(p+ln)2T$	$2lmn+2lm+2mn+2ln$

Table 3.9: Matrix product between encrypted matrixes: complexities

3.4.9 Square of an encrypted number

The protocol to compute the square value of an encrypted number is similar to the eMul protocol. Let we assume that $\mathcal{S}$ owns $[\![x]\!]$, encrypted with the public key of $\mathcal{C}$, and is interested in computing $[\![z]\!] = [\![x^2]\!]$ without revealing x and z to $\mathcal{C}$. This task can be accomplishied by using the eSquare function, shown in Protocol 5 that returns eSquare$([\![x]\!]) = [\![x^2]\!]$.

Computing eSquare requires 2 rounds (one in which $\mathcal{S}$ sends the obfuscated ciphertext and one in which $\mathcal{C}$ sends back the result) and a bandwidth of $4T$ bits (2 ciphertexts are sent). The computational complexity is equal to: 1 `enc` to obtain the encryption of $[\![r_x]\!]$ necessary in the obfuscation phase; 1 `mult` needed to obfuscate $[\![x]\!]$; 1 `dec` to obtain x_r in plain; 1 `enc` to encrypt the result of the plain product; 1 `exp` needed to compute $[\![x]\!]^{-2r_x}$; 1 `enc` to compute the value $[\![-r_x^2]\!]$ necessary to remove the obfuscation and finally 2 `mult` needed to compute the additions to $[\![w]\!]$, obtaining a total number of 5 `exp` operations. Table 3.10 summaries the complexities involved in the eSquare protocol.

Rounds	**Bandwidth**	# `exp`
2	$4T$	5

Table 3.10: eSquare: complexities

eSquare FUNCTION

inputs of $\mathcal{C}$: Nothing
inputs of $\mathcal{S}$: $\llbracket x \rrbracket$
output for $\mathcal{C}$: Nothing
output for $\mathcal{S}$: $\llbracket z \rrbracket = \mathsf{eSquare}(\llbracket x \rrbracket) = \llbracket x^2 \rrbracket$

client $\mathcal{C}$ server $\mathcal{S}$

chooses $r_x \in_R \{0,1\}^{\ell+\kappa}$;
$\llbracket x_r \rrbracket = \llbracket x + r_x \rrbracket = \llbracket x \rrbracket \llbracket r_x \rrbracket$;

$$\xleftarrow{\llbracket x_r \rrbracket}$$

decrypts $\llbracket x_r \rrbracket$;
$w = x_r^2 = (x + r_x)^2 =$
$= x^2 + 2xr_x + r_x^2$;
encrypts w;

$$\xrightarrow{\llbracket w \rrbracket}$$

$\llbracket z \rrbracket = \llbracket w \rrbracket \llbracket x \rrbracket^{-2r_x} \llbracket -r_x^2 \rrbracket =$
$= \llbracket w - 2xr_x - r_x^2 \rrbracket = \llbracket x^2 \rrbracket$.

Protocol 5: Interactive protocol that computes the square of an encrypted value.

3.4.10 Energy of an encrypted signal

Given the encrypted samples $\llbracket s_i \rrbracket$ of a signal, with $i = 1, \ldots, n$, $\mathcal{S}$ can obtain the encryption of the energy $\llbracket E_s \rrbracket$ by observing that $E_s = \mathbf{s}\mathbf{s}^\top$ and hence using the eScalProd algorithm, which can be optimized, resulting in the eEnergy protocol described in Protocol 6.

Similarly to the eScalarProd protocol, $\mathcal{C}$ computes the energy of the obfuscated signal, while $\mathcal{S}$ computes the obfuscation that it has to remove from the value sent by $\mathcal{C}$. The protocol requires two rounds. During the first round, if packing is used, $p = \lceil n/k \rceil$ packs are transmitted, where $k = \lfloor N/(\ell + \kappa) \rfloor$ is the maximum number of elements in each pack. During the second round only a value is transmitted. Analysing the computational complexity, we can observe that $\mathcal{S}$ has to perform n obfuscations (n `mult`) and $n - p$ `exp` plus $n - p$ `mult` to pack the obfuscated values. Computing the total obfuscation requires 1 `enc`, n `exp` and n `mult`. Finally to remove the obfuscation, $\mathcal{S}$ has to compute 1 `mult`. During the protocol $\mathcal{C}$ has to compute p `dec` and 1 `enc`.

eEnergy PROTOCOL

inputs of $\mathcal{C}$: Nothing
inputs of $\mathcal{S}$: $\llbracket \mathbf{s} \rrbracket = [\llbracket s_1 \rrbracket, \ldots, \llbracket s_n \rrbracket]$
output for $\mathcal{C}$: Nothing
output for $\mathcal{S}$: $\llbracket E_s \rrbracket = \mathsf{eEnergy}(\llbracket \mathbf{s} \rrbracket) = \mathsf{eEnergy}([\llbracket s_1 \rrbracket, \ldots, \llbracket s_n \rrbracket])$

client $\mathcal{C}$		server $\mathcal{S}$
		$\forall i = 1 \ldots n:$ chooses $r_i \in_R \{0,1\}^{\ell+\kappa}$, $\llbracket s_{i,r} \rrbracket = \llbracket s_i + r_i \rrbracket = \llbracket s_i \rrbracket \llbracket r_i \rrbracket$;
	$\xleftarrow{\llbracket s_{i,r} \rrbracket \ \forall i}$	
$\forall i$ decrypts $\llbracket s_{i,r} \rrbracket$; $w = \sum_{i=1}^{n} s_{i,r}^2 = \sum_{i=1}^{n} (s_i + r_i)^2 =$ $= \sum_{i=1}^{n} s_i^2 + \sum_{i=1}^{n} r_i^2 + \sum_{i=1}^{n} 2 s_i r_i$; encrypts w;		$\llbracket -ob_{tot} \rrbracket =$ $= \llbracket -\sum_{i=1}^{n} r_i^2 - \sum_{i=1}^{n} 2 s_i r_i \rrbracket =$ $= \llbracket -\sum_{i=1}^{n} r_i^2 \rrbracket \prod_{i=1}^{n} \llbracket s_i \rrbracket^{-2 r_i}$;
	$\xrightarrow{\llbracket w \rrbracket}$	
		$\llbracket E_s \rrbracket = \llbracket w \rrbracket \llbracket -ob_{tot} \rrbracket$.

Protocol 6: Interactive protocol that computes the energy of a signal.

The complexities of the eEnergy protocol are shown in Table 3.11.

Rounds	Bandwidth	# exp
2	$(p+1)2T$	$2n+2$

Table 3.11: eEnergy: complexities

3.4.11 Binary representation of an encrypted value

The protocol toEncBit [ST06a], illustrated in Protocol 7, allows to obtain the encryption $\llbracket x_{\ell-1} \rrbracket$, …, $\llbracket x_0 \rrbracket$ of the bits of a number x given its encryption $\llbracket x \rrbracket$, where ℓ is the number of bits necessary for its representation ($0 \leq x < 2^{\ell}$).

Suppose that $\mathcal{S}$ owns $\llbracket x \rrbracket$ encrypted with the public key of $\mathcal{C}$, he obfuscates the encrypted value by adding a random number and sends the obfuscated value to $\mathcal{C}$ that replies sending back the encryption of the bits. An addition protocol, operating on the bits and having as input the encrypted binary representation of the obfuscated value and the bits of the value used for blinding,

toEncBit PROTOCOL

inputs of $\mathcal{C}$: Nothing
inputs of $\mathcal{S}$: $\llbracket x \rrbracket$
output for $\mathcal{C}$: Nothing
output for $\mathcal{S}$: $\llbracket x_{\ell-1} \rrbracket, \dots, \llbracket x_0 \rrbracket$

client $\mathcal{C}$ | server $\mathcal{S}$

chooses $r \in_R \{0,1\}^{\ell+\kappa}$;
$\llbracket z \rrbracket = \llbracket x \rrbracket \llbracket -r \rrbracket = \llbracket x - r \rrbracket$;

$$\xleftarrow{\quad \llbracket z \rrbracket \quad}$$

decrypts $\llbracket z \rrbracket$;
encrypts $\llbracket z_i \rrbracket$ $\forall i = 0, \dots, \ell-1$;

$$\xrightarrow{\quad \llbracket z_i \rrbracket \ \forall i \quad}$$

$\llbracket c_0 \rrbracket = \llbracket z_0 \rrbracket^{r_0} = \llbracket z_0 r_0 \rrbracket$;
$\llbracket x_0 \rrbracket = \llbracket z_0 \rrbracket \llbracket r_0 \rrbracket \llbracket c_0 \rrbracket^{-2} = \llbracket z_0 + r_0 - 2c_0 \rrbracket$;
for $i = 1, \dots, \ell-1$

$$z_i \rightarrow \boxed{\llbracket a_i \rrbracket = \mathsf{eMul}(z_i, \llbracket c_{i-1} \rrbracket)} \begin{array}{l} \leftarrow \llbracket c_{i-1} \rrbracket \\ \rightarrow \llbracket a_i \rrbracket; \end{array}$$

$\llbracket c_i \rrbracket = \llbracket z_i \rrbracket^{r_i} \llbracket c_{i-1} \rrbracket^{r_i} a_i^{1-2r_i} =$
$= \llbracket r_i z_i + r_i c_{i-1} + z_i c_{i-1}(1 - 2r_i) \rrbracket$;
$\llbracket x_i \rrbracket = \llbracket z_i \rrbracket \llbracket r_i \rrbracket \llbracket c_{i-1} \rrbracket \llbracket c_i \rrbracket^{-2} =$
$= \llbracket z_i + r_i + c_i - 2c_{i-1} \rrbracket$;
endfor.

Protocol 7: Interactive protocol that returns the encrypted bynary representation of a value.

is used to remove the obfuscation. For each i the encryption of x_i and the carry c_i necessary in the successive iteration are computed from $\llbracket z_i \rrbracket$ (the i-th bit of the obfuscated value), r_i (the i-th bit ob the obfuscation value) and c_i (the i-th carry bit).

Computing toEncBit requires 2ℓ rounds. The first 2 rounds are necessary to obtain the encrypted binary representation of z and $(\ell + 1)2T$ bits are transmitted. The other rounds are necessary to compute the eMul protocols. Note that z_i are already available on $\mathcal{C}$'s side, hence it is not necessary to transmit them again and the total communication bandwidth is $2(\ell - 1)2T$ bits. The total number of transmitted bits is $(\ell + 1 + 2(\ell - 1))T = (3\ell - 1)2T$ bits. The protocol requires $(\ell + 1)$ `enc` on $\mathcal{S}$'s side to obtain the encryptions $[\![r]\!], [\![r_0]\!], \ldots, [\![r_{\ell-1}]\!]$, 1 `dec` and ℓ `enc` on $\mathcal{C}$'s side to obtain the binary representation of z. During the bitwise addition protocol that removes the obfuscation, for each eMul function $\mathcal{C}$ has to perform 1 `dec` and 1 `enc`, while $\mathcal{S}$ has to compute 1 `enc` for the encryption of an obfuscation value $r_{c_{i-1}}$ and 1 `exp` to remove the obfuscation $r_{c_{i-1}} z_i$ from the result. Moreover $\mathcal{S}$ has to compute several exponentiations having r_i as exponent. Considering that $\mathcal{S}$ knows r_i, we have

$$[\![a]\!]^{r_i} = [\![a\ r_i]\!] = \begin{cases} [\![0]\!] & \text{if } r_i = 0 \\ [\![a]\!] & \text{if } r_i = 1 \end{cases} \tag{3.9}$$

and

$$[\![a]\!]^{1-2r_i} = [\![a\ (1-2r_i)]\!] = \begin{cases} [\![a]\!] & \text{if } r_i = 0 \\ [\![a]\!]^{-1} & \text{if } r_i = 1. \end{cases} \tag{3.10}$$

On average only $\ell/2$ `exp` are actually needed, when $1 - 2r_i = -1$ and $[\![0]\!]$ can be precomputed. Table 3.12 summarizes the complexities of the toEncBit protocol.

Rounds	Bandwidth	# `exp`
2ℓ	$(3\ell-1)2T$	$6\ell + \ell/2 - 2$

Table 3.12: toEncBit: average complexities

3.4.12 Comparison of encrypted numbers

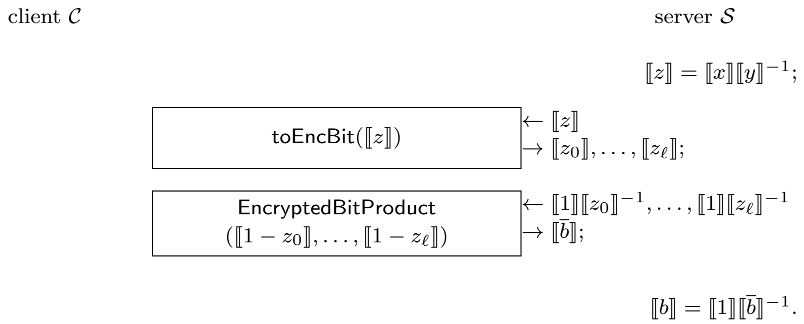
In this section we describe how two encrypted numbers can be compared. We limit the analysis to the computation of $[\![[x \overset{?}{=} y]]\!]$ and $[\![[x \overset{?}{<} y]]\!]$. All the other comparisons can be derived from these two building blocks. Letting $z = x - y$, we can observe that $[\![[x \overset{?}{=} y]]\!] = [\![[x - y \overset{?}{=} 0]]\!] = [\![[z \overset{?}{=} 0]]\!]$. Similarly $[\![[x \overset{?}{<} y]]\!] = [\![[z \overset{?}{<} 0]]\!]$.

Equality check

We consider the case in which $\mathcal{S}$ needs to obtain an encrypted bit $[\![b]\!]$ representing if two values x and y (ℓ bits long) are equal and one of them, or both, is encrypted. The protocol is described in Protocol 8.

EQUALITY COMPARISON PROTOCOL

inputs of $\mathcal{C}$: Nothing
inputs of $\mathcal{S}$: $[\![x]\!], y$
output for $\mathcal{C}$: Nothing
output for $\mathcal{S}$: $[\![b]\!] = [\![[x \overset{?}{=} y]]\!]$


client $\mathcal{C}$
server $\mathcal{S}$
$[\![z]\!] = [\![x]\!][\![y]\!]^{-1};$
toEncBit($[\![z]\!]$)
$\leftarrow [\![z]\!]$
$\rightarrow [\![z_0]\!], \ldots, [\![z_\ell]\!];$
EncryptedBitProduct
$([\![1 - z_0]\!], \ldots, [\![1 - z_\ell]\!])$
$\leftarrow [\![1]\!][\![z_0]\!]^{-1}, \ldots, [\![1]\!][\![z_\ell]\!]^{-1}$
$\rightarrow [\![\bar{b}]\!];$
$[\![b]\!] = [\![1]\!][\![\bar{b}]\!]^{-1}.$

Protocol 8: Interactive protocol that evaluates if two encrypted values are equal.

By observing that z ($\ell + 1$ bits long) is equal to 0 only if all its bits are null, $[z \overset{?}{=} 0] = 1 - \wedge_{i=0}^{\ell} \overline{z_i}$, where $\overline{z_i}$ denotes negation. The equality can be evaluated as

$$[z \overset{?}{=} 0] = 1 - \prod_{i=0}^{\ell} (1 - z_i).$$

$\prod_{i=0}^{\ell}(1 - z_i)$, named EncryptedBitProduct in the Protocol, can be computed in several ways. The most intuitive one is the following:

```
$[\![\bar{b}_1]\!] = \mathsf{eMul}([\![1 - z_0]\!], [\![1 - z_1]\!]);$
for $i = 2, \ldots, \ell$
    $[\![\bar{b}_i]\!] = \mathsf{eMul}([\![\bar{b}_{i-1}]\!], [\![1 - z_i]\!]);$
end
$[\![\bar{b}]\!] = [\![\bar{b}_\ell]\!].$
```

This solution is not optimal and requires $\mathcal{O}(\ell)$ rounds. Another implementation can be developed by considering that $\prod_{i=0}^{\ell}(1 - z_i) = \prod_{i=0}^{\ell/2}(1 - z_i) \cdot \prod_{i=\ell/2+1}^{\ell}(1 - z_i)$ and each product can be recursively splitted into products between two elements. In this way we obtain $\mathcal{O}(\log_2 \ell)$ rounds where several eMul functions are run in parallel. Note that being each $1 - z_i$ a bit, it is not necessary to obfuscate it by adding long random values, since they can be perfectly obfuscated by computing the XOR with a random bit. Given two bits $\overline{z_i}$, $\overline{z_j}$, they are obfuscated by computing $\overline{z_i} \oplus r_i = \overline{z_i} + r_i - 2\overline{z_i} r_i$ and $\overline{z_j} \oplus r_j = \overline{z_j} + r_j - 2\overline{z_j} r_j$, resulting in

$$\begin{aligned} w &= (\overline{z_i} + r_i - 2\overline{z_i} r_i)(\overline{z_j} + r_j - 2\overline{z_j} r_j) \\ &= \overline{z_i}\,\overline{z_j}(1 - 2r_i - 2r_j + 4r_ir_j) + \overline{z_i}(r_j - 2r_ir_j) + \overline{z_j}(r_i - 2r_ir_j) + r_ir_j. \end{aligned} \tag{3.11}$$

Having computed $\llbracket w \rrbracket$, $\mathcal{S}$ can remove the obfuscation to obtain $\overline{z_i z_j}$:

$$\overline{z_i z_j} = (\llbracket w \rrbracket \llbracket \overline{z_i} \rrbracket^{2r_ir_j - r_i} \llbracket \overline{z_j} \rrbracket^{2r_ir_j - r_j} \llbracket -r_ir_j \rrbracket)^{1/(1-2r_i-2r_j+4r_ir_j)}, \tag{3.12}$$

where

$$\llbracket \overline{z_i} \rrbracket^{2r_ir_j - r_i} = \begin{cases} \llbracket \overline{z_i} \rrbracket & \text{if } r_i = 1 \wedge r_j = 0 \\ \llbracket \overline{z_i} \rrbracket^{-1} & \text{if } r_i = 1 \wedge r_j = 1 \\ 1 & \text{else,} \end{cases} \tag{3.13}$$

$$\llbracket \overline{z_j} \rrbracket^{2r_ir_j - r_j} = \begin{cases} \llbracket \overline{z_j} \rrbracket & \text{if } r_i = 0 \wedge r_j = 1 \\ \llbracket \overline{z_j} \rrbracket^{-1} & \text{if } r_i = 1 \wedge r_j = 1 \\ 1 & \text{else,} \end{cases} \tag{3.14}$$

$$1/(1 - 2r_i - 2r_j + 4r_ir_j) = \begin{cases} 1 & \text{if } r_i = r_j \\ -1 & \text{if } r_i \neq r_j. \end{cases} \tag{3.15}$$

In this way in each round all the obfuscated bits can be packed together during the transmission from $\mathcal{S}$ to $\mathcal{C}$. From a computational point of view, each eMul requires 1 `exp`, 1 `enc` and 1 `dec` on the average. The total number of rounds is equal to $2\ell + 2 + 2\lceil \log_2 \ell \rceil$, where the first $2\ell + 2$ are for the toEncBit protocol and the others for the rest of the computation. In addition to the data transmitted during the toEncBit protocol, $\lceil \log_2 \ell \rceil$ cyphertexts are transmitted from $\mathcal{S}$ to $\mathcal{C}$ (1 cyphertext during each round by using packing), and ℓ cyphertexts from $\mathcal{C}$ to $\mathcal{S}$. The total number of cyphertexts transmitted during the whole protocol is $\lceil \log_2 \ell \rceil + 4\ell$. The protocol requires 1 `exp` to compute z, $6\ell + \ell/2 + 4$ `exp` for the toEncBit protocol, $\ell + 1$ `exp` to compute

$\overline{z_i}$, around ℓ `exp` to pack the values, $\lceil \log_2 \ell \rceil$ `dec` and ℓ `enc` on the $\mathcal{C}$ side, ℓ `exp` on average to remove the obfuscations and 1 `exp` to obtain b. The total complexity is shown in Table 3.13.

Rounds	Bandwidth	# `exp`
$2\ell + 2\lceil \log_2 \ell \rceil + 2$	$(4\ell + \lceil \log_2 \ell \rceil)2T$	$10\ell + \ell/2 + \lceil \log_2 \ell \rceil + 7$

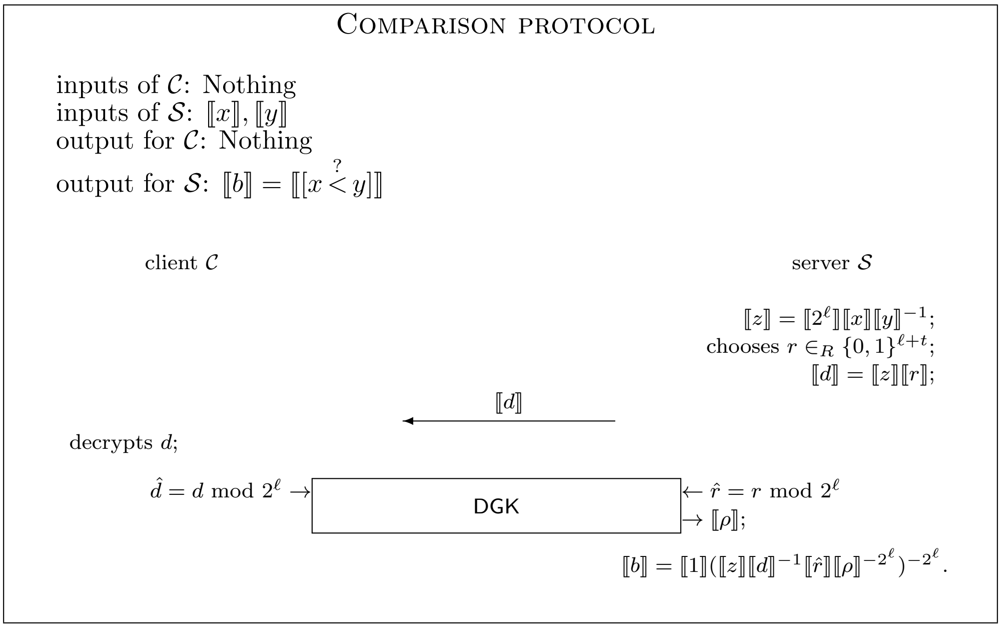
Table 3.13: Equality comparison: average complexities

Inequality check

Similarly to the equality check, $\mathcal{S}$ needs to compare two values x and y (ℓ bits long) and one of them, or both, is encrypted. It is possible to obtain $[\![b]\!] = [\![[x \overset{?}{<} y]]\!]$ by computing $[\![z]\!] = [\![2^\ell + x - y]\!] = [\![2^\ell]\!][\![x]\!][\![y]\!]^{-1}$, where 2^ℓ is used to map the difference in a positive range, and extracting its most significant bit, that is equal to 0 if and only if $x < y$, hence $[\![b]\!] = [\![1 - z_\ell]\!] = [\![1]\!][\![z_\ell]\!]^{-1}$. To extract the most significant bit we could use the toEncBit protocol, but this method is not efficient for this particular application where we are interested only in the most significant bit. A most efficient implementation [EFG+09] is described in Protocol 9.

The idea is that if $\mathcal{S}$ had an encryption of $z \bmod 2^\ell$, z_ℓ could be computed as $z_\ell = 2^{-\ell}(z - (z \bmod 2^\ell))$. $\mathcal{S}$ obfuscates z with a random value r and sends the result d to $\mathcal{C}$. At this point they start an interactive protocol to obtain $z \bmod 2^\ell = ((d \bmod 2^\ell) - r \bmod 2^\ell) \bmod 2^\ell = (\hat{d} - \hat{r}) \bmod 2^\ell = \tilde{z}$ transforming the problem back to a comparison between two values represented with the same number of bits. Note that if $\hat{d} \geq \hat{r}$, $\tilde{z}$ is the correct result. On the other hand, if $\hat{r}$ is larger an underflow occurs and we need to add 2^ℓ to obtain the correct result. Given $[\![\rho]\!]$, where ρ is a binary value indicating whether $\hat{r} > \hat{d}$ and is obtained by the DGK protocol (Protocol 10), $\mathcal{S}$ can compute $[\![z \bmod 2^\ell]\!] = [\![\tilde{z}]\!][\![\rho]\!]^{2^\ell} = [\![\tilde{z} + 2^\ell \rho]\!]$.

For efficiency reasons, in [EFG+09], the authors use a different homomorphic encryption scheme, namely the one proposed by Damgård et al. [DGK07, DGK09]. For simplicity we continue to use the Paillier scheme. Initially, $\mathcal{C}$ sends the encryptions of the bits $[\![\hat{d}_0]\!], \ldots, [\![\hat{d}_{\ell-1}]\!]$ to $\mathcal{S}$ that, then, randomly chooses $s \in_R \{1, -1\}$ and computes $[\![c_i]\!]$. Let us consider the case



COMPARISON PROTOCOL

inputs of $\mathcal{C}$: Nothing
inputs of $\mathcal{S}$: $[\![x]\!], [\![y]\!]$
output for $\mathcal{C}$: Nothing
output for $\mathcal{S}$: $[\![b]\!] = [\![[x \overset{?}{<} y]]\!]$

client $\mathcal{C}$
server $\mathcal{S}$
$[\![z]\!] = [\![2^\ell]\!][\![x]\!][\![y]\!]^{-1}$;
chooses $r \in_R \{0,1\}^{\ell+t}$;
$[\![d]\!] = [\![z]\!][\![r]\!]$;
$[\![d]\!]$
decrypts d;
$\hat{d} = d \bmod 2^\ell \rightarrow$
DGK
$\leftarrow \hat{r} = r \bmod 2^\ell$
$\rightarrow [\![\rho]\!]$;
$[\![b]\!] = [\![1]\!]([\![z]\!][\![d]\!]^{-1}[\![\hat{r}]\!][\![\rho]\!]^{-2^\ell})^{-2^\ell}$.

Protocol 9: Interactive protocol that evaluates $[x \overset{?}{<} y]$.

$s = 1$: if $\hat{d}$ is larger than $\hat{r}$, then all c_i's will be non-zero, otherwise, if $\hat{r}$ is larger than $\hat{d}$, exactly one c_i will be equal to zero, the one at the most significant differing bit-position. For $s = -1$ we have exactly the same situation, except that the zero occurs if $\hat{d}$ is larger. The factor of 3 that amplifies the sum ensures that the values are non-zero at least once even if a single w_j is set to 1. $\mathcal{S}$ now multiplicatively blinds the $[\![c_i]\!]$ with a random R_i and permutes the encryptions $[\![e_i]\!]$. For a secure multiplicative obfuscation we need to use random numbers as long as the ring length, hence it is not possible to pack the e_i values. At this point $\mathcal{S}$ sends them to $\mathcal{C}$, that now decrypts all e_i's and checks whether one of them is zero. It then encrypts a bit ρ stating if this is the case.

Analysing the protocol it is easy to observe that it requires 4 rounds. At the beginning $\mathcal{S}$ needs to compute 1 `enc` and 1 `exp` to compute $[\![z]\!]$ and 1 `enc` to obfuscate it. At this point 1 cyphertext is transmitted to $\mathcal{C}$ that performs 1 `dec`. During the DGK protocol we have ℓ `enc` on $\mathcal{C}$'s side, followed by the transmission of ℓ cyphertexts. At this point on $\mathcal{S}$'s side we have $\ell + 1$ `enc` for

DGK PROTOCOL

inputs of $\mathcal{C}$: $\hat{d}_0, \ldots, \hat{d}_{\ell-1}$
inputs of $\mathcal{S}$: $\hat{r}_0, \ldots, \hat{r}_{\ell-1}$
output for $\mathcal{C}$: Nothing
output for $\mathcal{S}$: $\llbracket \rho \rrbracket = \llbracket [\hat{d} \overset{?}{>} \hat{r}] \rrbracket$

```
client $\mathcal{C}$                                                    server $\mathcal{S}$

encrypts $\hat{d}_i \ \forall i$;
                        $\xrightarrow{\llbracket \hat{d}_0 \rrbracket, \ldots, \llbracket \hat{d}_{\ell-1} \rrbracket}$
                                                          chooses $s \in_R \{1, -1\}$;
                                                          for $i = \ell - 1$ downto $0$
                                                              $\llbracket w_i \rrbracket = \llbracket \hat{d}_i \oplus \hat{r}_i \rrbracket = \llbracket \hat{r}_i \rrbracket \llbracket \hat{d}_i \rrbracket^{1-2\hat{r}_i}$;
                                                              $\llbracket c_i \rrbracket = \llbracket \hat{d}_i \rrbracket \llbracket \hat{r}_i \rrbracket^{-1} \llbracket s \rrbracket (\prod_{j=i+1}^{\ell-1} \llbracket w_j \rrbracket)^3$
                                                                  $= \llbracket \hat{d}_i - \hat{r}_i + s + 3 \sum_{j=i+1}^{\ell-1} w_j \rrbracket$;
                                                              chooses $R_i \in_R \mathbb{Z}_N$;
                                                              $\llbracket e_i \rrbracket = \llbracket c_i \rrbracket^{R_i}$;
                                                          endfor;
                                                          permute $(\llbracket e_0 \rrbracket, \ldots, \llbracket e_{\ell-1} \rrbracket)$;
                        $\xleftarrow{\text{permuted } (\llbracket e_0 \rrbracket, \ldots, \llbracket e_{\ell-1} \rrbracket)}$
decrypts $e_i \ \forall i$;
if $0 \in \{e_0, \ldots, e_{\ell-1}\}$ then
    $\llbracket \rho \rrbracket = \llbracket 1 \rrbracket$;
else
    $\llbracket \rho \rrbracket = \llbracket 0 \rrbracket$;
end
                        $\xrightarrow{\llbracket \rho \rrbracket}$
```

Protocol 10: Interactive DGK protocol.

the $\hat{r}_i$ and s values. Considering that $\llbracket \hat{d} \rrbracket^{1-2\hat{r}_i}$ requires exponentiation only if $r_i = 1$, in each of the ℓ occurrences of the FOR cycle we have $3 + 1/2$ `exp`. At this point we have the transmission of ℓ cyphertexts to $\mathcal{C}$ that performs ℓ `dec` and 1 `enc`. DGK protocol ends with the transmission of 1 cyphertext. Observing the final formula that computes $\llbracket b \rrbracket$ we can observe that 2 `enc` ($\llbracket 1 \rrbracket$ and $\llbracket \hat{r} \rrbracket$) and 3 `exp` are required. The total complexity is shown in Table 3.14.

Rounds	**Bandwidth**	# `exp`
4	$(2\ell+2)2T$	$6\ell+\ell/2+11$

Table 3.14: Minor comparison: average complexities

3.4.13 Minimum selection

Starting from the comparison protocol, we can evaluate $z = \min\{x, y\}$ [EFG+09] as described in Protocol 11.

eMin PROTOCOL

inputs of $\mathcal{C}$: Nothing
inputs of $\mathcal{S}$: $[\![x]\!], [\![y]\!]$
output for $\mathcal{C}$: Nothing
output for $\mathcal{S}$: $[\![z]\!] = \mathsf{eMin}([\![x]\!], [\![y]\!])$

client $\mathcal{C}$ server $\mathcal{S}$

$\boxed{[\![b]\!] = [\![x \overset{?}{<} y]\!]}$ $\leftarrow [\![x]\!], [\![y]\!]$
$\rightarrow [\![b]\!]$;

$\boxed{[\![z]\!] = \mathsf{Multiplexer}([\![b]\!], [\![x]\!], [\![y]\!])}$ $\leftarrow [\![b]\!], [\![x]\!], [\![y]\!]$
$\rightarrow [\![z]\!]$.

Protocol 11: Interactive protocol that computes the minimum among two values.

Since the protocol is obtained by composing the $[\![x \overset{?}{<} y]\!]$ and the Multiplexer protocols, its complexity, shown in Table 3.15, is easily obtained by summing the complexities of $[\![x \overset{?}{<} y]\!]$ and Multiplexer.

Rounds	**Bandwidth**	# `exp`
6	$(2\ell+4)T$	$6\ell+\ell/2+17+1/2$

Table 3.15: eMin: average complexities

The minimum selection can be easily extended to more values by using the scheme in Figure 3.2. Moreover if the index of the minimum value is required it is sufficient to compute $[\![i_{min}]\!] = \mathsf{Multiplexer}([\![b]\!], [\![i]\!], [\![j]\!])$ together with $[\![x_{min}]\!] = \mathsf{Multiplexer}([\![b]\!], [\![x_i]\!], [\![x_j]\!])$ in each eMin.

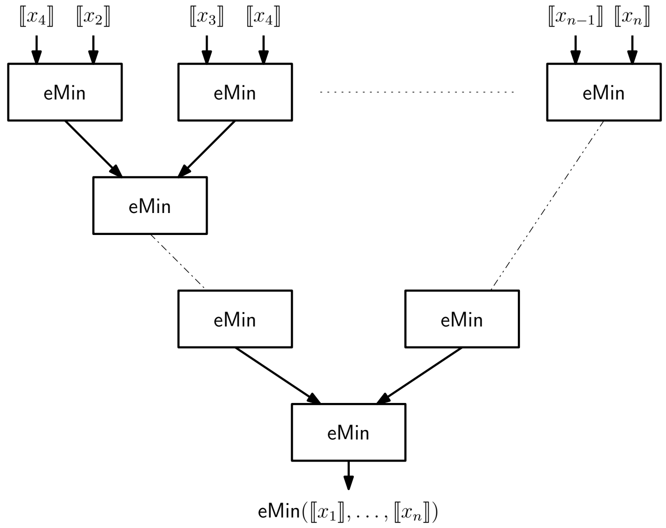


Figure 3.2: eMin protocol extended to more values.

3.4.14 Distances

In this section we show how the distance between two arrays can be computed, starting with the Euclidean distance and later analysing the Hamming distance.

Euclidean distance

Let suppose that $\mathcal{S}$ needs to compute the euclidean distance $\delta = d_E(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$ between a vector $[\![\mathbf{x}]\!] = [[\![x_1]\!], \ldots, [\![x_n]\!]]$ encrypted with the public key of $\mathcal{C}$ and a vector $\mathbf{y} = [y_i, \ldots, y_n]$ available in plain [EFG+09]. Due to the difficulty to compute the square root, we assume that $\mathcal{S}$ needs only to compute δ^2. eEuclideanDistance is shown in Protocol 12.

The distance computation can be performed by using the eEnergy protocol on a vector $\mathbf{d}$ whose elements are defined as the difference between the elements of $\mathbf{x}$ and the elements of $\mathbf{y}$. The complexity of the protocol can be obtained by the complexity of eEnergy, adding the effort to compute the $[\![d_i]\!]$ values (n `enc` and n `mult`). The total complexity is shown in Table 3.16.

Note that the protocol does not change if $\mathbf{y}$ is available to $\mathcal{S}$ only in eEuclideanDistance PROTOCOL

inputs of $\mathcal{C}$: Nothing
inputs of $\mathcal{S}$: $[\![\mathbf{x}]\!], \mathbf{y}$
output for $\mathcal{C}$: Nothing
output for $\mathcal{S}$: $[\![\delta^2]\!] = \mathsf{eEuclideanDistance}([\![\mathbf{x}]\!], \mathbf{y})$

client $\mathcal{C}$ server $\mathcal{S}$

$\forall i = 1 \ldots n:$
$[\![d_i]\!] = [\![x_i]\!][\![-y_i]\!] = [\![x_i - y_i]\!];$

$\boxed{[\![\delta^2]\!] = \mathsf{eEnergy}([\![\mathbf{d}]\!])}$ $\leftarrow [\![\mathbf{d}]\!] = [[\![d_1]\!], \ldots, [\![d_n]\!]]$
$\rightarrow [\![\delta^2]\!]$.

Protocol 12: Interactive protocol that computes the euclidean distance between two vectors.

Rounds	Bandwidth	# `exp`
2	$(p+1)2T$	$3n+2$

Table 3.16: eEuclideanDistance - complexities

encrypted form. In this case the complexity of the protocol is equal to the complexity of the eEnergy protocol.

If the vectors $\mathbf{x}$ and $\mathbf{y}$ are the inputs to the protocol respectively from $\mathcal{C}$ and $\mathcal{S}$, it is possible to evaluate the distance without interaction. $\mathcal{C}$ sends $[\![\sum_{i=1}^n x_i^2]\!]$ together with the encryption of the vector elements. $\mathcal{S}$ computes the distance as $[\![\delta^2]\!] = [\![\sum_{i=1}^n (x_i - y_i)^2]\!] = [\![\sum_{i=1}^n x_i^2 + \sum_{i=1}^n (-2x_i y_y) + \sum_{i=1}^n y_i^2]\!] = [\![\sum_{i=1}^n x_i^2]\!] \prod_{i=1}^n [\![x_i]\!]^{-2y_i} [\![\sum_{i=1}^n y_i^2]\!]$.

By using this solution no additional rounds are necessary and the protocol needs only the transmission of an additional cyphertext. To compute the distance n `exp` and 1 `enc` are performed. The total complexity is shown in Table 3.17.

Rounds	Bandwidth	# `exp`
0	$2T$	$n+1$

Table 3.17: eEuclideanDistance with $\mathbf{x}$ provided by $\mathcal{C}$ - complexities

Hamming distance

As in the Euclidean distance case, $\mathcal{S}$ is interested to compute the distance between $[\![\mathbf{x}]\!]$ and $\mathbf{y}$. If the elements of the vectors are binary, $x_i \in \{0,1\}$ and $y_i \in \{0,1\}$ $\forall i = 1,\ldots,n$, $\mathcal{S}$ can compute the Hamming distance without interaction with $\mathcal{C}$. Considering that the Hamming distance is defined as $d_H(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{n}(x_i \oplus y_i) = \sum_{i=1}^{n}(x_i + y_i - 2x_iy_i) = \sum_{i=1}^{n}(y_i + (1-2y_i)x_i)$, $\mathcal{S}$ can compute it in the encrypted domain as

$$[\![d_H([\![\mathbf{x}]\!],\mathbf{y})]\!] = [\![\sum_{i=1}^{n}(x_i \oplus y_i)]\!] = [\![\sum_{i=1}^{n} y_i]\!]\prod_{i=1}^{n}[\![x_i]\!]^{1-2y_i}.$$

The protocol has zero communication complexity. Considering that $1 - 2y_i \in \{-1,1\}$ is known to $\mathcal{S}$, $\mathcal{S}$ can compute the exponentiation only when $1-2y_i = -1$, hence the protocol requires 1 `enc` to encrypt the sum of y_i values, $n/2$ `exp` and n `mult`. The complexities are summarized in Table 3.18.

Rounds	**Bandwidth**	**#** `exp`
0	0	$n/2+1$

Table 3.18: Hamming Distance - average complexities

If both $\mathbf{x}$ and $\mathbf{y}$ are available as encrypted numbers we need to parallely compute $[\![x_iy_i]\!] = \mathsf{eMul}(x_i,y_i)$ $\forall i$, where eMul can be optimised as in Section 3.4.12 by considering that both x_i and y_i are bits.

3.5 Applications of homomorphic-based STPC

In addition to the basic protocols illustrated in the previous section, several complex applications of homomorphic-based STPC have been proposed. The use of the basic tools described to build protocols implementing more complex operations is a difficult task, requiring that different ad-hoc solutions be developed for each different operation. For this reason, instead of attempting to develop a general s.p.e.d. theory working at the protocol level, the most common signal processing tools and ad hoc protocols implementing them in the encrypted domain have been developed. Since they are not closely related to this thesis, we briefly review only two of them.

3.5.1 Discret Fourier Transform

The discrete Fourier Transform (DFT) is a discrete transform, used in Fourier analysis, defined as

$$S_k = \sum_{i=1}^{n} s_i W^{ik}, \quad k = 0, 1, \ldots, n-1 \tag{3.16}$$

where $W = e^{-j2\pi/n}$ and s_i is a finite sequence of length n. Among the appealing properties of the above transform is that it can be implemented via fast algorithms, noted as Fast Fourier Transforms (FFTs).

In [BPB08b, BPB08c, BPB08a] a scenario is considered in which the transform is computed on a sample-wise encrypted version of the input vector, that is $[\![\mathbf{s}]\!] = [[\![s_1]\!], \ldots, [\![s_n]\!]]$. Since the DFT transform coefficients are public, the expression in (3.16) can be computed by relying on the homomorphic property of Paillier cryptosystem. Let us consider a signal $x_i \in \mathbb{C}$, with $x_i = x_i^R + jx_i^I$, $x^R, x^I \in \mathbb{R}$. In the following, we will assume that the signal is bounded in amplitude $|x_i^{R,I}| \leq M$. In order to process x_i in the encrypted domain, we need to approximate it by suitable integers on a finite field. The same holds for the output of the DFT. This is accomplished by the following quantisation process

$$s_i = \lceil Q_1 x_i \rfloor = \lceil Q_1 x_i^R \rfloor + j \lceil Q_1 x_i^I \rfloor = s_i^R + js_i^I, \tag{3.17}$$

where Q_1 is an integer suitable scaling factor. Being x_i bounded, the quantised signal will satisfy $-Q_1 M \leq s_i^{R,I} \leq Q_1 M$. Considering that the cryptosystem encrypts integers modulo N, we need a biunivoc mapping between $s_i^{R,I} \mod N$ and $s_i^{R,I}$, so that we can always recover the correct value of $s_i^{R,I}$ from $s_i^{R,I} \mod N$. This can be achieved by imposing $2Q_1 M + 1 \leq N$.

The coefficients W^{ik} in (3.16) can be quantised by using the same strategy as above. In particular, we define

$$C_r = \lceil Q_2 W^r \rfloor = \lceil Q_2 \cos(2\pi r/n) \rfloor - j \lceil Q_2 \sin(2\pi r/n) \rfloor = C_r^R + jC_r^I \tag{3.18}$$

where Q_2 is the DFT coefficient scaling factor. Thanks to the properties of W, we have $-Q_2 \leq C_r^{R,I} \leq Q_2$.

Based on the definitions above, the integer approximation of the DFT is:

$$
\begin{align}
S_k &= \sum_{i=1}^{n} s_i C_{ik} = \nonumber \\
&= \sum_{i=1}^{n} (s_i^R C_{ik}^R - s_i^I C_{ik}^I) + j(s_i^R C_{ik}^I + s_i^I C_{ik}^R) = \tag{3.19} \\
&= \sum_{i=1}^{n} (s_i^R + s_i^I) C_{ik}^R - s_i^I (C_{ik}^R + C_{ik}^I) \nonumber \\
&\quad + j \sum_{i=1}^{n} (s_i^R + s_i^I) C_{ik}^R - s_i^R (C_{ik}^R - C_{ik}^I) \tag{3.20}
\end{align}
$$

with $k = 0, 1, \ldots, n-1$. Note that the representation in (3.19) requires 4 products and two sums, while the representation in (3.20) requires 3 products and 3 sums, hence it is more performing for a computation in the encrypted domain. Since all computations are between integers, the expression above can be evaluated in the encrypted domain by relying on homomorphic properties and the DFT in the encrypted domain can be evaluated as

$$
\begin{aligned}
\llbracket S_k \rrbracket &\triangleq \llbracket S_k^R \rrbracket + j \llbracket S_k^I \rrbracket; \\
\llbracket S_k^R \rrbracket &= \prod_{i=1}^{n} \llbracket Z_{ik} \rrbracket \llbracket s_i^R \rrbracket^{-(C_{ik}^R + C_{ik}^I)}; \\
\llbracket S_k^I \rrbracket &= \prod_{i=1}^{n} \llbracket Z_{ik} \rrbracket \llbracket s_i^I \rrbracket^{-(C_{ik}^R - C_{ik}^I)}; \\
\llbracket Z_{ik} \rrbracket &= (\llbracket s_i^R \rrbracket \llbracket s_i^I \rrbracket)^{C_{ik}^R}.
\end{aligned}
$$

The computation of the DFT results in a frequency domain representation amplified by a factor K with respect to the frequency domain representation X_K of the signal x_i. In general, $S(k)$ can be expressed as

$$
S(k) = KX(k) + \epsilon_S(k). \tag{3.21}
$$

where the value of both $K = Q_1 Q_2$ and $\epsilon_{S,max}$ is an approximation error depending on the scaling factors Q_1 and Q_2 and on the particular implementation of the DFT. Moreover, in order to implement the DFT by using a cryptosystem which encrypts integers modulo N, one must ensure that the biunivoc mapping still holds for $S(k) \mod N$. Hence, we need to find an upper bound on $S(k)$ such that $|S_{R,I}(k)| \leq Q_S$, and verify that $2Q_S+1 \leq N$. In [BPB08c] the authors derived that $Q_S = M\lfloor Q_1Q_2 + Q_1/\sqrt{2} + Q_2/\sqrt{2} + 1/2\rfloor$. In the case of the DFT of a real signal x_n both the real and the imaginary part of the integer DFT values will satisfy $|S_{R,I}(k)| \leq Q_S = MQ_1Q_2$.

In [BPB08a], the encrypted implementation of the Fast Fourier Transform (FFT) has been analysed as well, resulting in a less efficient solution that requires a lower number of modular operation, but introduces an amplification in the output that grows with the sample index, requiring periodically interaction to rescale the output.

3.5.2 Composite signal representation

A problem with the use of homomorphic encryption is that signals need to be encrypted sample-wise, as clearly comes out from Section 3.4.3 and Section 3.5.1. Samplewise encryption of signals poses some severe complexity problems since it introduces a huge expansion factor between the original signal samples and the encrypted ones.

By considering that plain signal samples are usually represented by few bits (e.g. 8 bits for images), we conclude that due to encryption, signals are expanded by a large factor: for instance, the size of a grey level 1000×1000 image will pass from 1Mbyte in the clear to 250 Mbytes in the encrypted domain. This huge expansion factor is clearly not affordable in many practical applications.

In order to solve these problems, an alternative representation of signals was proposed in [BPB10] that permits to greatly reduce the expansion factor introduced by encryption, while still allowing the exploitation of the homomorphic properties of the underlying cryptosystem to process signals in the encrypted domain. In addition to limiting the storage requirement, the proposed representation allows the parallel processing of different samples, thus providing a considerable reduction of computational complexity in terms of operations between encrypted messages.

The authors derived a new way to pack the sample starting from the protocol described in Section 3.4.4. The idea is to bundle R l-bit samples of the signal s_i $(i = 0, \dots, n-1)$ within $M = \lceil n/R \rceil$ composite messages $s_C(k)$ as follows:

$$s_C(k) = \sum_{j=0}^{R-1} s_j(k) B^j, \quad k = 0, 1, \dots, M-1 \tag{3.22}$$

where

- B is a base sufficiently large such that samples will remain distinct in the composite representation;
- R is chosen such that there is no ambiguity in the composite message;
- the k-th element of the composite signal $s_C(k)$ is obtained by partitioning the signal in the following way: $s_j(k) = s(jM + k)$. This representation is referred to as M-polyphase composite representation (M-PCR) and its graphical interpretation is provided in Figure 3.3.

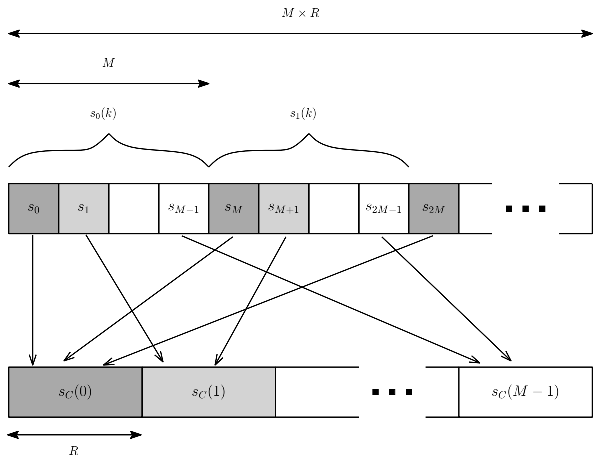


Figure 3.3: Graphical representation of M-polyphase composite representation of order R. The values inside the small boxes indicate the indices of the samples of s_i. Identically shaded boxes indicate values belonging to the same composite word.

B and R can be derived by considering that if $|s_i| < Q \; \forall i$, then

$$B > 2Q \tag{3.23}$$
$$B^R \le N. \tag{3.24}$$

If B is sufficiently large, we can correctly perform some operations on the samples; for example adding two composite messages will result in the addition of the single messages composing them (if $B > 4Q$), and multiplying the composite message by a constant factor C, will be equivalent to multiplying each single message by the same factor (if $B > 2QC$). If more operations are performed, B has to be chosen accurately. Considering that the samples can assume negative values, it is convenient to map them into positive numbers. This can be done by adding the value Q samplewise, that result in

$$\begin{aligned} s'_C(k) &= \sum_{j=0}^{R-1} (s_j(k) + Q) B^j \\ &= \sum_{j=0}^{R-1} s_j(k) B^j + Q \sum_{j=0}^{R-1} B^j \\ &= s_C(k) + Q \frac{B^R - 1}{B - 1} \\ &= s_C(k) + \omega_Q. \end{aligned} \tag{3.25}$$

Then, the following holds:

$$0 \leq a_C(k) + \omega_Q < N. \tag{3.26}$$

Moreover, the original samples can be obtained from the composite representation as

$$s_i(k) = \left\{ \left[(s_C(k) + \omega_Q) \div B^i \right] \mod B \right\} - Q. \tag{3.27}$$

Implementing the M-PCR of (3.22) in the encrypted domain is trivial, by considering that the parameters B, R, M are public. However, passing from the composite to the samplewise representation is not possible in the encrypted domain by means of homomorphic computations only, since such a conversion requires rounding and division. Then unpacking has to be carried out by the data owner P_1, or performed by means of a properly designed interactive protocol involving P_1 and P_2.

As shown in [BPB10], the M-PCR can be useful for linear transforms, convolution and linear filtering.

3.6 Fully homomorphic encryption

All the algorithms described above rely on additive homomorphic protocols such as Paillier cryptosystem. While additive (or multiplicative) encryption may be useful in several s.p.e.d. applications, the availability of a fully homomorphic cryptosystem, for which both additions and multiplications can be carried out in the encrypted domain without interaction between the parties, would be an invaluable resource, since it would dramatically increase the number of operations that could be performed on encrypted data without any interaction between the involved parties.

Devising a fully homomorphic cryptosystem has been one of the most challenging and long standing open problems of modern cryptography (see also Rivest et al. [RAD78a]) and only few proposals have been made in the literature, most of which suffer from various shortcomings, as we briefly summarize below.

Fellows and Koblitz [FK] proposed an asymmetric scheme named “Polly Cracker” which is based on the difficulty of solving systems of non-linear equations. According to the current state of knowledge, all its instantiations (and variations like PollyTwo [VL06]) are either insecure, inefficient, or loose their homomorphic property (e.g., see [FG07, dVMPT08]). Domingo-Ferrer proposed symmetric schemes based on polynomial interpolation [Fer96, DF02] but these have been broken afterwards [Wag03, Bao03, CKN06a]. Rappe [Rap04] showed constructions from (single-)homomorphic schemes over certain semigroups but for the latter no efficient solutions are known. Sander et al. [SYY99] described a fully homomorphic scheme over a semigroup, however, the homomorphism comes with the cost of a constant factor expansion per semigroup operation. Boneh et al. [BGN05] proposed a provably secure homomorphic encryption scheme that allows for arbitrary many additions but only *one* multiplication. A further problem is that the plaintext space needs to be small.

This challenging problem has recently found a positive answer thanks to a seminal paper by Gentry [Gen09]. Gentry’s cryptosystem is the most flexible full-homomorphic encryption (FHE) scheme proposed so far: it is public-key and allows arbitrary many additions and multiplications. However Gentry’s scheme its too inefficient to be used in practice.

We now provide an informal description of Gentry's scheme, as well as many other subsequent works on FHE. The encryption scheme is composed of two ingredients:

A Somewhat-Homomorphic Encryption Scheme: The starting point for FHE is a somewhat-homomorphic encryption scheme (SHE). An SHE scheme allows to compute polynomials of degree d on the encrypted boolean inputs without interaction (where d needs to be big enough for the bootstrapping). Usually SHE are based on the addition of random noise that can be easily removed during decryption., but the evaluation of polynomials of degree $> d$ produces a cyphertext containing a noise that impedes a correct decryption.

A Bootstrapping Procedure: Given an SHE with the additional where the decryption can be described by a polynomial of degree $< d$, one can obtain a FHE scheme by the following bootstrapping procedure: the public key of the FHE contains the public key of the SHE and, in addition, an *encryption of the secret key of the SHE* $C_{sk} = E_{pk}(sk)$ *under its own public key.* In practice the operation of noise reduction is performed without decryption, by using the encryption of the secret key of the SHE scheme. Now, given two encryptions of two bits $C_x = E_{pk}(x), C_y = E_{pk}(y)$ and the encryption of the secret key C_{sk}, one can compute, using the homomorphic property of the SHE, a new fresh encryption of $E_{pk}(x \mathsf{\ NAND\ } y)$! Being NAND a universal gate, any functionality can be evaluated by using an FHE scheme.

We included this brief description of Gentry's FHE scheme to stress that most of the inefficiency of FHE schemes is due to the bootstrapping process, while current SHE schemes offer reasonable performances. Therefore, while we cannot currently implement any computation using FHE (as the memory and computational requirements transcend the capability of current computers), some of the SHE schemes could soon be used in applications where one wants to compute functions of bounded but still interesting degree.

However, the practicability of Gentry's scheme is still subject of current research and discussions [Sch09, Coo09]. Furthermore, the security is based on new problems and thus requires further investigation. For this reason we see Gentry's result as a starting point for a new direction of research that aims for secure and efficient full-homomorphic encryption schemes.

Another recent construction proposed by Aguilar Melchor et al. [MGH08] is a public-key scheme as well but it is only leveled fully homomorphic, i.e., it supports only polynomials up to a fixed (but arbitrary) degree. The authors present two instantiations but the asymptotic complexity is not discussed. According to the authors, the more efficient scheme has a ciphertext length that depends exponentially on the number of multiplications and is based on a non-standard problem on lattices. Thus, the security and practicability of this solution needs to be further explored as well. In [DGHV10], the authors propose a FHE over the integers. Even if the scheme is really simple, implementation problems are related to the huge dimension of the public key.

In summary, although Gentry's result certainly represents a milestone in (theoretical) cryptography, it does not completely fulfill the requirements for s.p.e.d. applications. First, it does not work over (finite) fields which would be the natural algebraic structure for s.p.e.d. . Second, its practical efficiency is still not fully clear. Hence, the applicability for s.p.e.d. scenarios seems to be rather limited.

Chapter 4

Oblivious Transfer

Oblivious Transfer (OT) protocols allow one party, the sender $\mathcal{S}$, to transmit part of its inputs to another party, the chooser $\mathcal{C}$, in a manner that protects both of them: $\mathcal{S}$ is assured that $\mathcal{C}$ does not receive more information than it is entitled, while $\mathcal{C}$ is assured that $\mathcal{S}$ does not learn which part of the inputs is received by $\mathcal{C}$. OT is used in many privacy preserving applications, for example to sell digital goods avoiding that the seller discovers what the clients are paying for [AIR01]. Moreover, it is a key component in many applications of cryptography, as Garbled Circuits (see Chapter 5). In this chapter several protocols to implement OT are described (for their security analysis we refer to the original papers). Section 4.1 shows two basic OT protocols that let $\mathcal{C}$ obtain a message among two messages owned by $\mathcal{S}$ (1-out-2 OT), then in Section 4.2 the choice is extended among N messages (1-out-n OT). Section 4.3 illustrates how the 1-out-n OT protocols can be used to instantiate multiple parallel 1-out-2 OTs. In Section 4.4 we show how OT can be extended efficiently. Finally Section 4.5 shows how the OT can be performed in a setup phase to improve the online computation. For each protocol the communication and computation complexity are provided. The part that can be precomputed is analysed separately from the online part. When precomputation is not allowed the complexity is given by the sum of the complexities.

4.1 1-out-of-2 Oblivious Transfer

4.1.1 Protocol relying on Random Oracles

We start by presenting an efficient OT protocol that allows $\mathcal{C}$ to select among two messages owned by $\mathcal{S}$. The primitive of 1-out-of-2 Oblivious Transfer

(1-out-2 OT) was suggested by Even, Goldreich and Lempel [EGL85], as a generalization of Rabin's "oblivious transfer" [Rab81]. There have been

BELLARE AND MICALI PROTOCOL

inputs of $\mathcal{C}$: $b \in \{0,1\}$
inputs of $\mathcal{S}$: $m_0, m_1 \in \{0,1\}^t$
output for $\mathcal{C}$: m_b
output for $\mathcal{S}$: Nothing

Preliminary phase:
$\mathcal{S}$ chooses a random element $R \in_R \mathbb{Z}_N$ and publishes it together with N and a generator g.

Precomputation phase:

client $\mathcal{C}$	server $\mathcal{S}$
chooses $k \in_R \mathbb{Z}_N$; $PK_b = g^k$; $PK_{1-b} = R/PK_b$;	chooses $r_0, r_1 \in_R \mathbb{Z}_N$; precomputes g^{r_0}, g^{r_1};

Online phase:

client $\mathcal{C}$		server $\mathcal{S}$
	$\xrightarrow{PK_0}$	
		computes $PK_1 = R/PK_0$; $E_0 = H(PK_0^{r_0}) \oplus m_0$; $E_1 = H(PK_1^{r_1}) \oplus m_1$;
	$\xleftarrow{g^{r_0}, g^{r_1}, E_0, E_1}$	
$PK_b^{r_b} = (g^{r_b})^k = (g^k)^{r_b}$; $m_b = (E_b \oplus H(PK_b^{r_b}))$.		

Protocol 13: The Oblivious Transfer protocol using a Random Oracle proposed by Bellare and Micali

many suggestions for implementing OTs; the one that is considered here as starting point is proposed by Bellare and Micali [BM90], (Protocol 13), that relies on random oracle assumption. The protocol operates over a group $\mathbb{Z}_N$ (the same of the Paillier scheme) of prime order and the operations are performed in modulus N. Let g be a generator of the group, for which the computational Diffie-Hellman assumption [DH76, Sho97, NP01] holds, and $T = \lfloor \log_2 N \rfloor + 1$ the number of bits necessary to represent one element of $\mathbb{Z}_N$. The protocol uses a function $H(\cdot)$ which is assumed to be a random oracle ($H(\cdot) : \{0,1\}^T \rightarrow \{0,1\}^t$), in practice implemented by selecting the t least significant bits of a Hash function.

In the 1-out-2 OT, the sender ($\mathcal{S}$) has two strings (m_0, m_1) t bits long, while the input of the receiver ($\mathcal{C}$) is a bit b. $\mathcal{C}$ should learn m_b and nothing regarding m_{1-b}, while $\mathcal{S}$ should gain no information about b. Given a random element $R \in_R \mathbb{Z}_N$ known by both parties, $\mathcal{C}$ chooses a random $k \in_R \mathbb{Z}_N$ and computes the key $PK_b = g^k$ associated to its input bit b and derives the key associated to $\overline{b} = 1 - b$ multiplying R by the multiplicative inverse of PK_b in the ring $\mathbb{Z}_N$ ($PK_1 = R/PK_0$). Regardless of the value of the bit b, $\mathcal{C}$ sends PK_0 to $\mathcal{S}$ that derives PK_1, computes $PK_0^{r_0}$ and $PK_1^{r_1}$, where r_0 and r_1 are two random numbers, and uses them in a Hash function to obtain two bitstrings used to encrypt the messages m_0 and m_1 respectively. The encryptions, together with g^{r_0} and g^{r_1} are transmitted back to $\mathcal{C}$ that uses g^{r_b} to compute $PK_b^{r_b}$ and to decrypt the corresponding message, while he can not use $g^{r_{1-b}}$ to decrypt m_{1-b}, being $(g^{r_{1-b}})^k \neq PK_{1-b}^{r_{1-b}}$.

In this protocol $\mathcal{S}$ computes 4 `exp` (two of them can be precomputed before the protocol begins), while $\mathcal{C}$ computes 2 `exp` (one of which can be pre-computed). The communication from $\mathcal{C}$ to $\mathcal{S}$ is composed of one message of size T, and the communication from $\mathcal{S}$ to $\mathcal{C}$ is composed of two messages of size T and two elements of the same size of the inputs (t). Hence the protocol requires two rounds and the transmission of $3T + 2t$ bits.

In [NP01], the authors show that the complexity of the protocol can be reduced, without any security loss, by using the same random value on the $\mathcal{S}$'s side, namely $r = r_0 = r_1$. Since it holds that $PK_0 * PK_1 = R$, it also holds that $PK_0^r * PK_1^r = R^r$. Therefore, the online overhead of $\mathcal{S}$ is reduced to 1 `exp`, while the precomputation overhead is 2 `exp`, as before, without security loss. If $\mathcal{S}$ uses the same r for both encryptions, $\mathcal{S}$ has to transmit only a group element to $\mathcal{C}$ and the total communication is $2T + 2t$ bits. Table 4.1 summarises the complexities of the protocol.

Rounds	**Bandwidth**	# `exp`(Precomputation)	# `exp`(Online)
2	$2T + 2t$	3	2

Table 4.1: 1-out-2 OT relying on Random Oracle - Complexities

Elliptic curve based protocol The protocol proposed by Bellare-Micali, with the improvement of [NP01], can be implemented on elliptic curves.

ELLIPTIC CURVE OT BASED PROTOCOL

inputs of $\mathcal{C}$: $b \in \{0,1\}$
inputs of $\mathcal{S}$: $m_0, m_1 \in \{0,1\}^t$
output for $\mathcal{C}$: m_b
output for $\mathcal{S}$: Nothing

Preliminary phase:
$\mathcal{S}$ chooses a random point $C \in_R E(a,b)$ and publishes it together with p and a generator g.
Online phase:

client $\mathcal{C}$ server $\mathcal{S}$

chooses $k \in_R \mathbb{Z}_p$;
$PK_b = k * g$;
$PK_{1-b} = PK_b + C$;

$$\xrightarrow{PK_0}$$

$PK_1 = PK_0 + C$;
chooses $r \in_R \mathbb{Z}_p$;
$c_0 = H(r * PK_0) \oplus m_0$;
$c_1 = H(r * PK_1) \oplus m_1$;

$$\xleftarrow{r * g, c_0, c_1}$$

considering that $k * (r * g) = r * (k * g) = r * PK_b$,
$m_b = c_b \oplus H(r * PK_b) = (H(r * PK_b) \oplus m_b) \oplus H(r * PK_b)$.

Protocol 14: The Oblivious Transfer protocol based on Elliptic curves

An elliptic curve [Rab05] in a field $\mathbb{Z}_p$ used for cryptographic purposes provides the same level of security of an RSA cryptosystem with shorter messages (usually 160 bits instead of 1024 bits) and is defined as follows:

$$y^2 \bmod p = x^3 + ax + b \bmod p, \tag{4.1}$$

where a and b are integer constants in $\mathbb{Z}_p$. The elliptic curve $E(a,b)$ consists of the set of integer couples (x,y) that solves Equation 4.1, together with a single element O called the point at infinity. It is possible to define the addition operator $+$ between two points of the elliptic curve and the product operator $*$ between a point and a number (more details in [Rab05]). Given a generator point $g \in E(a,b)$ and a point $C \in E(a,b)$, known to both parties, the OT protocol on elliptic curves can be described as in Protocol 14. Similarly to the Bellare-Micali protocol, $\mathcal{C}$ computes PK_b, derives PK_{1-b} and sends PK_0 to $\mathcal{S}$ that uses it to derive PK_1 and with them encrypts m_0 and m_1 in c_0 and c_1 respectively. At this point $\mathcal{S}$ transmits c_0, c_1 and the additional value $r * g$ to $\mathcal{C}$ that uses the latter to decrypt c_b, while he is not able to decrypt c_{1-b}. The use of elliptic curves permits to perform operations on and transmit shorter cyphertexts, in fact instead to transmit group elements of size T each, points of the elliptic curve having size $2t$ are transferred, obtaining a bandwidth for the 1-out-2 OT that is $\sim 6t$.

4.1.2 Protocol not relying on Random Oracles

No real function can implement a true random oracle and even if Hash functions are usually considered secure, in high level security application a stronger assumption can be required. In [NP01] the authors propose a 1-out-2 OT having higher complexity, but not relying on random oracles. The security of the protocol is based on the Decisional Diffie-Hellman (DDH) assumption [DH76, Bon98, NP01], i.e. on the difficulty to differentiate between (g^a, g^b, g^{ab}) and (g^a, g^b, g^c) for randomly chosen a, b and c. Another similar protocol is proposed in [AIR01].

Even if protocols based on random oracles are sufficient for this thesis, we provide the description of the protocol proposed in [NP01] for completeness. The idea, shown in Protocol 15, is that $\mathcal{C}$ creates the bases of two encryption keys so that z_b, the one associated to b, is legitimate, while the other one, z_{1-b}, is badly formed and the corresponding ciphertext contains a random value. In such a way there is no need for $\mathcal{C}$ to have a public key. The two bases, together with additional data, are transmitted from $\mathcal{C}$ to $\mathcal{S}$, then $\mathcal{S}$ uses the bases to compute the encryption keys key_0 and key_1 and uses them to encrypt the messages m_0 and m_1. For the encryption it is possible to use an El Gamal like method that is semantically secure. At this point $\mathcal{S}$ transmits the cyphertexts obtained together with the additional data w_0, w_1 to $\mathcal{C}$ that is able to compute the decryption key key_b from w_b to decrypt m_b, while he is not able to compute key_{1-b}.

$\mathcal{C}$ is required to compute 3 `exp` when preparing g^{a_x}, g^{a_y} and $g^{c_b} = g^{a_x a_y}$ (as well as for selecting $g^{c_{1-b}}$ at random) and 1 `exp` and 1 `dec` online. Note

1-out-2 *OT* not relying on a Random Oracle

inputs of $\mathcal{C}$: $b \in \{0,1\}$
inputs of $\mathcal{S}$: $m_0, m_1 \in \{0,1\}^t$
output for $\mathcal{C}$: m_b
output for $\mathcal{S}$: Nothing

Precomputing phase:

client $\mathcal{C}$ | server $\mathcal{S}$

client $\mathcal{C}$:
chooses $a_x,\ a_y,\ z_{1-b} = g^{c_{1-b}} \in_R \mathbb{Z}_N$;
$x = g^{a_x};\ y = g^{a_y}$;
$c_b = a_x a_y$;
$z_b = g^{c_b}$;

server $\mathcal{S}$:
choose $(r_0, s_0), (r_1, s_1) \in_R \mathbb{Z}_N$;
precomputes g^{r_0}, g^{r_1};

Online phase:

client $\mathcal{C}$ | server $\mathcal{S}$

$$\xrightarrow{x, y, z_0, z_1}$$

server $\mathcal{S}$:
$w_0 = x^{s_0} g^{r_0}$;
$w_1 = x^{s_1} g^{r_1}$;
$key_0 = z_0^{s_0} y^{r_0}$;
$key_1 = z_1^{s_1} y^{r_1}$;
encrypts m_0 with key_0;
encrypts m_1 with key_1;

$$\xleftarrow{w_0, E[m_0], w_1, E[m_1]}$$

client $\mathcal{C}$:
$key_b = w_b^{a_y}$;
decrypts m_b.

Note that $z_b^{s_b} y^{r_b} = (g^{a_x a_y})^{s_b} (g^{a_y})^{r_b} = (g^{a_x s_b} g^{r_b})^{a_y} = (x^{s_b} g^{r_b})^{a_y} = w_b^{a_y}$

Protocol 15: The 1-out-2 Oblivious Transfer protocol not relying on a Random Oracle

that one of $\{g^{a_x}, g^{a_y}\}$ can remain fixed and there is no need to change it from transfer to transfer. Therefore $\mathcal{C}$ can compute offline 2 `exp` per transfer, and compute 1 `exp` online when it receives the message from $\mathcal{S}$. $\mathcal{S}$ has to compute 2 `exp` in the precomputation phase, and 6 `exp` and 2 `enc` online. The protocol requires 2 rounds. The transmission from $\mathcal{C}$ to $\mathcal{S}$ is composed by elements of $\mathbb{Z}_N$ represented with T bits each. Considering that one of $\{g^{a_x}, g^{a_y}\}$ remains fixed, only three elements are transmitted. During the transmission from $\mathcal{S}$ to $\mathcal{C}$ two elements of size T and two encrypted values of size $2T$ are transmitted. The total amount of data transmitted is $9T$ bits. The complexity of the protocol is shown in Table 4.2.

Rounds	**Bandwidth**	# `exp`(Precomputation)	# `exp`(Online)
2	$9T$	4	8

Table 4.2: 1-out-2 OT not relying on Random Oracle - Complexities

4.2 1-out-of-n Oblivious Transfer

Sometimes $\mathcal{C}$ can need to select among n messages hold by $\mathcal{S}$. Similarly to 1-out-2 OT, 1-out-n OT can be developed by using protocols based on random oracles or protocols not relying on random oracles.

4.2.1 Protocol relying on Random Oracles

The idea [NP01], put forward illustrated in Protocol 16, is to use the same value g^r for all the transfers between $\mathcal{S}$ and $\mathcal{C}$, and pass from 1-out-2 OT to 1-out-n OT by using n public-keys, so that $\mathcal{C}$ knows the correct secret-key of at most one message. In the precomputing phase $n-1$ public values are sent. Similarly to the 1-out-of-2 OT relying on Random Oracles, $\mathcal{C}$ transmits only PK_0; $\mathcal{S}$ uses it to obtain the other PK_i that are used in the Hash functions to encrypt the messages, while $\mathcal{C}$ computes PK_σ necessary to decrypt the desired message.

The computational overhead of $\mathcal{S}$ is consequently reduced to 1 `exp` per transfer regardless of the number of inputs. The precomputation phase, that can be used for all subsequent transfers, consists of n `exp` on $\mathcal{S}$ side and 1 `exp` on the $\mathcal{C}$ side. In each online phase, $\mathcal{S}$ performs only 1 `exp` per transfer, plus $n-1$ `mult` and n calls to $H(\cdot)$. During the online phase $\mathcal{C}$ computes 1 `exp`, 1 `mult` and an hash function. During the precomputation phase, $\mathcal{S}$ transmits n values of size T. Remember that this operation is performed only once for multiple OTs. The online phase requires two rounds and the communication overhead from $\mathcal{C}$ to $\mathcal{S}$ corresponds to one element of size T while the communication from $\mathcal{S}$ to $\mathcal{C}$ corresponds to n strings of size t and one element of size T. Table 4.3 shows the complexity of the protocol.

1-OUT-n *OT* PROTOCOL

inputs of $\mathcal{C}$: $\sigma \in \{0, \dots, n-1\}$
inputs of $\mathcal{S}$: $m_0, \dots m_{n-1} \in \{0,1\}^t$
output for $\mathcal{C}$: m_σ
output for $\mathcal{S}$: Nothing

Precomputation phase:

client $\mathcal{C}$ — server $\mathcal{S}$

server: chooses $r, R_i \in_R \mathbb{Z}_N, \; i = 1, \dots, n-1$;
(it will hold that $PK_0 \cdot PK_i = R_i$);
precomputes $R_i^r \; \forall i = 1, \dots, n-1$;

client: chooses $k \in_R \mathbb{Z}_N$;
$PK_\sigma = g^k$;

$$\xleftarrow{R_1, \dots, R_{n-1}, g^r}$$

client: computes decryption key
$PK_\sigma^r = (g^r)^k$;

Online phase:

client $\mathcal{C}$ — server $\mathcal{S}$

client: **if** $(\sigma \neq 0)$ **then** $PK_0 = R_\sigma / PK_\sigma$;

$$\xrightarrow{PK_0}$$

server: computes PK_0^r;
chooses $R \in_R \mathbb{Z}_N$;
for $i = 1, \dots, n-1$
$PK_i^r = R_i^r / PK_0^r$
without exponentiations;
$E_i = H(PK_i^r, R, i) \oplus m_i$;
endfor

$$\xleftarrow{E_0, \dots, E_{n-1}, R}$$

client: $m_\sigma = E_\sigma \oplus H(PK_\sigma^r, R, \sigma)$.

Protocol 16: 1-out-of-n Oblivious Transfers with the same g^r

Rounds	**Bandwidth**	# `exp`(Precomputation)	# `exp`(Online)
2	$nt + 2T$	$n + 1$	2

Table 4.3: 1-out-n *OT* relying on Random Oracle - Complexities

4.2.2 Protocol not relying on Random Oracles

A protocol that does not rely on the random oracle (Protocol 17) can be obtained by generalizing the 1-out-2 *OT* implemented in Protocol 15, without increasing the complexity on the $\mathcal{C}$'s side. Suppose that $\mathcal{C}$ is interested in m_σ. Then it chooses $c_\sigma = a_x a_y$ and sets $z_\sigma = g^{c_\sigma}$. The other z_j's can be defined as $z_j = z_\sigma g^{j-\sigma}$ and in particular $z_0 = g^{c_\sigma} g^{-\sigma}$. Therefore $\mathcal{C}$ sends $x = g^a, y = g^b$ and z_0. On the other side $\mathcal{S}$ computes each w_j and $z_j = z_0 g^j$ (note that g^j are values that can be computed only once and used for all the *OT*s) used to build the keys used for the message encryption. Then $\mathcal{S}$ transmits the encryptions and the w_j vales to $\mathcal{C}$ that uses w_σ to obtain key_σ

1-OUT-n *OT* NOT RELYING ON A RANDOM ORACLE

inputs of $\mathcal{C}$: $\sigma \in \{0, \dots, n-1\}$
inputs of $\mathcal{S}$: $m_0, \dots, m_{n-1} \in \{0,1\}^t$
output for $\mathcal{C}$: m_σ
output for $\mathcal{S}$: Nothing

Precomputation phase:

client $\mathcal{C}$	server $\mathcal{S}$
chooses $a_x,\ a_y \in \mathbb{Z}_N$;	**for** $j = 0, \dots, n-1$
$x = g^{a_x}$; $y = g^{a_y}$;	chooses $r_j, s_j \in_R \mathbb{Z}_N$;
$c_\sigma = a_x a_y$;	precomputes g^j, g^{r_j};
$z_\sigma = g^{c_\sigma}$;	**endfor**

Note that $z_0 = z_\sigma g^{-\sigma}$;
Online phase:

client $\mathcal{C}$		server $\mathcal{S}$
	$\xrightarrow{x, y, z_0}$	
		for $j = 0, \dots, n-1$
		$z_j = z_0 g^j$;
		$w_j = x^{s_j} g^{r_j}$;
		$key_j = z_j^{s_j} y^{r_j}$;
		encrypts m_j with key_j;
		endfor
	$\xleftarrow{w_j, E[m_j] \ \forall j = 0, \dots, n-1;}$	
$key_\sigma = w_\sigma^{a_y}$;		
decrypts m_σ.		

Protocol 17: The 1-out-n Oblivious Transfer protocol not relying on a Random Oracle

ad finally decrypt m_σ.

$\mathcal{C}$ needs to precompute 3 `exp` when preparing g^{a_x}, g^{a_y} and $g^{a_x a_y}$ (and as before one of $\{g^{a_x}, g^{a_y}\}$ can remain fixed), while 1 `dec` plus 1 `exp` are computed online to obtain m_σ. We can observe that $\mathcal{C}$ performs roughly as much work as in the random oracle *OT* construction of Protocol 16. $\mathcal{S}$ precomputes $2n$ `exp`, while in the online phase computes $3n$ `exp` and n `dec`. The protocol requires two rounds. $\mathcal{C}$ sends three elements of size T and receives from the sender n elements of size T and n encrypted values of size $2T$. The total complexity of the protocol is illustrated in Table 4.4.

Rounds	**Bandwidth**	# `exp`(Precomputation)	# `exp`(Online)
2	$(3n+3)T$	$2n+2$	$4n+2$

Table 4.4: 1-out-n *OT* not relying on Random Oracle: complexities

4.3 Bandwidth/Computation tradeoff for parallel 1-out-of-2 Oblivious Transfer

When many 1-out-2 *OT* are needed, the computation overhead of the single 1-out-2 *OT* can be reduced by performing m 1-out-2 *OT*s with the same transfer using a 1-out-n *OT* protocol. The idea [NP01] is to translate m calls to 1-out-2 *OT* into a 1-out-n *OT* for $n = 2^m$. While this reduces the computation complexity it increases the communication bandwidth, obtaining a computation/communication trade-off. If the number of 1-out-2 *OT*s is large, it could be convenient to partition them in blocks of size n.

Given m pairs $\{m_{i,0}, m_{i,1}\}_{i=1}^{m}$, $\mathcal{S}$ defines $n = 2^m$ strings $(M_0, \ldots, M_{n-1})$, corresponding to all combinations of m strings, one for each pair. Instead of engaging in m 1-out-2 *OT*s, the parties can engage in a single 1-out-n *OT* of one of these strings, as in Protocol 18.

The initialisation phase requires n `exp` by $\mathcal{S}$, but these are exploited for all the blocks ever sent by $\mathcal{S}$. During the transfer phase, $\mathcal{S}$ has to perform 1 `exp` and $n = 2^m$ `mult`, while $\mathcal{C}$ performs 1 `exp` when it sends the request and 1 `exp` when it receives the message (note that both of them can be done offline). Hence the on-line computational overhead per each 1-out-2 *OT* is $1/m$ `exp`,

Parallel 1-out-2 *OT* Protocol

inputs of $\mathcal{C}$: $b_i \in \{0,1\},\ i = 1,\dots,m$
inputs of $\mathcal{S}$: $(m_{i,0}, m_{i,1}),\ i = 1,\dots,m$
output for $\mathcal{C}$: $m_{i,b_i},\ i = 1,\dots,m$
output for $\mathcal{S}$: Nothing

Precomputing phase:

client $\mathcal{C}$ — server $\mathcal{S}$

chooses $k_{i,b} \in_R \mathbb{Z}_{2^t}\ \forall i \in \{1,\dots,m\}, b \in \{0,1\}$;
$\forall j =< j_1,\dots,j_m >\in \{1,\dots,n\}$:
computes $M_j =< k_{1,j_1}, k_{2,j_2},\dots,k_{m,j_m} >$;
chooses $K_j \in_R \mathbb{Z}_{2^t}$;
computes $M'_j = M_j \oplus H(K_j, j)$;

$$\xleftarrow{M'_j\ \forall j \in \{1,\dots,n\};}$$

Online phase:

client $\mathcal{C}$ — server $\mathcal{S}$

$\forall i \in \{1,\dots,m\}, b \in \{0,1\}$:
computes $s =< b_1,\dots,b_m >$; — computes $c_{i,b} = Enc_{k_{i,b}}(m_{i,b})$;

$s \rightarrow$ 1-out-n *OT* $\leftarrow K_1,\dots,K_n$
$K_s \leftarrow$

$$\xleftarrow{c_{i,b}\ \forall i \in \{1,\dots,m\}, b \in \{0,1\};}$$

$M_s =< k_{1,b_1},\dots,k_{m,b_m} >= M'_s \oplus H(K_s, s)$;
$m_{i,b_i} = Dec_{k_{i,b_i}}(c_{i,b_i})\ \forall i = 1,\dots,m.$

Protocol 18: Parallel 1-out-2 *OT* Protocol

for both $\mathcal{S}$ and $\mathcal{C}$, moreover $\mathcal{S}$ performs $2^m/m$ `mult`. It may seem as if the communication is $m \times n$ times the size of an input element, since there are n messages M_i of length mt bits each. However it is first possible to distinguish between group elements (which might be long) and private-keys (which can be short), furthermore another distinction can be done between online and offline communication, where the latter refers to messages that can be sent independently of the inputs. Only a single group element is sent from $\mathcal{C}$ to $\mathcal{S}$. The online communication from $\mathcal{S}$ to $\mathcal{C}$ can be reduced from $\mathcal{O}(m \times n)$ to $\mathcal{O}(n)$ by encrypting each couple $(m_{i,0}, m_{i,1})$ with a different couple of keys $(k_{i,0}, k_{i,1})$ t-bit long, and running the OT with the keys as inputs instead of the messages. The total length of the offline communication to transmit the encrypted M'_j from $\mathcal{S}$ to $\mathcal{C}$ is nmt bits. In the online phase $\mathcal{C}$ transmits PK_0 (2T bits long) to $\mathcal{S}$, while $\mathcal{S}$ sends a message of length $nt + 2mt$ bits to $\mathcal{C}$ (the keys K_j can be as long as $k_{i,b}$).

To sum up, the offline communication overhead is 2^m keys per transfer, while the online communication overhead is $2^m/m$ keys per transfer. The complexities of the protocol for each single 1-out-2 OT are shown in Table 4.5.

Rounds	**Offline Bandwidth**	**Online Bandwidth**	# `exp`(Online)
2	$2^m t$	$\sim 2^m t/m$	$1/m$

Table 4.5: Complexity of a single 1-out-2 OT performed in a parallel OT.

4.4 Extending OT efficiently

In this section we show how an high number of OT, maybe having long input messages, can be implemented from a fixed number of OT with relatively short input messages.

In the sequel we use the notation OT_t^n to indicate a parallel OT that performs n 1-out-2 OT with messages of length t bits. In this section we describe the protocol proposed by Ishai et al. [IKNP03] that reduces the bandwidth necessary for an OT_t^n. The security of this protocol can be proven as long as the receiver $\mathcal{C}$ is semi-honest, in [IKNP03] there is also an extension of the protocol to handle malicious receivers.

The authors provide two protocols. The first one, given a security parameter k (i.e. the bitlength of exchanged secrets) known by both parties, permits to implement an OT_t^n by an OT_k^n and an additional message. The second protocol permits to implement an OT_t^n from an OT_n^k. By combining them it is possible to reduce an OT_t^n to an OT_k^k.

The first protocol, shown in Protocol 19, efficiently reduces the oblivious transfer of long messages into the oblivious transfer of short messages. It is based on an OT_k^n where $\mathcal{C}$ inputs his selection bits b_i, while $\mathcal{S}$ inputs random short messages $(x_{i,0}, x_{i,1})$. The original messages $(m_{i,0}, m_{i,1})$ are encrypted by using a Pseudo-Random Generator $PRG()$[1], where $x_{i,0}$ and $x_{i,1}$ are used as

---

[1] Note that $PRG()$ can be implemented by a hash function.

REDUCING OT^n_t TO OT^n_k

inputs of $\mathcal{C}$: $b = (b_0, \dots, b_{n-1})$, with $b_i \in \{0,1\}$, $i = 0, \dots, n-1$
inputs of $\mathcal{S}$: n pairs $(m_{i,0}, m_{i,1})$, $m_{i,b} \in \{0,1\}^t$, $i = 0, \dots, n-1$
output for $\mathcal{C}$: m_{i,b_i}, $i = 0, \dots, n-1$
output for $\mathcal{S}$: Nothing

Offline phase:

client $\mathcal{C}$ server $\mathcal{S}$

choose $x_{i,0}, x_{i,1} \in_R \{0,1\}^k$;

Online phase:

client $\mathcal{C}$ server $\mathcal{S}$

$y_{i,j} = m_{i,j} \oplus PRG(x_{i,j})\ \forall i = 0, \dots, n-1,\ j \in \{0,1\}$;

$$b_i \rightarrow \boxed{\quad OT^n_k \quad} \leftarrow x_{i,0}, x_{i,1}$$
$$x_{i,b_i} \leftarrow \qquad\qquad i = 0, \dots, n-1;$$

$$\xleftarrow{(y_{i,0}, y_{i,1}), \forall i = 0, \dots, n-1;}$$

$m_{i,b_i} = y_{i,b_i} \oplus PRG(x_{i,b_i})\ \forall i = 0, \dots, n-1.$

Protocol 19: Protocol that reduces OT^n_t to OT^n_k

seed, and appended to the last transmission of the OT protocol. Received all the encrypted messages and the seeds x_{i,b_i}, $\mathcal{C}$ is able to decrypt the messages m_{i,b_i}. Considering the OT^n_k implemented by n OT^1_k of Protocol 14, the communication complexity is reduced to $6nk$ bits transmitted during the OT and an additional transmission of $2nt$ bits, resulting in $6nk + 2nt$ bits exchanged in 2 rounds.

The second protocol, shown in Protocol 20, permits to evaluate an OT^n_ℓ from an OT^k_n, i.e., instead of evaluating n OT of long messages, it is possible to evaluate a fixed number of OT with messages n bits long. $\mathcal{C}$ generates a random matrix T and $\mathcal{S}$ generates a random selector vector s and they use them in an OT^k_n protocol where $\mathcal{C}$ acts as sender and $\mathcal{S}$ as receiver. At the end of the OT protocol $\mathcal{S}$ obtains a matrix Q, whose column q^j is t^j or $t_j \oplus b^\top$, according to the selector bit b_j, i.e. $q^j = (s_j b^\top) \oplus t^j$. Note that $q_i = (b_i s) \oplus t_i$. In an additional round $\mathcal{S}$ sends $y_{i,0}$ and $y_{i,1}$ to $\mathcal{C}$, where $y_{i,0} = m_{i,0} \oplus H(i, q_i)$ and $y_{i,1} = m_{i,1} \oplus H(i, q_i \oplus s)$. Finally $\mathcal{C}$ computes m_{i,b_i} as $y_{i,b_i} \oplus H(i, t_i)$, in

REDUCING OT_t^n TO OT_n^k

inputs of $\mathcal{C}$: $b = (b_0, \ldots, b_{n-1})$, with $b_i \in \{0,1\},\ i = 0, \ldots, n-1$
inputs of $\mathcal{S}$: n pairs $(m_{i,0}, m_{i,1}),\ m_{i,b} \in \{0,1\}^t,\ i = 0, \ldots, n-1$
output for $\mathcal{C}$: $m_{i,b_i},\ i = 0, \ldots, n-1$
output for $\mathcal{S}$: Nothing

Offline phase:

client $\mathcal{C}$ | server $\mathcal{S}$

choose a random bit matrix $T \in_R \{0,1\}^{n \times k}$; | choose a selector vector $s \in_R \{0,1\}^k$;

Online phase:

client $\mathcal{C}$ | server $\mathcal{S}$

$$t^j, t^j \oplus b^\top \rightarrow \boxed{\quad OT_n^k \quad} \begin{matrix} \leftarrow s_j \\ \rightarrow q^j,\ j = 0, \ldots, k-1; \end{matrix}$$

```
                                        for $i = 0, \ldots, n-1$
                                            $y_{i,0} = m_{i,0} \oplus H(i, q_i)$;
                                            $y_{i,1} = m_{i,1} \oplus H(i, q_i \oplus s)$;
                                        endfor
```

$$\xleftarrow{(y_{i,0}, y_{i,1}), \forall i = 0, \ldots, n-1;}$$

```
for $i = 0, \ldots, n-1$
    $m_{i,b_i} = y_{i,b_i} \oplus H(i, t_i)$;
endfor.
```

Protocol 20: Protocol that reduces OT_ℓ^n to OT_n^k (Given the matrix T, we here indicate the j-th column by t^j and the i-th row by t_i, the same for matrix Q).

fact

$$m_{i,b_i} = \begin{cases} y_{i,0} \oplus H(i, t_i) = m_{i,0} \oplus H(i, q_i) \oplus H(i, t_i) = \\ \qquad = m_{i,0} \oplus H(i, (0 \cdot s) \oplus t_i) \oplus H(i, t_i) & b_i = 0 \\ y_{i,1} \oplus H(i, t_i) = m_{i,1} \oplus H(i, q_i \oplus s) \oplus H(i, t_i) = \\ \qquad = m_{i,1} \oplus H(i, (1 \cdot s) \oplus t_i \oplus s) \oplus H(i, t_i) & b_i = 1. \end{cases} \tag{4.2}$$

The communication complexity of the protocol results in the complexity of the OT_n^k ($6kn$ bits) and an additional message ($2nt$ bits) transmitted after the OT in another round.

Reducing the bandwidth necessary for a large number of OT (maybe with long messages) is possible by combining the two protocols: first of all the OT_t^n is reduced to an OT_n^k and then the latter is reduced an OT_k^k. The communication complexity of the protocol is given by the complexity of the OT_k^k and the two additional messages exchanged to pass from the OT_k^k to the OT_n^k and finally to the OT_t^n. From the total communication complexity shown in Table 4.6, we can easily observe that the protocol is advantageous when $6k^2 + 2n(k+t) < 6n\ell$.

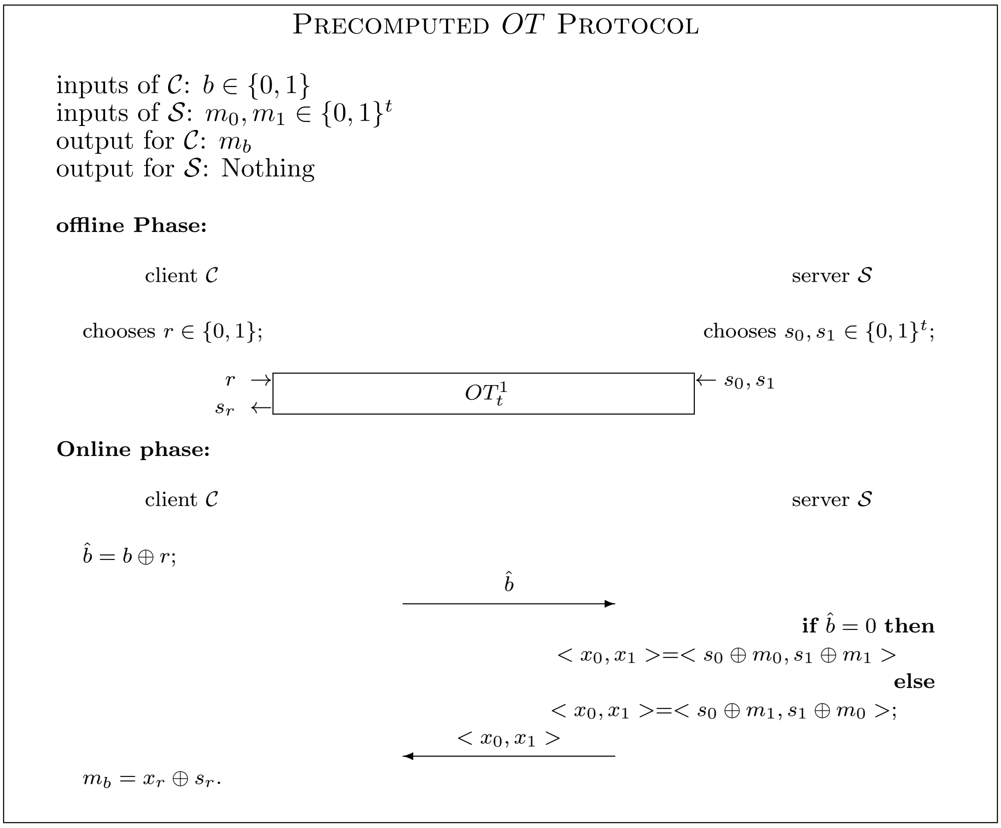
Rounds	**Online Bandwidth**
3	$6k^2 + 2kn + 2nt = 6k^2 + 2n(k+t)$

Table 4.6: Communication complexity of a parallel OT_t^n implemented by extending an OT_k^k.

4.5 Precomputing OT

To conclude the chapter we show how the OT can be moved to the offline phase, then performing simple operation in the online phase. This is useful when the parties involved know that they have to perform a secure protocol in the future. Finally we show that combining precomputation with the protocol of Section 4.4, the offline OT can be adapted to any OT_t^n.

In Protocol 21 the 1-out-2 OT protocol of [Bea95] with setup phase is proposed. All computationally expensive operations for OT are shifted into a setup phase by pre-computing OT. In the setup phase the parallel OT protocol is run on randomly chosen values: $\mathcal{S}$ submits the messages s_0 and s_1 (t bits long) and $\mathcal{C}$ chooses according to the bit r. Then, in the online phase, $\mathcal{C}$ uses its randomly chosen value r to mask his private inputs b and obtaining the bit $\hat{b}$, then $\mathcal{S}$ sends $\hat{b}$ to $\mathcal{S}$ that obfuscates his private inputs m_i ($i \in \{0,1\}$) by using his random values s_i from the setup phase, according to $\hat{b}$, i.e. $x_i = s_i \oplus m_i$ if $\hat{b} = 0$, otherwise $x_i = s_i \oplus m_{1-i}$. Finally, $\mathcal{S}$ sends x_0 and x_1 to $\mathcal{C}$ that uses the masks s_i he received from the OT protocol in the setup phase to remove the obfuscation and obtain the correct output value m_b. In addition to the cost of the OT in the setup phase, the online phase consists of two messages of size 1 bits and $2t$ bits and negligible computation (XOR



PRECOMPUTED OT PROTOCOL

inputs of $\mathcal{C}$: $b \in \{0,1\}$
inputs of $\mathcal{S}$: $m_0, m_1 \in \{0,1\}^t$
output for $\mathcal{C}$: m_b
output for $\mathcal{S}$: Nothing

offline Phase:

client $\mathcal{C}$ server $\mathcal{S}$

chooses $r \in \{0,1\}$; chooses $s_0, s_1 \in \{0,1\}^t$;

$r \rightarrow$ $\boxed{OT_t^1}$ $\leftarrow s_0, s_1$
$s_r \leftarrow$

Online phase:

client $\mathcal{C}$ server $\mathcal{S}$

$\hat{b} = b \oplus r$;

$\xrightarrow{\hat{b}}$

if $\hat{b} = 0$ **then**
$< x_0, x_1 >=< s_0 \oplus m_0, s_1 \oplus m_1 >$
else
$< x_0, x_1 >=< s_0 \oplus m_1, s_1 \oplus m_0 >$;

$\xleftarrow{< x_0, x_1 >}$

$m_b = x_r \oplus s_r$.

Protocol 21: Precomputed OT protocol with setup phase

of bitstrings).

The protocol can be easily extended to the parallel OT_t^n obtaining the complexity shown in Table 4.7. Moreover the offline complexity can be reduced as shown in Section 4.3 and Section 4.4.

Online Rounds	Offline Bandwidth	Online Bandwidth
2	$6nt$	$n(2t+1) \approx 2nt$

Table 4.7: Communication complexity of a precomputed OT_t^n.

Chapter 5

Garbled Circuits

Yao's Garbled circuits are a powerful tool that permits the evaluation in the encrypted domain of any functionality represented by an acyclic boolean circuit. We present the tool in Section 5.1 followed by significant implementation tricks in Section 5.2. Finally in Section 5.3 we present the implementation of useful subprotocols such as sum, product, comparison and, most of all, counter and logarithm, introduced for the first time in this thesis.

5.1 Garbled Circuits description

As shown in the seminal work by Yao [Yao82], any polynomial size functionality can be evaluated securely in a constant number of rounds and polynomial communication and computation overhead. The adopted technique is based on so called "garbled circuits".While Yao's protocol has long been thought to be of theoretical interest only, recent works have shown its efficiency [LPS08, KS08a, PSSW09] and usability by compilers for automatic generation of GC-based STPC protocols [MNPS04, PSS09].

Yao's Garbled Circuit approach [Yao86] is the most efficient method for the secure evaluation of a boolean circuit C. To describe garbled circuits in a few words, we can assert that Yao's idea is to encrypt (or garble) the nodes and the transitions of a boolean circuit such that the one who evaluates it may follow only a single evaluation path, defined by the circuit and the input attribute vector. Given a public function $y = f(\mathbf{x}_{\mathcal{C}}, \mathbf{x}_{\mathcal{S}})$, where $\mathbf{x}_{\mathcal{C}}$ is the array of $\mathcal{C}$'s private inputs and $\mathbf{x}_{\mathcal{S}}$ is the array of $\mathcal{S}$'s private inputs, it is possible to represent it by a boolean circuit C. $\mathcal{C}$ and $\mathcal{S}$ are interested to evaluate the circuit, without disclosing their inputs each other. At the end of the protocol the output will be available to $\mathcal{C}$ and optionally to $\mathcal{S}$.

The circuit, together with $\mathbf{x}_{\mathcal{C}}$ and $\mathbf{x}_{\mathcal{S}}$, is an input of a generic GC scheme, where one party ($\mathcal{S}$) constructs the circuit, then discloses the secrets necessary for the evaluation to the other party ($\mathcal{C}$) by using an Oblivious Transfer (OT) protocol and $\mathcal{C}$ uses them to evaluate the circuit. The general scheme, shown in Figure 5.1, can be associated to any function implementation and its three sections are described in detail in Section 5.1.1, Section 5.1.2 and Section 5.1.3.

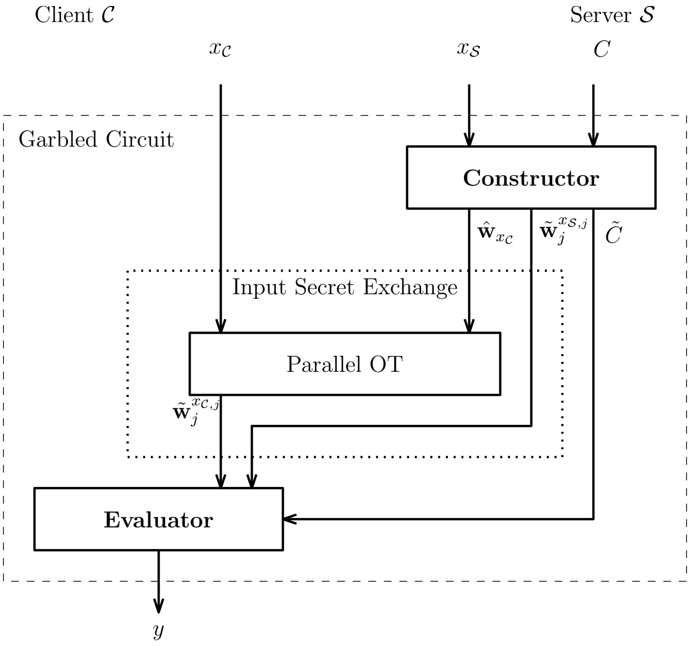


Figure 5.1: General scheme for Garbled Circuits.

A detailed proof of the security of Yao's garbled circuit protocol for semi-honest parties is provided in [LP09].

5.1.1 Contructor

The circuit **constructor** (server $\mathcal{S}$), creates a *garbled circuit* $\tilde{C}$:

- for each input, intermediate and output wire W_i of the circuit, the constructor randomly chooses a complementary garbled value $\widehat{w}_i = \langle \tilde{w}_i^0, \tilde{w}_i^1 \rangle$ consisting of two secrets, $\tilde{w}_i^0$ and $\tilde{w}_i^1$, where $\tilde{w}_i^j$ is the garbled value of W_i's value j, i.e. $\tilde{w}_i^j$ is a randomly chosen secret associated to j that does not reveal j;

- for each gate G_i, $\mathcal{S}$ creates and sends to the **evaluator** (client $\mathcal{C}$) a *garbled table* $\tilde{T}_i$ with the following property: given a set of garbled values of G_i's inputs, $\tilde{T}_i$ allows to recover the garbled value of the corresponding G_i's output, and nothing else.

Each secret is randomly chosen and is uniformly distributed in $(0, 2^t)$ (normally $t = 80$). Once the secrets are generated, for each gate, given the secrets $\widehat{w}_i$ and $\widehat{w}_j$ associated to the gate inputs wires and the secret $\widehat{w}_o$ associated to the gate output wire, the corresponding $\tilde{T}$ is generated in the following way:

$$\begin{bmatrix} \mathcal{E}nc_{\tilde{w}_i^0,\tilde{w}_j^0}(\tilde{w}_o^{g(0,0)}) \\ \mathcal{E}nc_{\tilde{w}_i^0,\tilde{w}_j^1}(\tilde{w}_o^{g(0,1)}) \\ \mathcal{E}nc_{\tilde{w}_i^1,\tilde{w}_j^0}(\tilde{w}_o^{g(1,0)}) \\ \mathcal{E}nc_{\tilde{w}_i^1,\tilde{w}_j^1}(\tilde{w}_o^{g(1,1)}) \end{bmatrix} \tag{5.1}$$

where $g(b_i, b_j)$ is the output of the gate. As to encryption $\mathcal{E}nc$, it is possible to use any symmetric encryption scheme having the following properties:

1. *Elusive range*: an encryption under one key is in the range of an encryption with a different key with negligible probability;
2. *Efficiently verifiable range*: given a key, a user can efficiently verify that a ciphertext is in the range of that key (this is feasible, for example, by appending 0^n at the end of the output secret).

For example, when preparing the $\tilde{T}$ of an AND gate, we obtain:

$$\begin{bmatrix} \mathcal{E}nc_{\tilde{w}_i^0,\tilde{w}_j^0}(\tilde{w}_o^0) \\ \mathcal{E}nc_{\tilde{w}_i^0,\tilde{w}_j^1}(\tilde{w}_o^0) \\ \mathcal{E}nc_{\tilde{w}_i^1,\tilde{w}_j^0}(\tilde{w}_o^0) \\ \mathcal{E}nc_{\tilde{w}_i^1,\tilde{w}_j^1}(\tilde{w}_o^1) \end{bmatrix} \tag{5.2}$$

The row of the tables are randomly scrambled to avoid that the evaluator understands the input values by the position of the message decrypted successfully.

5.1.2 Data exchange

Garbled values corresponding to $\mathcal{C}$'s inputs x_j are (obliviously) transferred to $\mathcal{C}$ with a parallel oblivious transfer protocol: $\mathcal{S}$ inputs complementary garbled values $\tilde{w}_j$ into the protocol; $\mathcal{C}$ inputs $x_{\mathcal{C},j}$ and obtains $\tilde{w}_j^{x_{\mathcal{C},j}}$. Oblivious Transfer can be instantiated efficiently with the protocols in [NP01, AIR01] which can be implemented using elliptic curve cryptography, As shown in [Bea95], the OTs can be pre-computed in a setup phase, such that they are not the performance bottleneck in Yao's protocol. Additionally, the number of computationally expensive public-key operations in the setup phase can be reduced to a constant number with the extensions proposed in [IKNP03].

Sequentially to the OT, $\mathcal{S}$ transmits the secrets $\tilde{w}_j^{x_{\mathcal{S},j}}$ relative to its input $x_{\mathcal{S},j}$ and the tables $\tilde{T}_i$ of the circuit.

5.1.3 Evaluator

$\mathcal{C}$ can evaluate the garbled circuit $\tilde{C}$ to obtain the garbled output simply by evaluating the garbled circuit gate by gate, using the garbled tables $\tilde{T}_i$. In each table the evaluator decrypts each row by using the input secrets previously obtained until it successfully performs a decryption. In the first gates only input secrets are used, while successively input secrets and/or secrets obtained as output from other gates are used. Finally, $\mathcal{C}$ determines the plain values corresponding to the obtained garbled output values using an output translation table received by $\mathcal{S}$. If the output is needed by $\mathcal{S}$, $\mathcal{C}$ transmits the garbled output.

5.2 Implementation details

Point and Permute To avoid that $\mathcal{C}$ performs 4 decryptions, the point and permute technique proposed in [MNPS04] can be used. Together with the secrets relative to each wire i, a permutation bit π_i is randomly chosen and associated to the value 0 so that $\pi_i^0 = \pi_i$ and $\pi_i^1 = 1 - \pi_i$.Note that the permutation bit reveals nothing about the wire value. During the construction of a table relative to a gate, in each row the constructor computes $\mathcal{E}nc_{\tilde{w}_i^{v_i},\tilde{w}_j^{v_j}}(\tilde{w}_o^{g(v_i,v_j)}||\pi_o^{g(v_i,v_j)})$, encrypting the permutation bit associated to the output together with the output secret, and finally permutes the table rows according to the permutation bits. The evaluator receives the permutation bits together with the input secrets and can use them to directly decrypt the correct row, obtaining new wire secrets and permutation bits that are used in the next tables. In this way the evaluator performs 1 decryption instead of 4 and the bits for the efficiently verifiable range property are no more necessary.

High-Speed Evaluation of GC In [MNPS04], the authors describe a protocol, named "Fairplay", that for performance reasons uses a cryptographic hash function $H(\cdot)$ (chosen from SHA-2 family) instead of the symmetric encryption $\mathcal{E}nc$ used in Yao's protocol, so that the encryption is carried out by replacing the symmetric encryption function with $\mathcal{E}nc_{\tilde{w}_i^{v_i},\tilde{w}_j^{v_j}}(\tilde{w}_o^{g(v_i,v_j)}||\pi_o^{g(v_i,v_j)}) = (\tilde{w}_o^{g(v_i,v_j)}||\pi_o^{g(v_i,v_j)}) \oplus H(\tilde{w}_i^{v_i}||\tilde{w}_j^{v_j}||s)$, where s is a unique identifier used once. Hence, creation of the garbled table of a non-XOR d-input gate requires 2^d invocations of H and its evaluation needs one invocation.

Improved Garbled Circuit with free XOR Another improvement to GCs is proposed in [KS08a], which allows "free" evaluation of XOR gates. More specifically, a garbled XOR gate has no garbled table and its evaluation consists of XOR-ing its garbled input values. This result in (*no communication*) and (*negligible computation*). The other gates, called *non-*XOR *gates*, are evaluated as described above.

The main idea of [KS08a] is, that the constructor $\mathcal{S}$ chooses a global key difference $\Delta \in_R \{0,1\}^t$ which is not revealed to the evaluator $\mathcal{C}$ and relates the garbled values associated to the wires as $\tilde{w}_i^0 = \tilde{w}_i^1 \oplus \Delta$. The usage of such garbled values allows for free evaluation of XOR gates with input wires W_1, W_2 and output wire W_3 by computing $\tilde{w}_3 = \tilde{w}_1 \oplus \tilde{w}_2$ (no communication and negligible computation). However, using related garbled values requires that the hash function $H(\cdot)$ used to create the garbled tables of non-XOR gates has to be modelled to be correlation robust (as defined in [IKNP03]). In practise, H can be chosen from the SHA-2 family.

5.3 Building blocks for GC: general overview

In the following we present the circuits describing several frequently used primitives, whose equivalent homomorphic solution, if available, has already been presented in Section 3.4. Differently from the circuits available in VLSI toolboxes and hardware compilers, the circuits here presented are designed in order to minimize the number of non-XOR gates.

Given a desired function and a circuit describing it, having n non-XOR gates, $\mathcal{C}$ inputs represented by $\ell_{\mathcal{C}}$ bits and $\mathcal{S}$ inputs represented by $\ell_{\mathcal{S}}$ bits, and considering that each secret-permutation bit pair is represented by t bits (usually t=80), its complexity can be computed by considering that:

- any protocol needs only the 2 rounds necessary for the Oblivious Transfer protocol, the garbled tables and the garbled secrets are appended to the last OT messages;

- the total amount of data transferred depends on the number of tables transmitted that represent the circuit ($4t \cdot n$ bits), the $\mathcal{S}$'s secrets transmission ($\ell_{\mathcal{S}} \cdot t$ bits), and the OT performed for transmitting the secrets relative to $\mathcal{C}$'s inputs. If precomputation is allowed, an offline OT is performed on random inputs (if $\ell_{\mathcal{C}} < 3t$ the bandwidth is $\approx 6\ell_C t$, otherwise, adopting all extensions provided in Chapter 4, it can be reduced to an OT_t^t having bandwidth $\approx 6t^2 + 4\ell_C t$) and then, during the online phase, the online phase of the OT is performed ($2\ell_{\mathcal{C}} \cdot t$ bits as in Protocol 21). Sometimes even circuit garbling and transmission can be performed offline;

- the computational complexity on $\mathcal{S}$'s side relies on the generation of the secrets (negligible) and the tables creation ($4n$ hash functions). Sometimes this operation can be performed offline;

- the computational complexity on $\mathcal{C}$'s side relies on the online computation of a hash function for each non-XOR gate (n hash functions) and XOR functions (negligible).

Provided the description of a generic circuit, its total complexity is summarised in Table 5.1.

Offline	**Circuit**		**Online**		
Bandwidth	# Hash ($\mathcal{S}$)	Bandwidth	Rounds	Bandwidth	# Hash ($\mathcal{C}$)
$\approx \min\{6t^2 + 4\ell_{\mathcal{C}}t \, 6\ell_{\mathcal{C}}t\}$	$4n$	$4nt$	2	$\ell_{\mathcal{S}}t + 2\ell_{\mathcal{C}}t$	n

Table 5.1: General average GC complexities (Circuit complexity owns to offline or online phase according to the specific scenario)

In these sections we present improved circuit constructions that, differently from the circuits available in VSLI toolboxes and hardware compilers, are designed to minimize the number of non-XOR gates. Several frequently used primitives, such as integer addition, subtraction, multiplication, comparison, etc., are presented. For each circuit its complexity can be derived according to Table 5.1, hence in the further we only underline the number of non-XOR gates composing the circuit.

5.3.1 Addition (ADD)

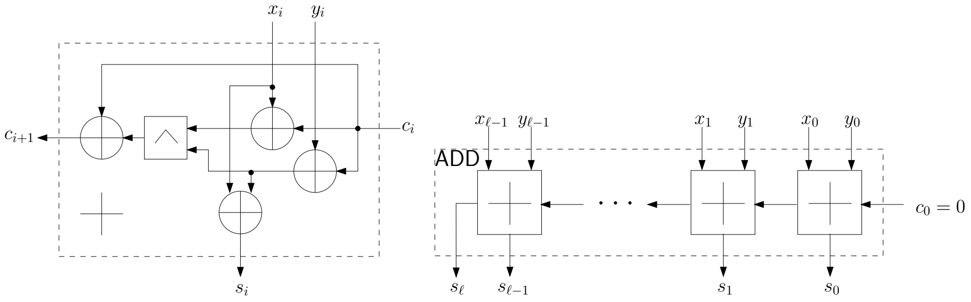
Addition circuits (ADD) [KSS09b] that add two unsigned integer values $x^{(\ell)}$, $y^{(\ell)}$, represented with ℓ bits, can be efficiently obtained as a chain of 1-bit adders $(+)$, often called full-adders, as shown in Figure 5.2 (b). (The first 1-bit adder has constant input $c_0 = 0$ and can be replaced by a smaller half-adder). Each 1-bit adder has as inputs the carry-in bit c_i from the previous 1-bit adder and the two input bits x_i, y_i. The outputs are the carry-out bit $c_{i+1} = (x_i \wedge y_i) \vee (x_i \wedge c_i) \vee (y_i \wedge c_i)$ and the sum bit $s_i = x_i \oplus y_i \oplus c_i$ (the latter can be computed "for free" [KS08a]).

The construction of a 1-bit adder shown in Figure 5.2(a) computes the carry-out bit as $c_{i+1} = c_i \oplus ((x_i \oplus c_i) \wedge (y_i \oplus c_i))$. The construction for a 1-bit adder consists of four free XOR gates and a single 2-input AND gate. The overall size of the addition circuit is $|\mathsf{ADD}^{\ell}| = \ell \cdot |+| = \ell$ non-XOR gates and the output is represented by $\ell + 1$ bits.

5.3.2 Sum of k values

Let us suppose that we want to add k values x_i, each represented with ℓ bits.

The ADD circuits have inputs whose size increases during the computation: if a reverse tree structure with depth $p = \lceil \log_2 k \rceil$ is used, the inputs length



(a) 1-bit Adder (+) (b) Addition Circuit (ADD)

Figure 5.2: Addition Circuit: 1-bit Adder (a) and ADD (b).

of each adder is $\ell + i - 1$ bits, where $i \in [1 \ldots p]$ indicates the level position in the tree. For simplicity we consider that $k = 2^p$.

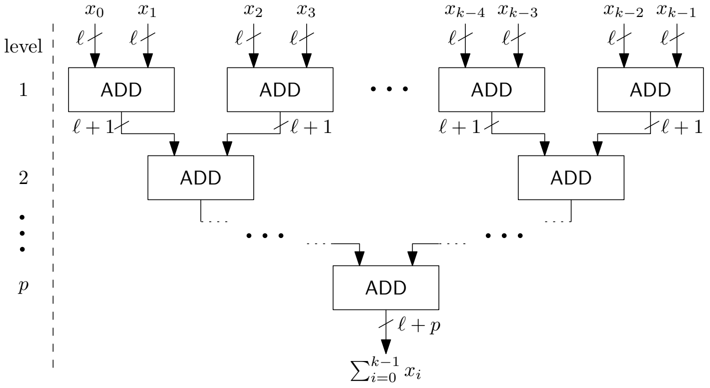

level
1
2
p
x_0 x_1 x_2 x_3 x_{k-4} x_{k-3} x_{k-2} x_{k-1}
ℓ
ADD
$\ell+1$
$\ell+p$
$\sum_{i=0}^{k-1} x_i$

Figure 5.3: Reverse tree structure for the computation of n additions.

For each level i of the tree, there are exactly $k/2^i$ ADD circuits. The total number of non-XOR gates is

$$\begin{aligned} & \sum_{i=1}^{p} k/2^i \ (\ell + i - 1) \\ = & \left[k(\ell-1) \sum_{i=1}^{p} \left(\frac{1}{2}\right)^i \right] + \left[k \sum_{i=1}^{p} i \left(\frac{1}{2}\right)^i \right]. \end{aligned} \tag{5.3}$$

We need to solve two series: $\sum_{i=1}^{p}(1/2)^i$ and $\sum_{i=1}^{p} i(1/2)^i$. Recalling that

$2^p = k$,

$$\sum_{i=1}^{p}\left(\frac{1}{2}\right)^i = \sum_{i=0}^{p}\left(\frac{1}{2}\right)^i - 1 = \frac{1-(1/2)^{p+1}}{1-1/2} - 1$$
$$= 2(1 - \frac{1}{2\cdot 2^p}) - 1 = 1 - \frac{1}{k} = \frac{k-1}{k}. \quad (5.4)$$

The second series is a bit more difficult. We consider that $\sum_{i=0}^{p} x^i = \frac{1-x^{p+1}}{1-x}$ $\forall x \in (0,1)$, hence

$$\begin{aligned}
\frac{\partial}{\partial x}\sum_{i=0}^{p} x^i &= \frac{\partial}{\partial x}\frac{1-x^{p+1}}{1-x};\\
\sum_{i=0}^{p} i x^{i-1} &= \frac{-(p+1)x^p(1-x)+1-x^{p+1}}{(1-x)^2};\\
\frac{1}{x}\sum_{i=1}^{p} i x^{i} &= \frac{-(p+1)(1-x)x^p+1-x\cdot x^{p}}{(1-x)^2};\\
\sum_{i=1}^{p} i x^{i} &= x\frac{1-x^p[(p+1)(1-x)+x]}{(1-x)^2}.
\end{aligned}$$

Considering $x = 1/2$ and recalling again that $2^p = k$ we obtain

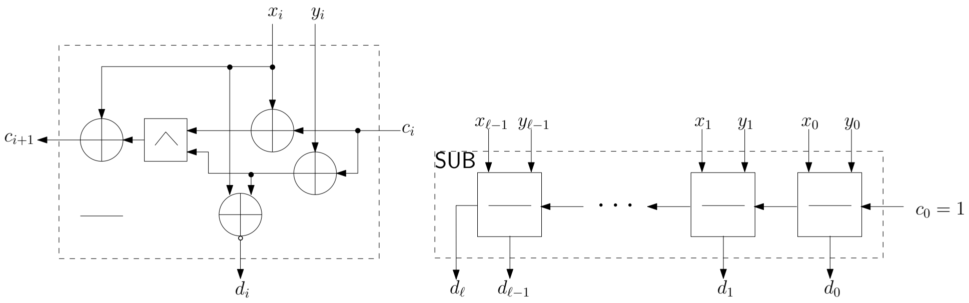
$$\begin{aligned}
\sum_{i=1}^{p} i\left(\frac{1}{2}\right)^i &= \frac{1}{2}\frac{1-1/2^p[(p+1)(1/2)+1/2]}{(1/2)^2}\\
&= 2\left(1-\frac{1}{k}\left(\frac{p}{2}+1\right)\right) = 2\frac{2k-p-2}{2k} = \frac{2k-p-2}{k}. \quad (5.5)
\end{aligned}$$

Substituting (5.4) and (5.5) in (5.3) and removing the constraint $2^p = k$, we obtain that the total number of non-XOR gates needed for the addition of k ℓ-bit values is $|k\ \mathsf{ADD}^{\ell}| = k(\ell-1)(k-1)/k + k(2k-p-2)/k = \ell(k-1)+k-p-1$. The output is represented with $\ell + \log_2 k$ bits.

5.3.3 Subtraction (SUB)

Subtraction in two's complement representation [KSS09b] is defined as $x^{(\ell)} - y^{(\ell)} = x^{(\ell)} + \bar{y}^{(\ell)} + 1$. Hence, a subtraction circuit (SUB) can be constructed analogously to the addition circuit from 1-bit subtractors ($-$) as shown in

Figure 5.4 (b). Each 1-bit subtractor computes the carry-out bit $c_{i+1} = (x_i \wedge \bar{y}_i) \vee (x_i \wedge c_i) \vee (\bar{y}_i \wedge c_i)$ and the difference bit $d_i = x_i \oplus \bar{y}_i \oplus c_i$. We instantiate the 1-bit subtractor efficiently as shown in Figure 5.4(a) to compute $c_{i+1} = x_i \oplus ((x_i \oplus c_i) \wedge (y_i \oplus c_i))$, hence the overall size of the circuit is $|\mathsf{SUB}^\ell| = \ell \cdot |-| = \ell$ non-XOR gates. The output of the circuit is represented by $\ell + 1$ bits where the most significant bit is the sign bit.



(a) 1-bit Subtractor ($-$) (b) Subtraction Circuit (SUB)

Figure 5.4: Subtraction Circuit: 1-bit Subtractor (a) and SUB(b).

5.3.4 Controlled Addition or Subtraction(ADDSUB)

A block ADDSUB [BFL+11] which can add/subtract the unsigned ℓ-bit value $y^{(\ell)}$ to/from the signed ℓ-bit value $x^{(\ell)}$ in two's complement depending on a control bit ctrl can be naturally constructed from ADD: If ctrl $= 1$, $y^{(\ell)}$ must be subtracted from $x^{(\ell)}$. This can be done by converting $y^{(\ell)}$ into two's complement and adding it to $x^{(\ell)}$. The resulting circuit is shown in Figure 5.5. As each of the bit-addition blocks ($+$) can be implemented with one non-XOR gate, this circuit has size $|\mathsf{ADDSUB}^\ell| = \ell$ non-XOR gates. Its output is $\ell + 1$ bits long.

5.3.5 Changing representation

Subtraction result is usually given in two's complement form, but sometimes a sign/modulus representation may be needed for further computation or vice-versa. To convert an $(\ell + 1)$-bit signed integer value $x^{(\ell+1)}$ from 2's complement to sign/magnitude representation [KSS09b], the least significant

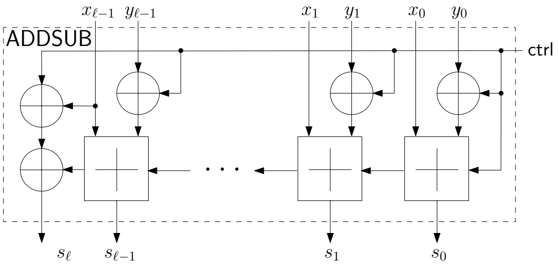


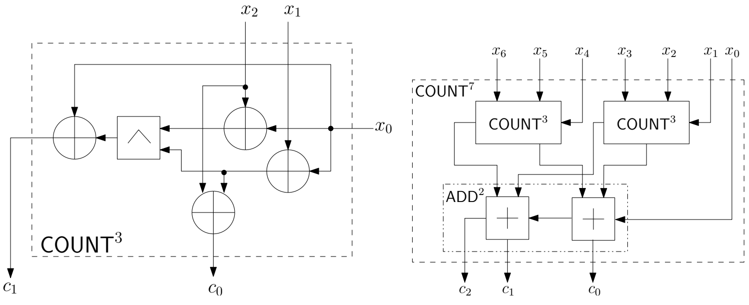
Figure 5.5: Controlled addition or subtraction circuit (ADDSUB).

ℓ bits of $x^{(\ell+1)}$ are added to or subtracted from 0 depending on the sign of $x^{(\ell+1)}$, i.e., the most significant bit of $x^{(\ell+1)}$, using an ADDSUB block of size ℓ non-XOR gates.

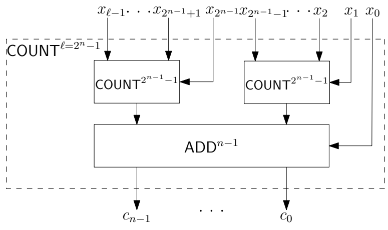
5.3.6 Counter (COUNT)

We now present a circuit that returns the number of non-null bits in a bitstring $x^{(\ell)}$. The simplest way to do this is by using a cascade of half bit adders, requiring $\ell - 1$ levels. In the first level, two bits are added and in the next levels an additional bit is added. Recalling that XOR gates are evaluated for free by the GCs, we see that this implementation is not very efficient. In fact each level needs about $\log_2(level)$ non-XOR gates. In [BGL10] a recursive construction is proposed to design a Boolean circuit that implements a counter and that can be implemented with less than ℓ non-XOR gates by using the 1-bit adder and the SUM proposed in Section 5.3.1. A schematic view of the construction is shown in Figure 5.6.

In Figure 5.6(a), the block COUNT_3 is obtained by using a 1-bit adder where the carry input is used for the third input, instead of setting it equal to 0. The COUNT^7 in Figure 5.6(b) is developed by using a COUNT^3 for the inputs $x_1 \dots x_3$, another COUNT^3 for the inputs $x_4 \dots x_6$ and an ADD^2 having as input the results of the two COUNT^3 and x_0 as input of the carry input. Recursively the COUNT^ℓ in Figure 5.6(c), where $\ell = 2^n - 1$, can be developed by using a $\mathsf{COUNT}^{2^{n-1}-1}$ for the inputs $x_1 \dots x_{2^{n-1}-1}$, another $\mathsf{COUNT}^{2^{n-1}-1}$ for the inputs $x_{2^{n-1}} \dots x_{\ell-1}$ and a ADD^{n-1} having the results of the two $\mathsf{COUNT}^{2^{n-1}-1}$ as input and x_0 in the carry input. This immediately implies



(a) COUNT^3 (b) COUNT^7



(c) COUNT$^\ell$ ($\ell = 2^n - 1$)

Figure 5.6: Counter Circuit: recursive construction.

a recursive construction for any input bitsize $\ell = 2^n - 1$. A generic COUNT$^{\ell'}$ where $\ell' \neq \ell - 1$ can be obtained by using a greater COUNT$^{\ell-1}$ having $\ell - 1 - \ell'$ null inputs and cutting the unused gates.

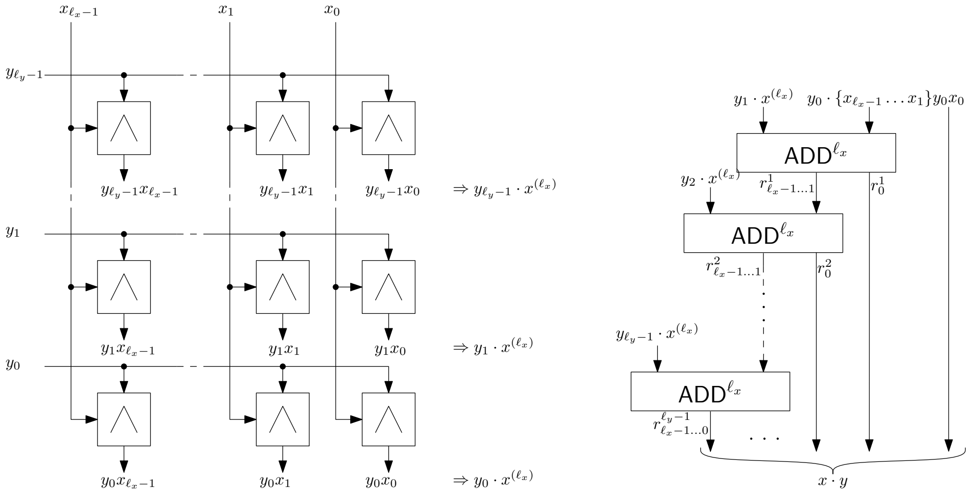
Table 5.2 shows the complexity of the COUNT as a function of the number of non-XOR gates present in the circuit. Given a generic input bitlength ℓ, the size of the COUNT circuit is $|\text{COUNT}^\ell| \lessapprox \ell - \log_2(\ell+1)$ and the output is represented with $\log_2 \ell$ bits. Note that the protocol allows to parallelize the computation by evaluating the sub-COUNT blocks in parallel.

Circuit	non-XOR gates
COUNT^3	1
COUNT^7	$2 + 2 = 4$
COUNT^{15}	$2 * 4 + 3 = 11$
COUNT$^{\ell=2^n-1}$	$\ell - n = \ell - \log_2(\ell+1)$

Table 5.2: Number of non-XOR gates for COUNT$^\ell$.

5.3.7 Product (MUL)

Multiplication circuits (MUL) that multiply two integers $x^{(\ell_x)}$, $y^{(\ell_y)}$, represented with ℓ_x and ℓ_y bits respectively, can be constructed according to the "school method" for multiplication, i.e., adding up the bitwise multiplications of y_i and $x^{(\ell_x)}$ shifted corresponding to the position: $x^{(\ell_x)} \cdot y^{(\ell)} = \sum_{i=0}^{\ell_y-1} 2^i (y_i \cdot x^{(\ell_x)})$. This circuit is composed by $\ell_x \ell_y$ 1-bit multipliers (2-input AND gates) (Figure 5.7(a)) and $(\ell_y - 1)$ adders of ℓ_x-bit each (Figure 5.7(b)).



(a) Bitwise ANDs (b) Shifts and Additions

Figure 5.7: Product circuit.

The size of the multiplication circuit is $|\mathsf{MUL}^{(\ell_x,\ell_y)}| = \ell_x \ell_y + (\ell_y - 1)\ell_x = 2\ell_x \ell_y - \ell_x$ non-XOR gates (note that it is convenient to set the longer input as $\mathbf{x}$). The output is $\ell_x + \ell_y$ bits long. If $\ell_x = \ell_y = \ell$ then the circuit size is $2\ell^2 - \ell$ non-XOR gates and the output is represented with 2ℓ bits.

5.3.8 Scalar Product and Matrix Product

Given the addition and multiplication circuits it is possible to define the scalar and matrix product.

We first consider the scalar product between two vectors $\mathbf{x} = [x_1^{(\ell_x)} \dots x_k^{(\ell_x)}]$

and $\mathbf{y} = [y_1^{(\ell_y)} \dots y_k^{(\ell_y)}]$. The circuit computing $\mathbf{x}\mathbf{y}^\top = \sum_{i=1}^{k} x_i^{(\ell_x)} y_i^{(\ell_y)}$ is composed by k MUL and $k-1$ ADD.

Each MUL has inputs represented with ℓ_x and ℓ_y bits and is composed by $2\ell_x\ell_y - \ell_x$ non-XOR gates. Totally $k(2\ell_x\ell_y - \ell_x)$ non-XOR gates are necessary for the products. The outputs of the products are added together as shown in Section 5.3.2.

Considering that the inputs to the n ADD circuits ares represented with $\ell_x + \ell_y$ bits, the total number of non-XOR gates needed for a scalar product is $\approx k(2\ell_x\ell_y - \ell_x) + (\ell_x + \ell_y)(k-1) + k - \log_2 k - 1$. We can observe that the output of the scalar product is a value represented by $\ell_x + \ell_y + \log_2 k$ bits.

The circuit that computes a matrix product between the (l, m) matrix

$$\mathbf{x} = \begin{bmatrix} x_{1,1}{}^{(\ell_x)} & \cdots & x_{1,m}^{(\ell_x)} \\ \vdots & \ddots & \vdots \\ x_{l,1}{}^{(\ell_x)} & \cdots & x_{l,m}^{(\ell_x)} \end{bmatrix}$$

and the (m, n) matrix

$$\mathbf{y} = \begin{bmatrix} y_{1,1}{}^{(\ell_y)} & \cdots & y_{1,n}{}^{(\ell_y)} \\ \vdots & \ddots & \vdots \\ y_{m,1}{}^{(\ell_y)} & \cdots & y_{m,n}{}^{(\ell_y)} \end{bmatrix},$$

is obtained by using $l \cdot n$ circuits computing the scalar product.

5.3.9 Linear Filtering

Implementing a FIR (Finite Impulse Response) filter by using GC is not as easy as with homomorphic encryption. We suppose to have to filter a sampled signal $\mathbf{x} = \{x_1^{(\ell_x)}, \dots, x_n^{(\ell_x)}\}$ by applying a filter having integer coefficients $c_0^{(\ell_c)}, \dots, c_k^{(\ell_c)}$ $(k \ll n)$. An encrypted filtered sample is computed as

$$y_i = \sum_{j=0}^{k} x_{i-j} c_j. \tag{5.6}$$

As in Section 3.4.3, we pay attention to the indexes, by limiting the computation to $y_k, \dots, y_n$ or by choosing a value for x_{i-j} $(x_{i-j} = x_0$ or $x_{i-j} = 0)$ when

$i - j < 0$. Considering that a filter having even real magnitude response, has symmetric coefficients, i.e. $c_j = c_{k-j}$ we can optimize the filter computation as

$$y_i = \sum_{j=0}^{k/2} (x_{i-j} + x_{i-k+j})c_j. \tag{5.7}$$

The circuit that filters each sample is composed by $k/2$ SUB, $k/2$ MUL and $k/2 - 1$ ADD. Each SUB has inputs with bitlength ℓ_x and output bitlength $\ell_x + 1$. The output of each MUL is $\ell_x + \ell_c + 1$ bit long. The sum of the product results can be computed by using a reverse tree structure of ADD circuits, as shown in Section 5.3.2, and the filtered samples are represented by $\ell_x + \ell_c + \lceil \log_2 k \rceil$ bits.

The total number of non-XOR gates is $\approx n[k/2(\ell_x) + k/2(2(\ell_x + 1)\ell_c - \ell_x - 1) + (\ell_x + \ell_c + 1)(k/2 - 1) + k/2 - \log_2 k - 2] = n[k\ell_x\ell_c + (1/2)k\ell_x + (3/2)k\ell_c - \ell_x - \ell_c + k/2 - \log_2 k - 3]$.

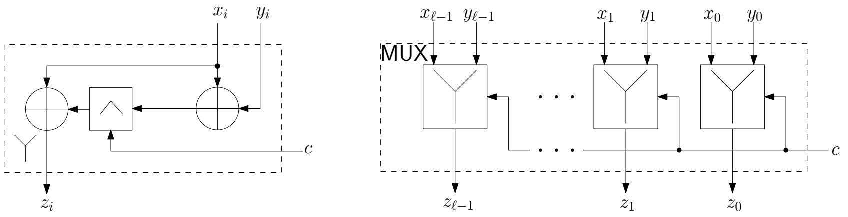
5.3.10 Multiplexer (MUX)

An ℓ-bit multiplexer circuits MUX selects its output $z^{(\ell)}$ to be one of the ℓ-bit inputs $x^{(\ell)}$ and $y^{(\ell)}$, depending on the selection bit c. MUX can be developed by using the circuit proposed in [KS08a], summarized in the following.

An ℓ-bit multiplexer circuit MUX selects its output $z^{(\ell)}$ to be its left ℓ-bit input $x^{(\ell)}$ if the input selection bit c is 0, respectively its right ℓ-bit input $y^{(\ell)}$ otherwise. As shown in Figure 5.8(b), the block can be obtained from ℓ parallel Y blocks that are 1-bit multiplexers. The Y gates have three inputs x_i, y_i, c and one output z_i. Y blocks can be instantiated efficiently as shown in Figure 5.8(a): this instantiation needs only a 2-input AND gate and two free XOR gates. Overall, the ℓ-bit multiplexer is composed by $|\mathsf{MUX}^{\ell}| = \ell$ non-XOR gates.

5.3.11 Square value (SQR)

The circuit computing the square of an ℓ bit integer number $x^{(\ell)}$ can be derived by the MUL circuit. The matrix of AND gates is symmetric, being $x_i x_j = x_j x_i$ and the elements on the diagonal are $x_i x_i = x_i$, hence only the computation of $(\ell - 1) + (\ell - 2) + \ldots + 1 = \frac{\ell(\ell-1)}{2}$ AND gates is necessary. The final result



(a) Y Block (b) Multiplexer Circuit

Figure 5.8: Y block (a) and Multiplexer circuit (b).

is obtained by $\ell-1$ ADDs each of ℓ bit values. The overall circuit is hence composed by $\frac{\ell(\ell-1)}{2}+(\ell-1)\ell = 3/2(\ell^2-\ell)$ non-XOR gates. The output is represented by 2ℓ bits.

A more efficient implementation can be obtained by observing that

$$(x^{(\ell)})^2 = (\sum_{i=0}^{\ell-1} x_i 2^i)^2 = \sum_{i=0}^{\ell-1}(x_i 2^i)^2 + 2\sum_{i=0}^{\ell-2}\sum_{j=i+1}^{\ell-1} x_i x_j 2^{i+j}.$$

Considering that $(x_i 2^i)^2 = x_i 2^{2i}$, we obtain $\sum_{i=0}^{\ell-1}(x_i 2^i)^2 = [x_{\ell-1}0\ldots x_2 0 x_1 0 x_0]$. $\sum_{i=0}^{\ell-2}\sum_{j=i+1}^{\ell-1} x_i x_j 2^{i+j}$ can be written as

$$\sum_{i=0}^{\ell-2} 2^i x_i \sum_{k=1}^{\ell-1-i} x_{i+k} 2^{i+k} = \sum_{i=0}^{\ell-2} 2^{2i+1} x_i \cdot [x_{\ell-1}\ldots x_{i+1}].$$

The circuit (Figure 5.9) is composed by $\frac{\ell(\ell-1)}{2}$ AND gates necessary for the computation of each $x_i x_j$, $\ell-2$ ADDs whose input bitlength changes from $\ell-2$ to 1 bits and the final adder, composed by $2\ell-2$ non-XOR gates, that sums the result obtained from the previous adder to the bitstring $[x_{\ell-1}0\ldots x_2 0 x_1 0 x_0]$. To construct the circuit it is necessary to pay particular attention to the shifts introduced by the 2^{2i+1} elements.

The total number of non-XOR gates necessary for this optimized solution is $|\mathsf{SQR}^\ell| = \frac{\ell(\ell-1)}{2}+\frac{(\ell-1)(\ell-2)}{2}+2\ell-2 = \ell^2-1$, saving $\ell^2/2-\ell+1$ non-XOR gates respect to the previous implementation.

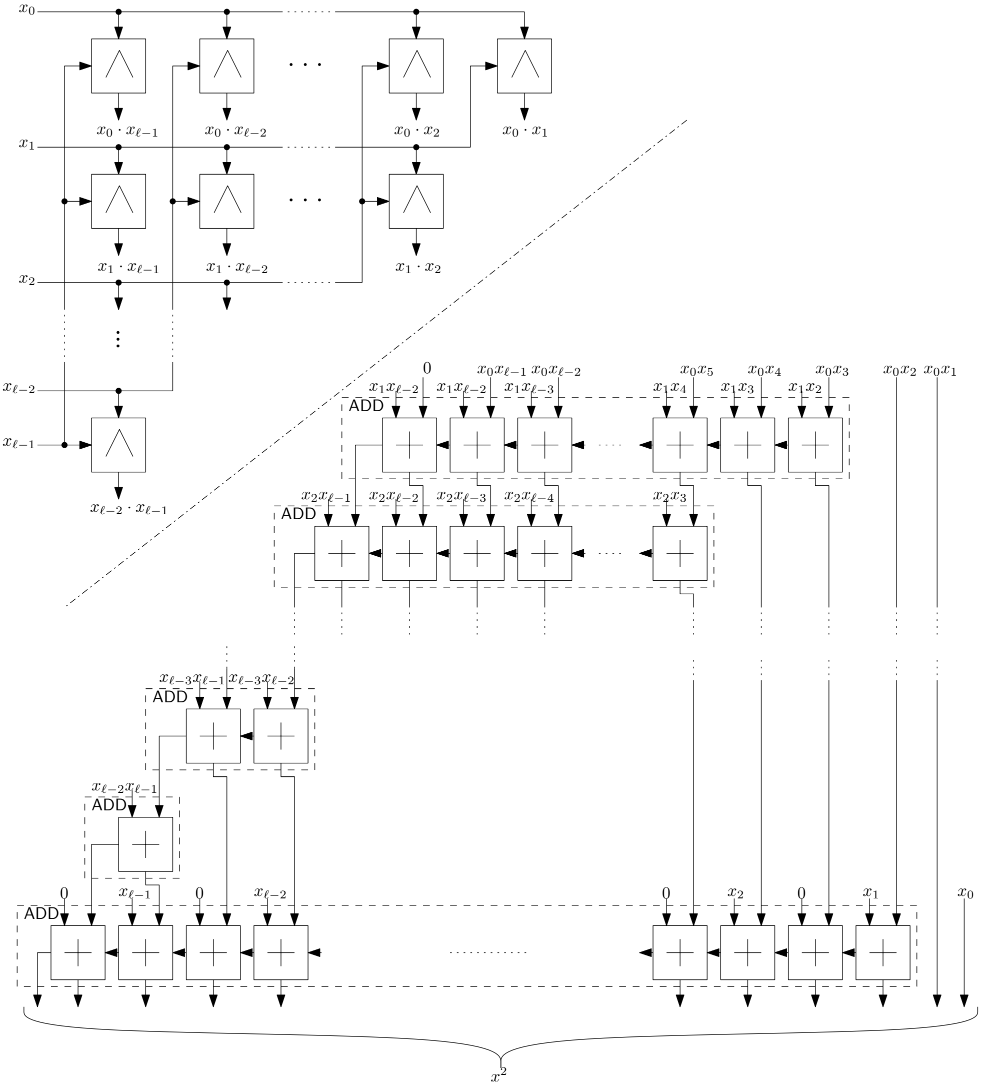


Figure 5.9: Square circuit.

5.3.12 Energy computation

Given the ℓ-bit samples ${s_i}^{(\ell)}$ of a signal, with $i = 1, \dots, k$, the energy $E_s = \sum_{i=1}^{k} s_i^2$ is computed by adding together the output of k SQR circuits. As shown in Section 5.3.2, the additions are performed by using a reverse tree structure, where the inputs of the first level are represented with 2ℓ bits. The total number of non-XOR gates of the circuit is $\approx k \cdot (\ell^2 - 1) + 2\ell(k-1) + k - \log_2 k - 1 = k\ell^2 + 2k\ell - 2\ell - \log_2 k - 1$.

5.3.13 Comparison

We distinguish between two comparisons: *equality* ($[x^{(\ell)} \overset{?}{=} y^{(\ell)}]$) and CMP (*greater than*: $[x^{(\ell)} \overset{?}{>} y^{(\ell)}]$). Other comparisons can be implemented by using the greater than circuit.

Equality

We start by observing that computing the bitwise XOR of two identical bitstrings, say $x^{(\ell)}$ and $y^{(\ell)}$ produces an all zero string.

Computing $z = \bigvee_{i=0}^{\ell-1} x_i \oplus y_i$, where $\vee$ denotes the OR gate, we obtain a bit z that assumes the value 0 if ($[x^{(\ell)} \overset{?}{=} y^{(\ell)}]$), 1 otherwise. The circuit implementing the equality test (Figure 5.10) is composed by $\ell - 1$ non-XOR gates and the output is represented by 1 bit.

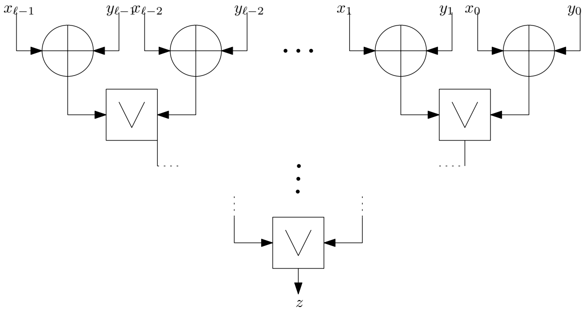


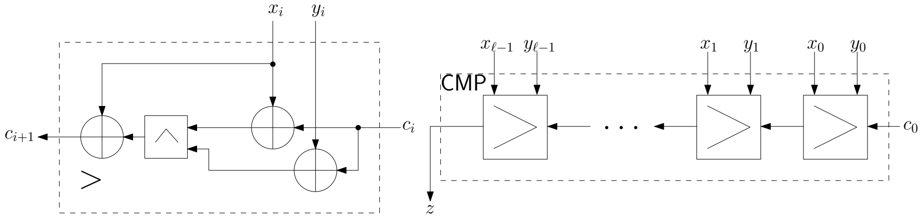
Figure 5.10: Equality test circuit ($\overset{?}{=}$).

Greater Than

We now present a circuit construction for the comparison of two ℓ-bit integers $x^{(\ell)}$ and $y^{(\ell)}$, i.e.,

$$z = \left[x^{(\ell)} > y^{(\ell)}\right] := \begin{cases} 1 & \text{if } x^{(\ell)} > y^{(\ell)}, \\ 0 & \text{otherwise.} \end{cases}$$

As shown in Figure 5.11(b), a comparison circuit (CMP) can be obtained starting from ℓ sequential 1-bit comparators ($>$). (The first 1-bit comparator has constant input $c_1 = 0$ and can be replaced by a smaller gate). A generic 1-bit comparator, shown in Figure 5.11(a), uses one 2-input AND gate and three free XOR gates. Note that this bit comparator is exactly the bit subtractor shown in Figure 5.4(a) restricted to the carry output: $[x^{(\ell)} \overset{?}{>} y^{(\ell)}] \Leftrightarrow [x^{(\ell)} - y^{(\ell)} - 1 \overset{?}{\geq} 0]$ which coincides with an underflow in the corresponding subtraction denoted by subtractor's most significant output bit $d_{\ell+1}$. The size of the comparison circuit is $|\mathsf{CMP}^{\ell}| = \ell$ non-XOR gates and the output is represented by 1 bit.



(a) $>$ block (b) Comparison ciruit

Figure 5.11: Comparison Circuit: $>$ block (a) and CMP (b). If $c_0 = 0$ the circuit evaluates $x > y$, otherwise it evaluates $x \geq y$.

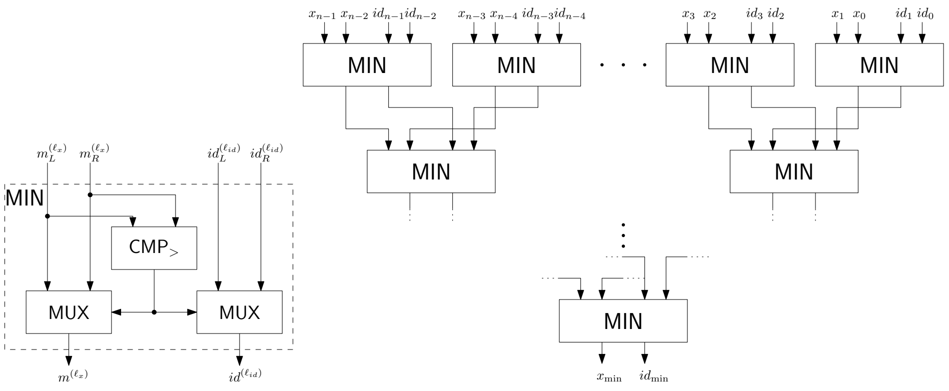
Comparison circuits for $[x^{(\ell)} \overset{?}{<} y^{(\ell)}]$, $[x^{(\ell)} \overset{?}{\geq} y^{(\ell)}]$, or $[x^{(\ell)} \overset{?}{\leq} y^{(\ell)}]$ can be obtained from the improved circuit for $[x^{(\ell)} \overset{?}{>} y^{(\ell)}]$ by interchanging $x^{(\ell)}$ with $y^{(\ell)}$ and/or setting the initial carry to $c_1 = 1$.

5.3.14 Minimum Value

The comparison and multiplexer blocks presented above can be combined to obtain a minimum circuit (MIN) which selects the minimum value of a list of values.

We suppose to have a list of n ℓ_x-bit values $x_i^{(\ell_x)}$, $i = 1 \ldots n$ each having an ℓ_d-bit identifier $id_i^{(\ell_{id})}$ associated. Without loss of generality we assume that n is a power of two. Our goal is to obtain both the minimum value $m^{(\ell_x)} = x_{min}^{(\ell_x)} = \min_i\{x_i^{(\ell_x)}\}$ and the identifier $id_{min}^{(\ell_{id})}$ associated to it.

Performance of MIN mainly comes from the building blocks used for integer comparison. More specifically, the minimum value and minimum identifier are selected pairwise in a tournament-like way using a reverse tree of minimum blocks ($\mathsf{MIN}s$) as shown in Figure 5.12(b). As shown in Figure 5.12(a), each minimum block gets as inputs the minimum ℓ-bit values $m_L^{(\ell_x)}$ and $m_R^{(\ell_x)}$ of its left and right subtrees T_L, T_R together to their identifiers $id_L^{(\ell_{id})}$ and $id_R^{(\ell_{id})}$, and outputs the minimum ℓ_x-bit value $m^{(\ell_x)}$ and ℓ_{id}-bit minimum identifier $id^{(\ell_{id})}$ of the tree. First, the two minimum values are compared with a comparison circuit (Section 5.3.13). If the minimum value of T_L is larger than that of T_R (in this case, the comparison circuit outputs the value 1), $m^{(\ell_x)}$ is chosen to be the value of T_R with an ℓ_x-bit multiplexer block (Section 5.3.10). Similarly, the minimum identifier $id^{(\ell_{id})}$ is chosen to be $id_R^{(\ell_{id})}$ with an ℓ_{id}-bit multiplexer block. Alternatively, if the comparison yields 0, the minimum value of T_L and its $id_L^{(\ell_{id})}$ are selected by the two multiplexers. The size of the overall circuit is $|\mathsf{MIN}^{\ell_x,\ell_{id},n}| = (n-1)\cdot(|\mathsf{CMP}^{\ell_x}| + |\mathsf{MUX}^{\ell_x}| + |\mathsf{MUX}^{\ell_{id}}|) = (n-1)(2\ell_x + \ell_{id})$



(a) Minimum block

(b) Minimum tree

Figure 5.12: Minimum Block (a) and Minimum Circuit (b).

non-XOR gates.

Some variants of the MIN construction are described in the following.

Only the minimum value is provided as output: in this case in each minimum block the MUX that selects among the identifiers is deleted. To obtain the total number of non-XOR gates, it is sufficient to set $\ell_{id} = 0$ obtaining a circuit having size $2(n-1)\ell_x$.

Only the identifier associated to the minimum value is provided as output: in the last level of the reverse tree the MUX selecting between $m_L^{(\ell_x)}$ and $m_R^{(\ell_x)}$ is deleted obtaining a circuit composed by $2(n-1)\ell_x + (n-2)\ell_{id})$ non-XOR gates[1].

The index is used instead of the identifier: if the index i_{min} is given as output we can replace each identifier id_i in the protocol with the value i represented with $p = \log_2 n$ bits. Anyway a more efficient circuit can be developed. Given the depth p of the reverse tree, at level d of the resulting tree we propagate only the d least significant bits $i_d^{(d)}$ of the minimum index. More specifically, each minimum block at depth d, gets as inputs the d least significant bits of their minimum indices $\mathbf{i}_{d,L}^{(d)}$ and $\mathbf{i}_{d,R}^{(d)}$ (together with $m_{d,L}^{(\ell_x)}$ and $m_{d,R}^{(\ell_x)}$), and outputs the minimum ℓ-bit value $m_{d+1}^{(\ell_x)}$ and $(d+1)$-bit minimum index $i_{d+1}^{(d+1)}$ of the tree. If the minimum value of T_L is larger than that of T_R (that is the comparison circuit outputs value 1), the minimum index $i_{d+1}^{(d+1)}$ is set to 1 concatenated with the minimum index of T_R selected by using a d-bit multiplexer. Alternatively, if the comparison yields 0, the value 0 concatenated with the minimum index of T_L is output. Overall, the size of the efficient minimum circuit is $|\mathsf{MIN}^{\ell_x,n}| = (n-1)\cdot(|\mathsf{CMP}^{\ell_x}| + |\mathsf{MUX}^{\ell_x}|) + \sum_{d=1}^{p} \frac{n}{2^d}|\mathsf{MUX}^{j-1}|$. Recalling that $2^p = n$, similarly to Section 5.3.2, we obtain $\sum_{d=1}^{p} \frac{n}{2^d}|\mathsf{MUX}^{j-1}| = n\sum_{d=1}^{p}(\frac{1}{2})^j(j-1) = n(\sum_{d=1}^{p} j(\frac{1}{2})^j)) - \sum_{d=1}^{p} j = n(\frac{n-1}{n} + \frac{2n-\log_2 n-2}{n}) = 3n - \log_2 n - 3$. Hence the circuit is composed by $2(n-1)\ell_x + 3n - \log_2 n - 3$ non-XOR gates.

---

[1] Note that this cut is not only needed for efficiency reasons, but even for granting security, whenever the minimum value has to be kept secret.

5.3.15 Distances

In this section we show how the distance between two arrays can be computed by using GC, starting with the Euclidean distance and later analysing the Hamming distance.

Euclidean

To evaluate the Euclidean distance between two vectors $\mathbf{p} = [p_0^{(\ell)}, \dots, p_n^{(\ell)}]$ and $\mathbf{q} = [q_0^{(\ell)}, \dots, q_n^{(\ell)}]$, where each element is represented with ℓ bits, we have to design a circuit computing $\delta = \sum_{i=1}^{n} (p_i - q_i)^2$. First of all the differences are computed by using SUB^ℓ circuits and the results are represented with $\ell + 1$ bits. Then we need to convert the results in their sign/modulus representation by using ADDSUB^ℓ circuits before computing the square differences by using SQR^ℓ circuits whose outputs are represented by 2ℓ bits. Finally δ is obtained by adding all the square differences with a reverse tree structure and the final output is represented by $2\ell + \lceil \log_2 n \rceil$ bits.

The circuit, shown in Fig. 5.13, is composed by $n|\mathsf{SUB}^\ell| + n|\mathsf{ADDSUB}^\ell| + n|\mathsf{SQR}^\ell| + |\{n - \mathsf{ADD}\}^{2\ell}| \approx n\ell + n\ell + n(\ell^2 - 1) + 2(n-2)\ell + n - 1 - \log_2(n - 1) - 1 = n\ell^2 + 4n\ell - 4\ell - \log_2(n-1) - 2$ non-XOR gates.

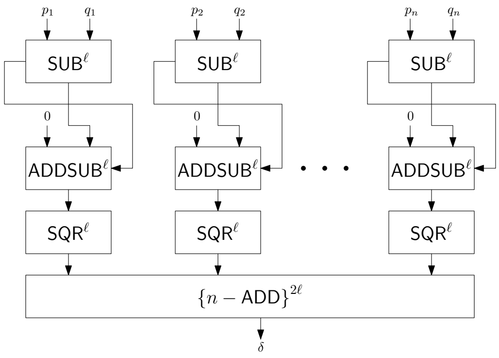


Figure 5.13: Euclidean Distance circuit.

Hamming

The Hamming distance is used when the elements of two vectors $\mathbf{p} = [p_0^{(1)}, \ldots, p_n^{(1)}]$ and $\mathbf{q} = [q_0^{(1)}, \ldots, q_n^{(1)}]$ are represented by 1 bit and is evaluated as $\delta = \sum_{i=0}^{n-1} p_i \oplus q_i$. The XOR among the two vectors is again represented with n bits, while the distance can be represented with $\lceil \log_2(m) \rceil$ bits. The circuit computing the Hamming distance, shown in Figure 5.14, is composed by n (free) XOR whose binary results are summed together by using a COUNT^n, obtaining the result $\delta^{(\lceil \log_2 n \rceil)}$ represented with $\lceil \log_2 n \rceil$ bits. The number of non-XOR gates composing the circuit is $|\mathsf{COUNT}^n| = n - \log_2(n+1)$.

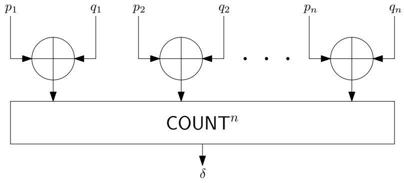


Figure 5.14: Hamming Distance circuit.

5.3.16 Logarithm (LOG)

Let $a^{(\ell)}$ be a positive integer number represented with ℓ bits and $a_{\ell-1} \ldots a_0$ its binary representation. The goal is to compute the logarithm of $a^{(\ell)}$ by using the circuit LOG proposed in [BGL10]. Instead of computing $b^{(\lceil \log_2 \ell \rceil)} = \lfloor \log_2 a^{(\ell)} \rfloor$ (in the next we use the notation $b^{(\log_2 \ell)}$ for simplicity) we evaluate a protocol that returns

$$b'^{(\log_2 \ell)} = \begin{cases} \lfloor \log_2 a^{(\ell)} \rfloor + 1 & \text{if } a^{(\ell)} > 0 \\ 0 & \text{if } a^{(\ell)} = 0. \end{cases} \tag{5.8}$$

The correct result is obtained by subtracting 1 to $b'^{(\log_2 \ell)}$. Note that the carry bit $b_{\log_2 \ell}$ is equal to 1 only if $a^{(\ell)} = 0$ and it can be used to identify a protocol error.

The idea of the protocol is as follows. Let p be the position of the most significant non-zero bit of $a^{(\ell)}$. From $a^{(\ell)}$ we build a number $c^{(\ell)}$ that has 0 bits in positions $a_{\ell-1}$ to a_{p+1} and 1 bits between a_p and a_0. Counting the number of one bits of $c^{(\ell)}$, we obtain the result. To obtain the value $c^{(\ell)}$

we can proceed as follows: we set $c_{\ell-1} = a_{\ell-1}$, then we compute recursively $c_i = c_{i+1} \vee a_i$, $\forall i = \ell - 2$ down to 0.

For example, if $a^{(8)} = 00001011$ we obtain $c^{(8)} = 00001111$ and counting the number of non-zero bits we obtain the result $\log_2 a^{(8)} = 4$. Observe that to compute $c^{(\ell)}$ we can see that we require $\ell - 1$ OR($\vee$) gates. At this point we use a COUNT^{ℓ} circuit to count the number of bits equal to 1. Finally the value is corrected by using a $\mathsf{SUB}^{\log_2 \ell}$ circuit.

The overall circuit (Figure 5.15) is composed by $|\mathsf{LOG}^{\ell}| = (\ell - 1)|OR| + |\mathsf{COUNT}^{\ell}| + |\mathsf{SUB}^{\log_2 \ell}| \approx (\ell - 1) + (\ell - \log_2 \ell) + \log_2 \ell = 2\ell - 1$ non-XOR gates.

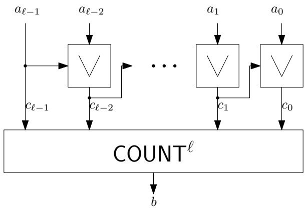


Figure 5.15: Logarithm circuit.

Chapter 6

Hybrid Protocols

As shown in the previous chapters, HE and GC have different characteristics, each with its pros and cons. In this chapter we first comapre HE and GC solutions for different functionalities to understand which of them is preferable and then show how GC and HE can be combined together to develop more efficient hybrid protocols.

In Section 6.1 we compare the implementation of the most important functionalities by using GC and HE to underline the importance of hybrid protocols that let the designer use the most efficient solution for each subprotocol. Section 6.2 shows how HE and GC subprotocols can be concatenated to develop hybrid protocols.

6.1 Comparison between HE and GC solutions

HE and GC solutions can be compared from both a computational and communication point of view. A bandwidth comparison is easy to perform, while it is really difficult to compare protocols from a computational aspect. Which is more expensive? n `exp` or m Hash functions? Being a computational comparison difficult, we focus only on communication complexity analysis and we evaluate computational complexity of real applications in the next chapters by analyzing real implementations of the protocols and measuring the execution times.

Before starting the comparison it is important to underline what can be precomputed in GC subprotocols. OT can always be precomputed: an OT_k^k is performed offline and then extended online to OT_t^n for any desired n. Differently the garbled circuit transmission can be precomputed only if $\mathcal{S}$ and $\mathcal{C}$ know that sometimes they have to evaluate a given functionality on data with fixed bitlength. Specifically, garbled circuit transmission can not be precomputed whenever

- $\mathcal{S}$ and $\mathcal{C}$ cooperate to evaluate many functionalities that can change each time;
- the data bitlength can change;
- the application is performed so frequently that there are not inactivity periods.

In the following we compare HE and GC solutions for some subprotocols presented previously. We consider short term security [GQ09], hence the security parameter for HE is $T = 1024$, while the security parameter for GC is $t = 80$.

6.1.1 Addition

Let us consider the sum (or subtraction) of two ℓ-bit numbers, the first one owned by $\mathcal{C}$ and the other by $\mathcal{S}$. By using HE, the sum between the 2 numbers is performed on the $\mathcal{S}$'s side without interaction. Only the transmission of the input cyphertext having size $2T$ (we always consider $\ell < T$) is required. By using GC and considering the online circuit transmission, the transmission of $4\ell t + 2\ell t + \ell t = 7\ell t$ bits is needed. The bandwidth is reduced to $3\ell t$ bits if the circuit is transmitted offline. Note that the HE solution returns a cyphertext available on the $\mathcal{S}$'s side, while GC outputs the result, or its relative secrets, on $\mathcal{C}$'s side.

A comparison of the bandwidth of the different solutions is shown in Figure 6.1. We can observe that GC is preferable to HE for small bitlengths. Generally the sum is a cheap operation with both the tools and the choice is made according to the previous or following operations.

6.1.2 Product

The complexity of the product computed by using HE changes a lot according to who owns what. To compare HE and GC products, we use two different representative examples. Other scenarios have complexity similar to the proposed two.

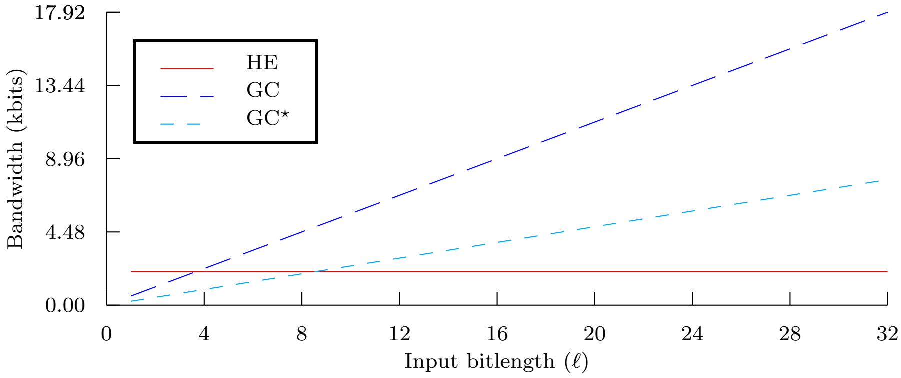


Figure 6.1: Communication Complexity of the sum (GC* indicates GC with offline circuit transmission).

In the first case an ℓ_x-bit operand is input from $\mathcal{C}$ and the other ℓ_y-bit operand is input from $\mathcal{S}$. The homomorphic solution has a bandwidth complexity of $2T$ bits and the result is available to $\mathcal{S}$ for further computation. The GC solution has complexity $(2\ell_x+\ell_y)t$ if the circuit is transmitted offline and $(2\ell_x\ell_y - \ell_x)4t + (2\ell_x + \ell_y)t = (8\ell_x\ell_y - 2\ell_x + \ell_y)t$ bits if the circuit is transmitted online. The comparison among GC and HE solutions is shown in Figure 6.2 where $\ell_x = \ell_y = \ell$ for simplicity. In the second scenario we suppose that both x and y are coming from previous computation and can not be revealed to anybody. Supposing that the operands come from HE computation,

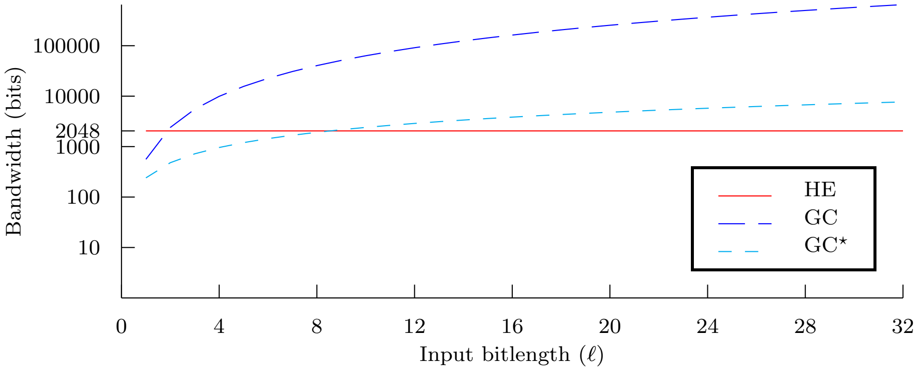


Figure 6.2: Communication complexity of product between operands input by $\mathcal{C}$ and $\mathcal{S}$ (GC* : GC with offline circuit transmission).

the HE solution requires the transmission of $4T$ bits and the output is still a cyphertexts available on $\mathcal{S}$'s side. The GC solution, supposing that $\mathcal{C}$ already obtained secrets relative to the operands from previous computation, requires a bandwidth of $(2\ell_x\ell_y - \ell_x)4t$ bits for the circuit transmission only if it is not transmitted offline. Complexities for this scenario are shown in Figure 6.3, where $\ell_x = \ell_y = \ell$ for simplicity.

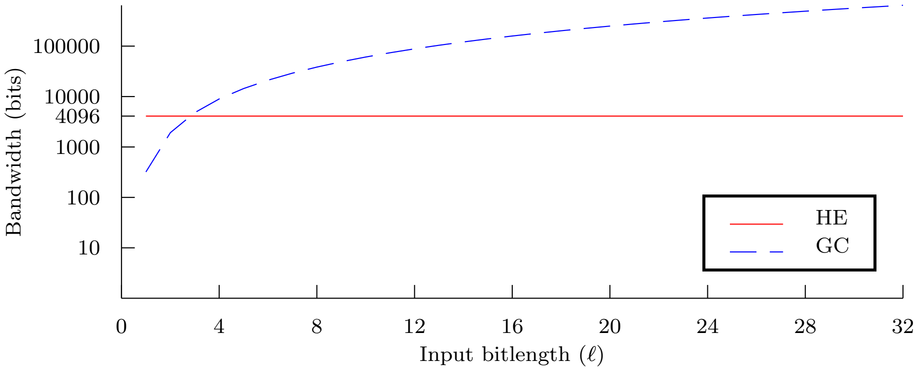


Figure 6.3: Communication complexity of product between secret values coming from previous computation.

From the complexity analysis we observe that multiplication by GC is more convenient than from HE only for small bitlengths of the operands or, sometimes, when the circuit can be transmitted offline. Similar results can be obtained for the square values and hence a HE solution is preferable even for scalar (and matrix) products, linear filtering and energy computation. Note that linear filtering applied to real signals with a fast sampling rate does not permit offline transmission of the circuit.

6.1.3 Multiplexer

We assume that a multiplexer is used to choose between two ℓ-bit values x ($\mathcal{C}$'s input) and y ($\mathcal{S}$'s input) according to a selection bit b coming from a previous computation. Different scenarios have similar complexity. The HE solution first requires the transmission of the cyphertext $[\![x]\!]$ and then other 2 cyphertexts are transmitted during the protocol, resulting in the transmission of $6T$ bits. GC protocol needs the transmission of secrets relative to y (ℓt

bits), the exchange through OT of the secrets relative to x ($2\ell t$ bits) and the circuit transmission if not performed offline ($4\ell t$ bits). A comparison of the complexities of the proposed solutions is provided in Figure 6.4. Similarly to the sum, GC solution is more efficient than the homomorphic one for small bitlengths.

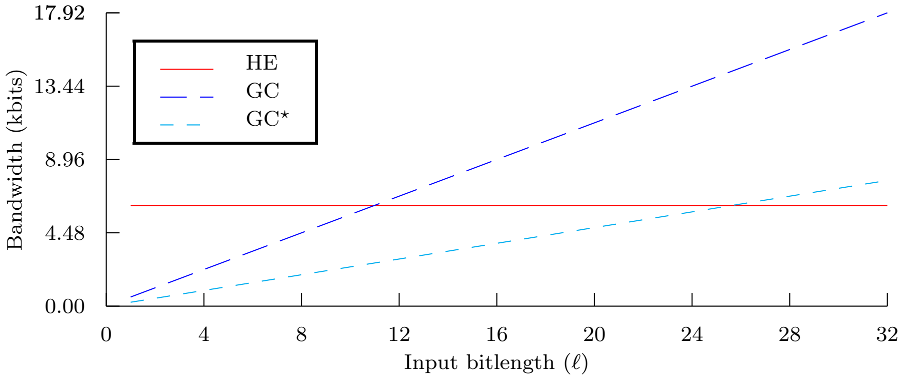


Figure 6.4: Communication Complexity of the multiplexer (GC* : GC with offline circuit transmission).

6.1.4 Comparison and minimum value

Let suppose that two ℓ bit values are compared. The first one is owned by $\mathcal{C}$ while the second one is owned by $\mathcal{S}$.

The HE solution requires the transmission of a cyphertext and $(4\ell + \lceil\log_2 \ell\rceil)2T$ bits for the evaluation of the "equality" comparison or $(2\ell+2)2T$ bits for the "lower than" comparison.

The GC solutions requires $3\ell t$ bits for the secret transmission and $\approx 4\ell t$ bits for the comparison circuit transmission[1], if not performed offline. As we can see in Figure 6.5, GC solutions are indeed more efficient than HE solutions.

This result, together with the one obtained by the multiplexer comparison, makes GC more efficient for minimum selection too.

---

[1] We approximate the equality check protocol complexity, $4(\ell-1)t$ bits, with the "lower than" complexity.

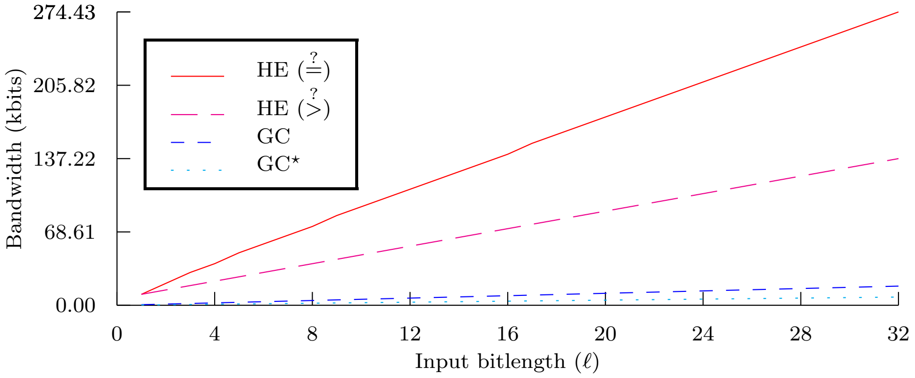


Figure 6.5: Communication Complexity of the comparison (GC* : GC with offline circuit transmission).

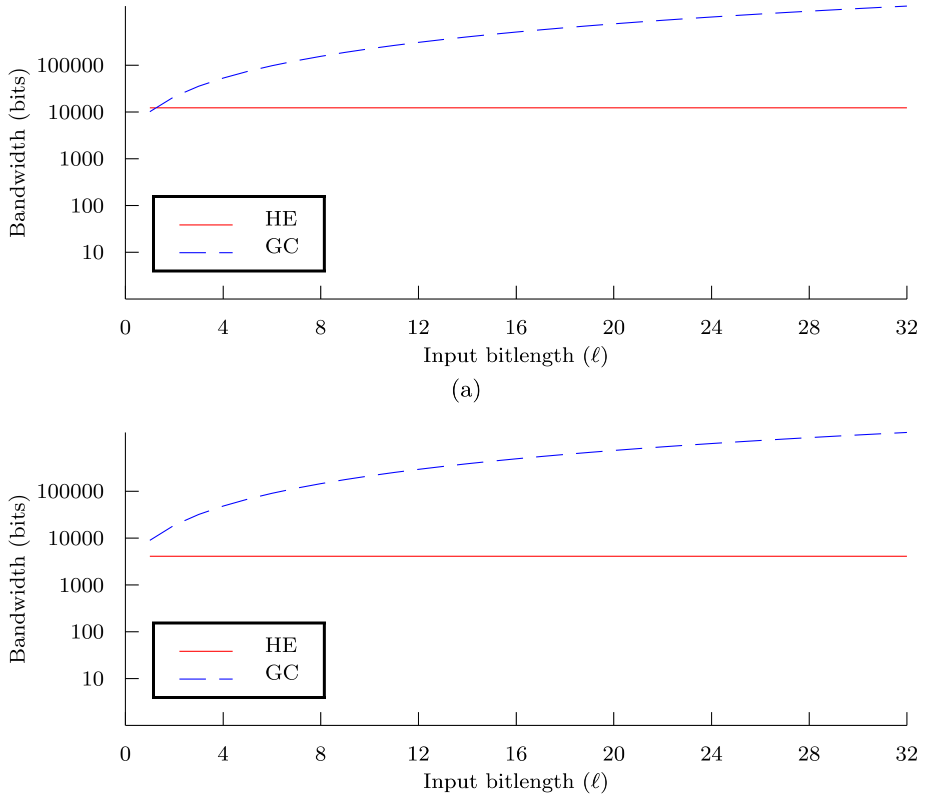
6.1.5 Distances

Euclidean distance

We consider the computation of the Euclidean distance between two arrays composed by n elements, each one represented by ℓ bits. We distinguish two cases.

In the first one the two arrays are respective inputs of $\mathcal{C}$ and $\mathcal{S}$. The homomorphic solution first needs the transmission of $n+1$ cyphertexts relative to the inputs and no more bandwidth is needed for the protocol. The GC solution requires $2n\ell t + n\ell t$ bits for the input secret transmission and $(n\ell^2 + 4n\ell + 3n - 4\ell - \log_2(n-1) - 6)4t$ bits for the circuit, if not transmitted offline. The complexity is shown in Figure 6.6(a) where $n = 5$, similar results hold for different n.

In the second scenario the two arrays come from previous computationS. During THE HE protocol $(p+1)2T$ bits are exchanged by using packing, where $p = \lceil n(\ell + \kappa)/N \rceil$, $\kappa = 80$ is the obfuscation security parameter. The GC protocols requires only the transmission of the circuit if not performed offline, i.e. $(n\ell^2 + 4n\ell + 3n - 4\ell - \log_2(n-1) - 6)4t$ bits. The comparison among the two solutions (with $n = 5$) is provided in Figure 6.6(b). tHE HE solution is indeed better than GC.



(a)

(b)

Figure 6.6: Communication Complexity of the Euclidean distance with arrays input from $\mathcal{C}$ and $\mathcal{S}$ (a) and with arrays coming from previous computation (b).

Hamming distance

We compare the HE and GC protocols that evaluate the Hamming distance between two binary arrays composed by n elements.

Given two n-bit arrays, owned by $\mathcal{C}$ and $\mathcal{S}$ respectively, the HE solution requires only the initial transmission of n cyphertexts ($n2T$ bits) while the GC solution needs the transmission of $2nt + nt$ bits for the input secrets and $(n - \lfloor \log_2(n+1) \rfloor)4t$ bits for the circuit, if not transmitted offline. Comparison results are shown in Figure 6.7. As opposed to the Euclidean distance, the Hamming distance based on GC is preferable. Intuitively this depends on the

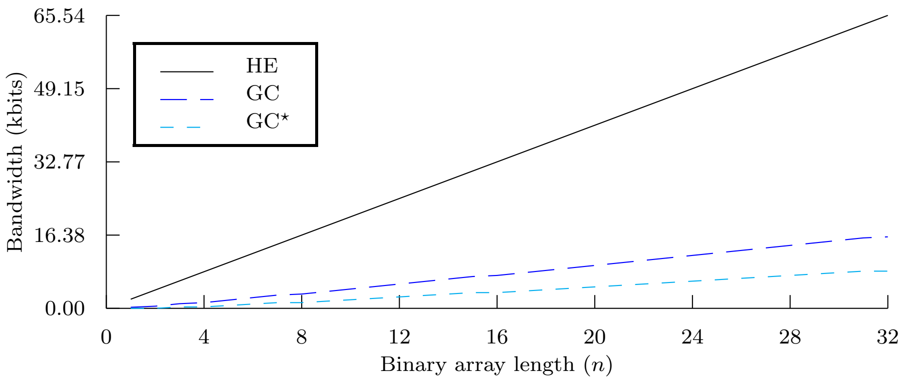


Figure 6.7: Communication Complexity of the Hamming distance (GC* : GC with offline circuit transmission).

binary nature of the elements of the arrays.

Note that if the Hamming distance is computed between arrays coming from previous computation, the HE complexity is replaced by the complexity of n more expensive eMul protocols, while the GC complexity decreases, removing the necessity of the secret input transmission. This makes GC even more performing.

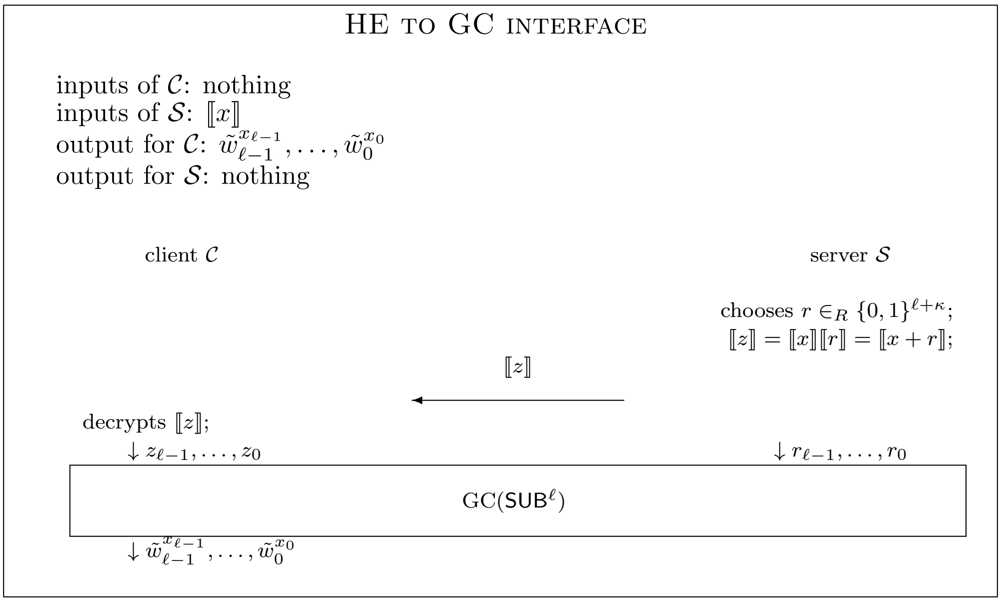
6.2 Hybrid HE and GC protocols

For more complex protocols, one may desire to take the best of the GC and HE worlds for different parts of the protocol, thus resulting in hybrid protocols [KSS09a]. Doing so, however, requires that the subprotocols are connected in such a way that the security of the whole system is preserved. At the same time it is necessary that the representation used to for the input and output values are adapted to the subprotocol requirements.

6.2.1 Interface from HE to GC subprotocols

We now show how we can use a GC subprotocol after an HE subprotocol. We suppose that $\mathcal{S}$ holds a value $[\![x]\!]$, coming from previous HE computations. We also assume that the overall MPC protocol requires that x is used in a subprotocol based on GC, eventually together with other variables $\mathbf{y}_\mathcal{C}$, $\mathbf{y}_\mathcal{S}$, owned by $\mathcal{C}$ and $\mathcal{S}$ respectively, with the aim of evaluating a given functionality $f(x, \mathbf{y}_\mathcal{C}, \mathbf{y}_\mathcal{S})$, without that the exact value of x is revealed either to $\mathcal{C}$ or $\mathcal{S}$, because this would cause a leakage of information. We only assume that an upper bound of x is known, i.e. we know that x can be represented by ℓ bits ($x < 2^\ell$).

$\mathcal{S}$ and $\mathcal{C}$ evaluates the interface shown in Protocol 22 to let $\mathcal{C}$ obtain the secrets relative to x. In the same time they carry out all the transmissions relative to the functionality $f(\cdot)$: the OT that permits to $\mathcal{C}$ to obtain the secrets relative to $\mathbf{y}_\mathcal{C}$ and the transmission of the secrets relative to $\mathbf{y}_\mathcal{S}$ from $\mathcal{S}$. Finally, once the obfuscation has been removed, $\mathcal{C}$ evaluates the circuit that computes $f(\cdot)$.


HE TO GC INTERFACE
inputs of $\mathcal{C}$: nothing
inputs of $\mathcal{S}$: $[\![x]\!]$
output for $\mathcal{C}$: $\tilde{w}_{\ell-1}^{x_{\ell-1}}, \dots, \tilde{w}_0^{x_0}$
output for $\mathcal{S}$: nothing
client $\mathcal{C}$
server $\mathcal{S}$
chooses $r \in_R \{0,1\}^{\ell+\kappa}$;
$[\![z]\!] = [\![x]\!][\![r]\!] = [\![x+r]\!]$;
$[\![z]\!]$
decrypts $[\![z]\!]$;
$\downarrow z_{\ell-1}, \dots, z_0$
$\downarrow r_{\ell-1}, \dots, r_0$
GC(SUB$^\ell$)
$\downarrow \tilde{w}_{\ell-1}^{x_{\ell-1}}, \dots, \tilde{w}_0^{x_0}$

Protocol 22: Interactive interface that returns to $\mathcal{C}$ the secrets relative to x, given the cyphertext $[\![x]\!]$ available to $\mathcal{S}$.

In the interface protocol, $\mathcal{S}$ picks a random number r in $\mathbb{Z}_{2^{\ell+\kappa}}$, where κ is a security obfuscation parameter[2], and additively blinds x by using the homomorphic property of the Paillier cryptosystem, obtaining $[\![z]\!] = [\![x]\!][\![r]\!] = [\![x+r]\!]$. Then $\mathcal{S}$ transmits $[\![z]\!]$ to $\mathcal{C}$. $\mathcal{C}$ decrypts the cyphertext and uses the

---

[2] Similarly to the security parameter for symmetric encryption schemes, usually $\kappa = 80$.

ℓ least significant bits of z[3] in an OT protocol returning the secrets $\{\tilde{w}_i^{z_i}\}_{i=0}^{\ell-1}$ to be used in the subtraction GC. Moreover $\mathcal{S}$ sends to $\mathcal{C}$ the keys $\{\tilde{w}_j^{r_j}\}_{j=0}^{\ell-1}$ relative to the ℓ least significant bits of r. Finally $\mathcal{C}$ uses all the keys to evaluate a garbled circuit that, after a subtraction, returns the keys $\{\tilde{w}_i^{x_i}\}_{i=0}^{\ell-1}$ relative to x and then uses them to evaluate the desired functionality.

Let us analyze the various steps of the interfacing protocol above. $\mathcal{S}$ performs 1 `enc` to blind $[\![x]\!]$. During the transmission round, $\mathcal{S}$ sends a cyphertext ($2T$ bits) to $\mathcal{C}$. $\mathcal{C}$ performs 1 `dec` and then $\mathcal{C}$ and $\mathcal{S}$ run the online phase, consisting of ℓ *OT*s, thus transmitting - in 2 rounds - $\approx 2\ell t$ bits. During the second round $\mathcal{S}$ appends the $\approx \ell t$ bits of the keys relative to r. Moreover if the circuit transmission is not performed offline, $\mathcal{S}$ computes 4ℓ Hash operations before transmitting the $4\ell t$ bits relative to the garbled tables in the second round. Finally $\mathcal{C}$ evaluates the gates of the subtraction circuit, performing ℓ Hash operations to evaluate the non-Xor gates. Note that circuit garbling and the secret and circuit transmission can be performed together to the circuit that has x as inputs, hence the only additional communication round of the protocol is the one needed for the cyphertext transmission. The interface complexity is shown in Table 6.1.

Circuit		**Computation**		**Communication**	
# Hash	Bandwidth	# `exp`	# Hash	Rounds	Bandwidth
4ℓ	$4\ell t$	2	ℓ	1	$\approx 2T + 3\ell t$

Table 6.1: Complexity of the HE to GC interface.

6.2.2 Interface from GC to HE subprotocols

We now show how we can use a HE subprotocol after a GC subprotocol. The solution is similar to the one provided for the conversion from HE to GC. We suppose that $\mathcal{C}$ knows the secrets $\tilde{w}_{\ell-1}^{x_{\ell-1}}, \ldots, \tilde{w}_0^{x_0}$ relative to a value x, coming from previous GC computations, while the overall MPC protocol requires to use $[\![x]\!]$ in a subprotocol based on HE, possibly together with other variables, without revealing the exact value of x to either $\mathcal{C}$ or $\mathcal{S}$, because this would

---

[3] A subtraction circuit starts the computation from the least significant bit and after the processing of the first ℓ input bits the original value is already obtained, hence processing of the other most significant bits is not necessary.

cause a leakage of information. $\mathcal{S}$ and $\mathcal{C}$ evaluate together the interface shown in Protocol 23 to let $\mathcal{S}$ obtain $[\![x]\!]$.

GC TO HE INTERFACE

inputs of $\mathcal{C}$: $\tilde{w}_{\ell-1}^{x_{\ell-1}}, \dots, \tilde{w}_0^{x_0}$
inputs of $\mathcal{S}$: nothing
output for $\mathcal{C}$: nothing
output for $\mathcal{S}$: $[\![x]\!]$

client $\mathcal{C}$		server $\mathcal{S}$
		chooses $r \in_R \{0,1\}^{\ell+\kappa}$;
$\downarrow x_{\ell-1}, \dots, x_0$		$\downarrow r_{\ell+\kappa-1}, \dots, r_0$
	$\mathsf{GC}(\mathsf{ADD}^{\ell+\kappa})$	
$\downarrow z^{(\ell+\kappa+1)} = x^{(\ell+\kappa)} + r^{(\ell+\kappa)}$; encrypts z;		
	$\xrightarrow{[\![z]\!]}$	$[\![x]\!] = [\![z]\!][\![-r]\!] = [\![z-r]\!]$.

Protocol 23: Interactive interface that returns the cyphertext $[\![x]\!]$ to $\mathcal{S}$, given the secrets $\tilde{w}_{\ell-1}^{x_{\ell-1}}, \dots, \tilde{w}_0^{x_0}$ relative to x available to $\mathcal{C}$.

In this interface protocol, $\mathcal{S}$ picks a random number r in $\mathbb{Z}_{2^{\ell+\kappa}}$, where κ is a security obfuscation parameter, and uses it in a garbled addition circuit to blind x obtaining $z^{(\ell+\kappa+1)} = x^{(\ell+\kappa)} + r^{(\ell+\kappa)}$. The addition circuit is prepared and transmitted (maybe offline) together to the circuit implementing the functionality that produces the value $x^{(\ell)}$. The same holds for the transmission of the secrets relative to $r^{(\ell+\kappa)}$. Hence no additional rounds are required for the addition. Then $\mathcal{C}$ obtains z, encrypts it and transmits it to $\mathcal{S}$. Finally $\mathcal{S}$ removes the obfuscation by using the additive homomorphic property.

Let us analyze the complexity of the interfacing protocol described above. $\mathcal{S}$ transmits the $\ell + \kappa$ secrets relative to r to $\mathcal{C}$ that obtains the secrets of x by evaluating another circuit. If not performed offline, an additional $\mathsf{ADD}^{\ell+\kappa}$ circuit is transmitted. After having obtained the plain z as output of the circuit, $\mathcal{C}$ performs 1 `enc` and transmits a cyphertext to $\mathcal{S}$. Finally, to remove the obfuscation, $\mathcal{S}$ performs 1 `enc` of $-r$. The interface complexity is shown in Table 6.2.

Circuit		**Computation**		**Communication**	
# Hash	Bandwidth	# `exp`	# Hash	Rounds	Bandwidth
$4(\ell+\kappa)$	$4(\ell+\kappa)t$	2	$\ell+\kappa$	1	$2T+(\ell+\kappa)t$

Table 6.2: Complexity of the HE to GC interface.

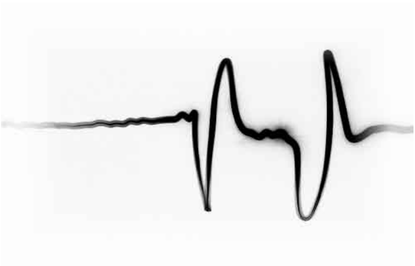
Part II

Electrocardiogram Classification

Secure Two-Party Computation has many interesting application fields, such as biometric analysis, research on encrypted databases, data aggregation, etc. In this thesis we focus on Biomedical applications, specifically on ElectroCardioGram analysis. Privacy preserving protocols may be used in remote healthcare scenarios, whenever the patient is interested to protect his biomedical data, even from unauthorized access from hospital staff, while the remote health care provider is interested to protect its intellectual property, i.e. the data input to the protocol.

To demonstrate that privacy preserving biomedical analysis is feasible, we propose an implementation of the ElectroCardioGram (ECG) classification algorithm proposed by Ge et al. [GSK02], that classifies each single heart beat according to 6 classes representing different diseases. Two different implementations are proposed. Both of them are based on the new proposed Linear Branching Programs tool, that perfectly fits the algorithm proposed by Ge et al. The first one is implemented by using a full-GC protocol and the second one by using an Hybrid protocol. Moreover ECG beats have been also classified by using a secure implementation of a Neural Networks, having the features of LBP as input.

A study of the complexity as well as a prototype implementation shows that, thanks to specific optimization, we can develop protocols with performance close to real time, opening the way to real application of s.p.e.d. techniques for remote biomedical analysis.



> "I take risks, sometimes patients die.
> But not taking risks causes more patients to die,
> so I guess my biggest problem is I've been cursed with the ability to do the math."
> *Gregory House (House M.D., Season 1, Episode 11: "Detox")*

Chapter 7

Plain Protocol

Before describing the application of privacy preserving protocols to ElectroCardioGram (ECG) classification, we present the plain domain protocol ([GSK02] and [ASSK07, ch.8]) for ECG classification that inspired us.

In this chapter, we provide a simple introduction to heart and electrocardiogram in Section 7.1 and then describe the overall architecture of the plain domain version of the ECG classification system in Section 7.2, where we provide a detailed description of the various modules the system consists of, and finally we give some performance result of the plain protocol. For sake of brevity, we focus on the modules that, according to the privacy-preserving implementation described in the next chapter, must be implemented securely, and we give only a brief description of the modules that are carried out in the plain domain by one of the two parties.

7.1 Introduction to Electrocardiogram and heart diseases [ASSK07]

The heart is an efficient muscular organ that pumps blood throughout the body. Blood brings nutrients and oxygen to tissues, and carries away metabolic waste and carbon dioxide for excretion through the kidneys and the lungs, respectively.

Pumping is efficient only when the heart contracts in a coordinated manner. Blood must first fill the atria, and then be pumped into the ventricles before being forcefully ejected. This coordination is achieved by an elaborate electrical conduction system that controls the precise timing for depolarizing the substantial mass of electrically excitable myocardium. This delicate control starts with an intrinsic self-excitable cardiac pacemaker (the *Sinus-Atrial node*) which sets the rate at which the heart beats. The pacemaker spontaneously generates regular electrical impulses, which then spread through the conduction system of the heart and initiate contraction of the myocardium.

The electrocardiogram. ECG is a graphical recording of the electrical signals generated by the heart. The signals are generated when cardiac muscles depolarize in response to electrical impulses generated by pacemaker cells. Upon depolarization, the muscles contract and pump blood throughout the body. The ECG reveals many things about the heart, including its rhythm, whether its electrical conduction paths are intact, whether certain chambers are enlarged, and even the approximate ischemic location in the event of a heart attack (myocardial infarction).

Surface (skin) electrodes are used to detect the depolarization of excitable myocardium. The placement of the electrodes determines the directional viewpoint of the heart. Each viewpoint is called a lead. The standard ECG as recorded by clinicians is the 12-lead ECG, which uses 10 electrodes. A single-lead ECG recorder would typically have three electrodes: the positive electrode, the negative electrode and an indifferent electrode (ground or right-leg drive electrode). Electrodes detect ionic current flow within the body by detecting the potential difference between them as current flows through resistive tissue. The leads are classified as:

THE LIMB LEADS (BIPOLAR) - LEADS I, II, III: The limb leads are called as such because the electrodes are attached to the limbs as in Figure 7.1: left arm, right arm and a leg (usually the left leg). Three views are immediately obtained. Leads I, II and III are commonly referred to bipolar leads as they use only two electrodes to derive a view. One electrode acts as the positive electrode while the other as the negative electrode (hence bipolar).

AUGMENTED LIMB LEADS (UNIPOLAR) - LEADS AVL, AVR, AVL: The signals from the limb electrodes can be combined to give further views called the augmented leads. One of the limb electrodes serves as the positive electrode. The negative electrode is virtual, being the average of the signals from the remaining two limb electrodes. In contrast to Leads I, II and III, the augmented leads are known as unipolar leads.

Precordial Leads (Unipolar) - Leads V1,...,V6: Six views of the heart signals across the front (ventral aspect) of the chest are given from these six chest electrodes (Figure 7.1). The views fall along the transverse plane (i.e. looking into the chest). The positive electrode is the chest electrode. The negative electrode is a virtual electrode commonly called the Wilson Central Terminal (WCT), realized by electrically averaging the signals from the three electrodes LA, RA and LL. The WCT is thus the electrical center of the heart.

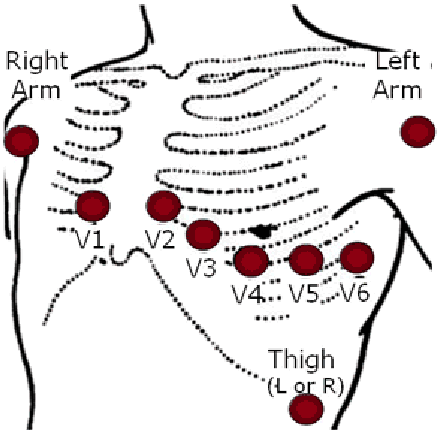


Figure 7.1: Positions where to place electrodes.

A typical ECG recording from a normal person (Figure 7.2) is shown below. The ECG is principally described by waves that are labeled using the letters P, QRS and T. The meaning of the waves may be broadly described as follows:

- P-wave corresponds to the depolarization of the atrial myocardium (muscles of upper chambers of the heart), and indicates the start of atrial contraction that pumps blood to the ventricles.
- The Q, R, and S waves are usually treated as a single composite wave known as the QRS-complex. The QRS-complex reflects the depolarization of ventricular myocardium, and indicates the start of ventricular contraction that pumps blood to the lungs and the rest of the body.

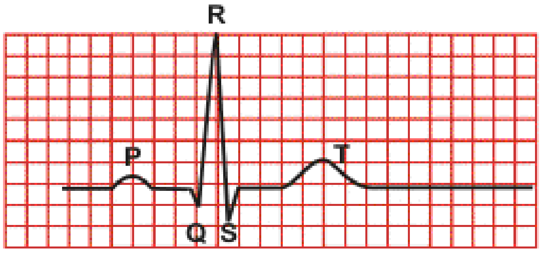


Figure 7.2: Waves on ECG trace.

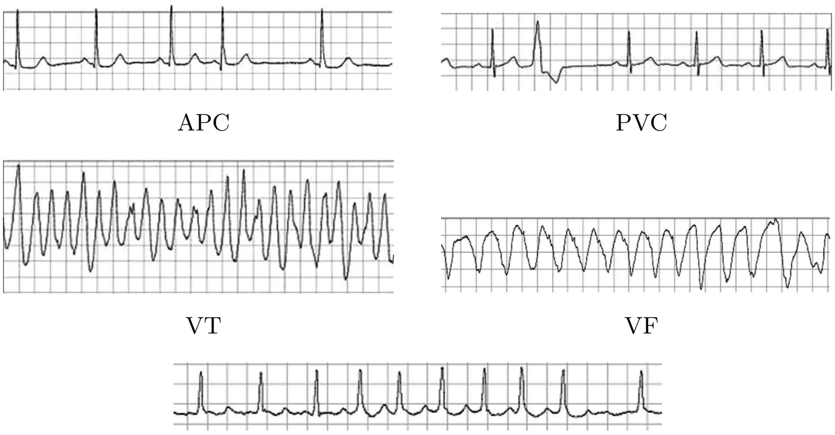
- The T-wave corresponds to the depolarization of the ventricular myocardium, which is a necessary recovery process for the myocardium to depolarize and contract again. The end of the T-wave coincides with the end of ventricular contraction. Atrial depolarization (Ta-wave) is usually not visible as it normally coincides with the QRS-complex (and is buried in the larger waveform).

Diseases. The manner in which the heart contracts over time determines the rhythm of the heart. Normal sinus rhythm (NSR) is the normal rhythm of the heart when there is no disease or disorder affecting it.

Rhythms that deviate from NSR are called arrhythmias since they are abnormal and dysfunctional. Arrhythmias can be life-threatening. If the heart rate is too slow as in bradycardia, perfusion may be insufficient and this can adversely affect vital organs. Similarly, if the heart rate is too fast, the ventricles are not completely filled before contraction and pumping efficiency drops, adversely affecting perfusion.

Many types of arrhythmias exist. We here present only those we consider for the classification (Figure 7.3).

ATRIAL PREMATURE CONTRACTIONS(APC): it results in an earlier than expected occurrence of a (non-sinus) P'-wave followed by a QRS-complex and a T-wave. This happens because of an ectopic pacemaker firing before the Sinus-Atrial (S-A) node does. The ectopic pacemaker may reside in any part of the atria outside the S-A node.



APC

PVC

VT

VF

SVT

Figure 7.3: Examples of Arrhythmias considered for the classification.

Premature Ventricular Contractions (PVC): it is an extra (abnormal) ventricular contraction originating from a pacemaker located in the ventricles. More than one pacemaker may be involved, each generating its own bizarre-shaped QRS-complex. PVCs usually do not depolarize the atria or the S-A node and hence the P-waves maintain their underlying rhythm and occur at the expected time. PVCs are not preceded by (ectopic) P-waves and may occur anywhere in the heart beat cycle. More PVC can degenerate in Ventricular Tachycardia.

Ventricular Tachycardia (VT): the heart rate is 110 to 250 beats per minute. The QRS complex is abnormally wide, bizarre in shape, and of a different direction from the normal QRS complex. VT is considered life-threatening as the rapid rate may prevent effective ventricular filling and result in a drop in cardiac output. It can also degenerate into ventricular fibrillation, which is lethal.

Ventricular Fibrillation (VF): it occurs when numerous ectopic pacemakers in the ventricles cause different parts of the myocardium to contract at different times in a non-synchronised fashion. Ventricular contraction is uncoordinated, resulting in insignificant or no cardiac output. Perfusion to vital organs is compromised and death ensues in minutes. Defibrillation (DC shock) is used to abort ventricular fibrillation.

SupraVentricular Tachycardia (SVT): the heart rate ranges from 160 to 240 beats per minute. SVT may occur as a result of a re-entry circuit involving an accessory pathway between the atria and ventricles (Atrioventricular re-entry tachycardia). The onset and termination of SVT is abrupt, and may occur in repeated episodes that last for seconds, hours or days. P-waves are usually buried in the QRS complex and hence not visible.

7.2 ECG Classification

In this thesis we are interested in classifying each QRS complex (corresponding to a single heart beat) according to 6 classes introduced in the previous section, namely: Normal Sinus rhythm (NSR), Atrial Premature Contractions (APC), Premature Ventricular Contractions (PVC), Ventricular Fibrillation (VF), Ventricular Tachycardia (VT) and SupraVentricular Tachycardia (SVT). The classification algorithm we used is inspired to the works by Ge et al. ([GSK02] and [ASSK07, ch.8]). Specifically, it relies on a rather general technique based on AR models for ECG description and a subsequent QDF classifier. Specifically, each ECG interval corresponding to one heart beat is modeled by means of a 4-th order AR model. The AR model parameters and two features related to the modeling error are fed to a QDF classifier that takes the final decision about the presence of a particular disease.

The choice of the algorithm is justified first of all by the good classification accuracy it ensures, secondly because it fits well the requirements set by the SSP framework, finally because of its generality. As a matter of fact, both AR models and QDF are often used in automatic medical diagnosis, for instance in [BCA+93] the authors compare various autoregressive models applied to time series of ECG. In [GHX07] and in [ZJGX04] scalar autoregressive models and multivariate autoregressive are studied to extract ECG features, and then these features are combined with quadratic discriminant function to classify various heart disease. Logistic regression and discriminant analysis are used in [TWBT95] to identify particular coronary heart diseases. QDF is used by

[FLK08] to realize an investigation on fibromyalgia. In [KTB+03] linear and quadratic discriminant function analysis and neural computational system are used to detect prostate carcinoma.

For this reason the privacy-preserving solutions we have developed can represent the basis for a large number of different implementations addressing a variety of diverse topics in biomedical signal processing.

7.2.1 Overall architecture of the classifier

The overall architecture of the classifier, as proposed by Ge et al., is summarized by the block diagram in Fig. 7.4. The input of the system is an ECG signal sampled at 250 sample per second. The ECG is first filtered (notch filter) to remove noise artifacts [1]. The R peaks, each corresponding to an heart beat, are detected, and 300 samples surrounding each peak are extracted. The extracted samples are modeled by a 4-th order AR model whose parameters, together with two additional parameters accounting for the modeling error, are used by the QDF block to produce the feature vector that will be used for the subsequent classification. Each block of Fig. 7.4 is described in the following.

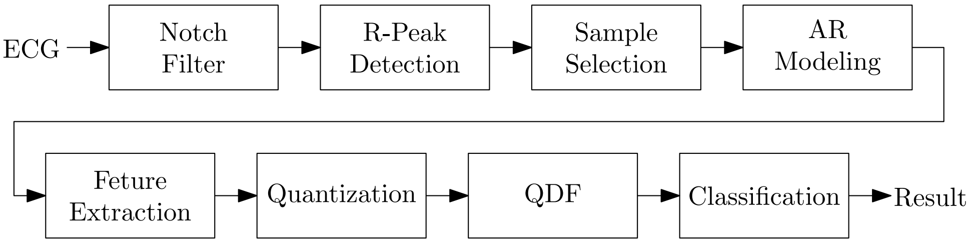


Figure 7.4: Block Diagram.

Notch filter. Noise removal is the first step of the chain, such a step is needed to remove noise due to power line interference, electrode contact's noise, motion artifact and base line wander. Frequencies lower than a cutoff frequency $f_l = 0.7Hz$ and closest to $f_p = 50Hz$ have to be filtered. The filter

---

[1] We will discuss noise affecting ECG further in Section 10.1, before facing the problem of measuring signal quality in the encrypted domain.

proposed in [VAS85] is used, whereby the filtered signal is computed from the original signal through the following equation:

$$y_n = \left(\sum_{i=0}^{(M-3)/2} h_{ik} \cdot \left(x_{n-ki} + x_{n-k(M-1)+ki}\right) \right) + h_{\frac{M-1}{2}k} \cdot x_{n-k\left(\frac{M-1}{2}\right)}$$

where $k = 5$, $M = 51$, y_n is the output signal sample, x_n is the input signal sample and h_{ik} are the filter impulse response coefficients, that are computed to obtain a periodic frequency spectrum of the filter $H_t(f)$ according to Figure 7.5 and are defined as:

$$h_{ik} = \frac{\sin\left(2\pi(ik - \frac{M-1}{2})\frac{f_l}{f_p}\right)}{\pi(ik - \frac{M-1}{2})} \qquad h_{\frac{M-1}{2}k} = 1 - \frac{2f_l}{f_p}. \tag{7.1}$$

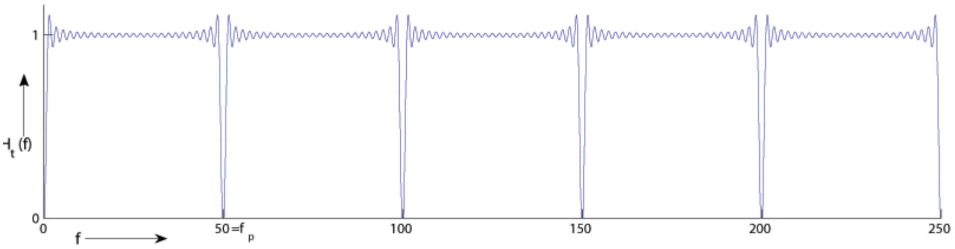


Figure 7.5: The distorted frequency response as derived from a truncated impulse response (Note: scales are linear).

R peaks detection. The Pan-Tompkins algorithm [PT85] is applied to determine the R-peaks positions. Figure 7.6 shows a block scheme that summarizes the algorithm.

The algorithm starts by filtering the ECG to distinguish the R peak from other peak that may be mistakenly identified as an R peak. The cutoff frequency of the low pass filter is about 11 Hz with gain 36. The filter is defined by the following equation:

$$y_n = 2y_{n-1} - y_{n-2} + x_n - 2x_{n-6} + x_{n-12}. \tag{7.2}$$

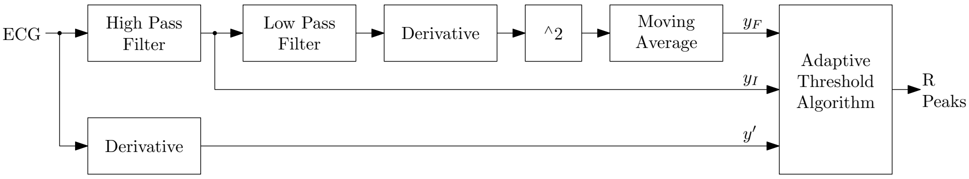


Figure 7.6: Pan - Tompkins Algorithm.

The cutoff frequency of the high pass filter is 5 Hz with gain 32. The filter is defined by the following differences equation:

$$y_n = y_{n-1} - \frac{x_n}{32} + x_{n-16} - x_{n-17} + \frac{x_{n-32}}{32}. \tag{7.3}$$

The output of the high pass filter is named y_I. Then, a derivative step is performed to obtain the QRS-complex slope information. The difference equation to realize this is:

$$y_n = \frac{1}{8}\left(2x_n + x_{n-1} - x_{n-3} - 2x_{n-4}\right). \tag{7.4}$$

At this point samples are squared to amplify the high slope of the R peak. Finally a moving average filter with window size $N = 30$ samples is applied to the squared signal:

$$y_n = \frac{1}{N}\sum_{i=0}^{N} x_{n-N+i}. \tag{7.5}$$

The output of the moving average filter is named y_F.

R peaks are detected using upward and downward thresholding on the signal produced by the High Pass Filter y_I and the signal after Moving Average Filter y_F. Thresholds are computed dynamically. Two sets of threshold are used, each of which has two threshold levels.

The R-peaks are detected by using the adaptive thresholds algorithm described in [PT85]. We avoid a detailed description of the algorithm, being it out of the purpose of this thesis, but we provide the general idea behind it.

The algorithm examines both the signals y_I and y_F to identify the R peaks. The first step is to determinate the peaks by the sign change in the signal derivatives y'. In each signal the peaks are then compared with the threshold. If it is greater than the threshold, the peak is classified as a signal peak, otherwise as a noise peak. In both the cases the value of the peak is used to update the relative threshold. If a peak is identified from thresholding as a signal peak in both the signals y_I and y_F, than it is classified to be an R peak. Whenever the algorithm does not find an R peak in the interval starting after the last identified R peak and long 1.66 times the mean RR interval, there is a rollback so the algorithm restarts from the last R peak found and halves the thresholds.

ECG Chunk Selection. Once an R peak is identified, 100 samples before the peak and 200 samples following the peak are selected corresponding to 1.2 seconds of the ECG signal. At the end of this step ECG samples containing the to-be-classified peak are obtained.

AR Modeling. An autoregressive model is a linear predictor that attempts to estimate the value of an output y_n from the previous p inputs $(x_{n-1}, x_{n-2}, \ldots, x_{n-p})$, through an equation having the following form:

$$y_n = \sum_{i=1}^{p} a_i x_{n-i} + \epsilon_n \tag{7.6}$$

where $\{a_i\}_{i=1..p}$ are the AR coefficients, $\{\epsilon_i\}_{i=1..N}$ is the prediction error sequence and p is the model degree. The AR model may be used to predict the successive values of the input. The goal is to derive the best values for the AR coefficients in the sense of minimizing the mean square error between the original sequence and the one predicted by the model. As shown in [GSK02], for an accurate description of ECG signals an AR model of order $p = 4$ is sufficient. The vector $\mathbf{a}$ with the AR coefficients can be calculated solving the linear system $\mathbf{Ra} = \mathbf{r}$, where $\mathbf{R}$ is the autocorrelation matrix of the input samples:

$$\mathbf{R} = \begin{pmatrix} 1 & r_1 & r_2 & r_3 \\ r_1 & 1 & r_1 & r_2 \\ r_2 & r_1 & 1 & r_1 \\ r_3 & r_2 & r_1 & 1 \end{pmatrix} \tag{7.7}$$

with

$$\begin{aligned} r_k &= \frac{c_k}{c_0}, & (7.8)\\ c_k &= \frac{1}{N-k}\sum_{n=1}^{N-k}(x_n-\bar{x})(x_{n+k}-\bar{x}), & (7.9)\end{aligned}$$

N is the number of samples and $\bar{x}$ is the mean of the input sequence; and where $\mathbf{r} = (r_1, r_2, r_3, r_4)^\top$.

The AR coefficients can be computed solving the linear system $\mathbf{R}\,\mathbf{a} = \mathbf{r}$:

$$\begin{pmatrix} 1 & r_1 & r_2 & r_3 \\ r_1 & 1 & r_1 & r_2 \\ r_2 & r_1 & 1 & r_1 \\ r_3 & r_2 & r_1 & 1 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{pmatrix} = \begin{pmatrix} r_1 \\ r_2 \\ r_3 \\ r_4 \end{pmatrix} \tag{7.10}$$

Feature Extraction. From the AR model, six features are extracted. The feature vector is the following:

$$\mathbf{f} = (f_1, f_2, f_3, f_4, f_5, f_6)^\top = (a_1, a_2, a_3, a_4, n_1, n_2)^\top \tag{7.11}$$

The first four features are the coefficients of the AR model; n_1 is the number of times that the amplitude of $|\epsilon_n|$ exceeds an empiric threshold:

$$th = 0.25 \max_n (|\epsilon_n|) \tag{7.12}$$

and $n_2 = 296 - n_1$.

Quadratic Discriminant Function. With QDF classifications, the classifier does not operate directly on the feature vector $\mathbf{f}$. Instead a composite feature vector $\mathbf{f^c}$ is computed containing the features in $\mathbf{f}$, their square values and their cross products, namely:

$$\begin{aligned} \mathbf{f^c} &= (1, f_1, \ldots, f_6, f_1^2, \ldots, f_6^2, f_1 f_2, \ldots, f_1 f_6, f_2 f_3, \ldots, f_2 f_6, \ldots, f_5 f_6)^\top \\ &= (f_1^c, f_2^c \ldots f_{28}^c)^\top & (7.13)\end{aligned}$$

The vector $\mathbf{f^c}$ is projected onto 6 directions β_i, obtaining a 6-long vector $\mathbf{y}$, that represents the input for the final classification step:

$$\mathbf{y} = \mathbf{B}\mathbf{f^c} \tag{7.14}$$

where $\mathbf{B}$ is a matrix whose rows are the vectors β_i. The matrix $\mathbf{B}$ contains part of the knowledge embedded within the classification system, and is computed by relying on a set of training ECG's. In particular, the matrix $\mathbf{B}$ is computed by minimizing the mean square error between the actual vector $\mathbf{y}$ and the target values of $\mathbf{y}$. A vector used for the training set is said $\mathbf{y_t}$. The desired output for each of the diseases we want to classify is given in Table 7.1.

y_1	y_2	y_3	y_4	y_5	y_6	Disease
1	1	x	1	1	x	NSR
1	1	x	1	x	1	NSR
1	1	x	1	-1	x	APC
1	1	x	1	x	-1	PVC
-1	x	-1	x	x	x	VF
-1	x	1	x	x	x	VT
1	-1	x	x	x	x	SVT

Table 7.1: Classification pattern ("x" means that the value of the variable does not influence the classification.).

Let us now consider the estimation of a single row of $\mathbf{B}$, that is β_i. During the training,for each ECG segment j, the composite feature vector $\mathbf{f^c}_j$ is computed. Assuming that D is the number of ECG segments used for the training of a particular β_i, the vector that minimizes the error between the target output and the actual one is given by (see [ASSK07] for more details):

$$\beta_i = (\mathbf{A}_i^\top \mathbf{A}_i)^{-1} \mathbf{A}_i^\top \tilde{\mathbf{y}}_{it}. \tag{7.15}$$

where $\mathbf{A}_i = (\mathbf{f^c}_1, \mathbf{f^c}_2 \ldots \mathbf{f^c}_D)$ is a $D \times 28$ matrix containing the composite feature vectors of the training set, and where $\mathbf{y_t}_i = (y_{i1}, \ldots, y_{iD})^\top$ is the column vector of the i-th component of the target vector responses ($y_{ij} \in \{-1, +1\}$).

Classification. To classify an ECG segment, the composite feature vector $\mathbf{f_c}$ is computed and used to compute the vector $\mathbf{y}$ by means of equation (7.14). The signs of the values y_i are used to actually classify the ECG, by means of the tree given in Figure 7.7.

The structure of the tree depends on the fact that there is a multiple dichotomy of six classes of samples. The Euclidean center distance between

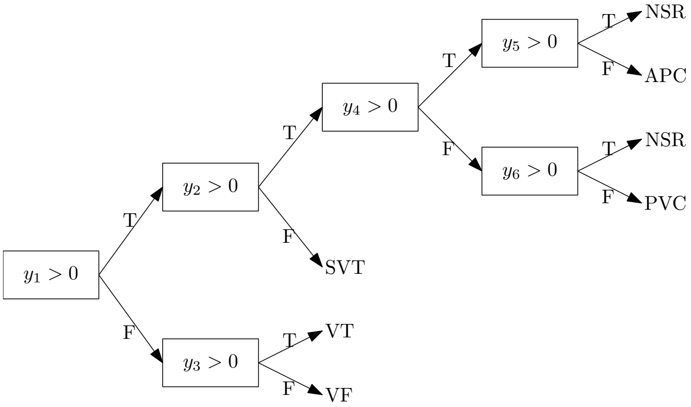


Figure 7.7: The decision graph leading to ECG segment classification. Given the array $y_1, \ldots, y_6$, the tree is traversed according to the result of the comparison of the values with 0 in each node, following the true (T) or false (F) edges.

these classes were computed for determining the groupings of classes at each stage, obtaining that APC/NSR/PVC, VT/VF and SVT form one group respectively due to small values of the Euclidean center distance within the same group and large values between different groups [ASSK07, ch.8]. Therefore, VT/VF was separated from APC/NSR/PVC and SVT in stage one (y_1). Similarly, in the second stage (y_2), VT and VF were differentiated. In the third stage (y_3), SVT was distinguished from NSR, APC and PVC. In the later stages (y_4, y_5, y_6), NSR, APV and PVC were distinguished from each other and classified.

7.2.2 Classification results

The classification results on test data provided by Ge et al. in [ASSK07, ch.8] are given in Table 7.2. The authors selected 200 hearth beats for each class and used 60 of them for training, the others for testing. The mean classification accuracy was 96.1%.

We implemented the protocol proposed by Ge et al. and, since their data set was not available, in our experiments we used a different dataset obtained by signals still coming from PhysioBank [GAG$^+$00]. The set is built by se-

Testing data set		NSR	APC	PVC	SVT	VT	VF	Accuracy result
140	NSR	135	3	1	0	1	0	96.4%
140	APC	8	131	0	1	0	0	93.5%
140	PVC	4	0	133	2	1	0	95.0%
140	SVT	0	1	2	137	0	0	97.9%
140	VT	0	0	0	0	136	4	97.1%
140	VF	0	0	1	0	4	135	96.4%

Table 7.2: QDF-based classification results (results from [ASSK07, ch.8]).

lecting 200 samples of each class from 1 or at most 2 patterns. From these samples, we selected 60 samples at random for each class selected for the training set and we used the remaining are used for the test set. This set provides performance lower than that illustrated in [GSK02], anyway our goal is not to reproduce similar results, but to demonstrate that a s.p.e.d. implementation of the protocol can provide results comparable to a plain implementation. The algorithm we developed was able to classify correctly 88.3% of the test set. Table 7.3 shows the confusion matrix obtained. We will refer to this data set in our privacy preserving implementations described in the next chapters.

Testing data set		NSR	APC	PVC	SVT	VT	VF	Accuracy result
140	NSR	132	8	0	0	0	0	94.3%
140	APC	22	118	0	0	0	0	84.3%
140	PVC	0	2	138	0	0	0	98.6%
140	SVT	0	0	0	140	0	0	100.0%
140	VT	0	0	0	0	100	40	71.4%
140	VF	0	0	0	0	26	114	81.4%

Table 7.3: QDF-based classification results obtained in our tests.

Chapter 8

Privacy preserving Classification by using Linear Branching Programs

An important contribution of this thesis is given by the definition of Linear Branching Programs (LBP), together with a cryptographic protocol for its efficient secure evaluation [BFK+09a, BFL+09, BFL+11]. The notion of LBP is a natural generalization of binary classification trees and Ordered Binary Decision Diagrams (OBDDs). Compared to the above, LBPs have a more general branching condition that depends on the comparison of a linear combination of the inputs with a threshold.

Section 8.1 introduces Linear branching programs, together with two different implementations and their efficiency comparison. Then in Section 8.2 we apply LBP to ECG classification, showing the results obtained in terms of efficiency and classification accuracy.

8.1 Linear Branching Programs (LBP)

Before describing our implementations of the secure ECG classifier, we formally define the notion of linear branching programs. We do so by generalizing the BP definition used in [BPSW07]. We note that BPs – and hence also LBPs – generalize binary classification or decision trees and Ordered Binary Decision Diagrams (OBDDs) used in [KJGB06, Sch08].

Definition 2 (Linear Branching Program)**.** *Let* $\mathbf{x}^{(\ell)} = [x_1^{(\ell)}, .., x_n^{(\ell)}]$ *be the attribute vector of signed* ℓ*-bit integer values. A binary* **Linear Branching Program (LBP)** $\mathcal{L}$ *is a triple* $\langle \{P_1, .., P_z\}, \mathit{Left}, \mathit{Right} \rangle$*. The first element is a set of* z *nodes consisting of* d *"decision nodes"* $P_1, .., P_d$ *followed by* $z - d$ *"classification nodes"* $P_{d+1}, .., P_z$*.*
Decision nodes P_i*,* $1 \le i \le d$ *are the internal nodes of the LBP. Each* $P_i :=$

$\left\langle \mathbf{a_i}^{(\ell)}, t_i^{(\ell')} \right\rangle$ is a pair, where $\mathbf{a_i}^{(\ell)} = [a_{i,1}^{(\ell)}, .., a_{i,n}^{(\ell)}]$ is the linear combination vector consisting of n signed ℓ-bit integer values and $t_i^{(\ell')}$ is the signed ℓ'-bit integer "threshold" value with which $\mathbf{a_i}^{(\ell)} {\mathbf{x}^{(\ell)}}^\top = \sum_{j=1}^{n} a_{i,j}^{(\ell)} x_j^{(\ell)}$ is compared in the node. Left(i) is the index of the next node if $\mathbf{a_i}^{(\ell)} {\mathbf{x}^{(\ell)}}^\top \leq t_i^{(\ell')}$; Right$(i)$ is the index of the next node if $\mathbf{a_i}^{(\ell)} {\mathbf{x}^{(\ell)}}^\top > t_i^{(\ell')}$. Functions Left() and Right() are such that the resulting directed graph is acyclic.
Classification nodes $P_j := \langle c_j \rangle$, $d < j \leq z$ are the leaf nodes of the LBP consisting of a single classification label c_j each.

To evaluate the LBP $\mathcal{L}$ on an attribute vector $\mathbf{x}^{(\ell)}$, we start with the first decision node P_1. If $\mathbf{a_1}^{(\ell)} {\mathbf{x}^{(\ell)}}^\top \leq t_1^{(\ell')}$, move to node *Left*(1), else to *Right*(1). We then repeat this process recursively (with corresponding $\mathbf{a_i}^{(\ell)}$ and $(t_i^{(\ell')})$), until we reach one of the classification nodes and obtain the classification $c = \mathcal{L}(\mathbf{x}^{(\ell)})$.

In the general case of LBPs, the bit-length ℓ' of the threshold values $t_i^{(\ell')}$ has to be chosen according to the maximum value of the linear combinations:

$$abs(\mathbf{a_i}^{(\ell)} {\mathbf{x}^{(\ell)}}^\top) = abs(\sum_{j=1}^{n} a_{i,j}^{(\ell)} x_j^{(\ell)}) \leq \sum_{j=1}^{n} 2^{2(\ell-1)} = n 2^{2(\ell-1)}$$
$$\Rightarrow \ell' = 1 + \lceil \log_2(n 2^{2(\ell-1)}) \rceil = 2\ell + \lceil \log_2 n \rceil - 1. \tag{8.1}$$

As noted above, LBPs can be seen as a generalization of previous representations:

- *Branching Programs (BP)* as used in [BPSW07] are a special case of LBPs where in each node only an element of the attribute vector is compared with the threshold. In detail in each decision node P_i of the BP, only the α_i-th input $x_{\alpha_i}^{(\ell)}$ is compared with the threshold value $t_i^{(\ell')}$, where $\alpha_i \in \{0, .., n\}$ is a private index. In this case, the linear combination vector $\mathbf{a_i}^{(\ell)}$ of the LBP decision node degrades to a *selection vector* $\mathbf{a_i} = [a_{i,1}, .., a_{i,n}]$, with exactly one entry $a_{i,\alpha_i} = 1$ and all other entries $a_{i,j \neq \alpha_i} = 0$. The bit-length of the threshold values $t_i^{(\ell')}$ is set to $\ell' = \ell$.

- *Ordered Binary Decision Diagrams (OBDD)* as used in [KJGB06, Sch08] are a special case of BPs with bit inputs ($\ell = 1$) and exactly two classification nodes ($P_{z-1} = \langle 0 \rangle$ and $P_z = \langle 1 \rangle$).

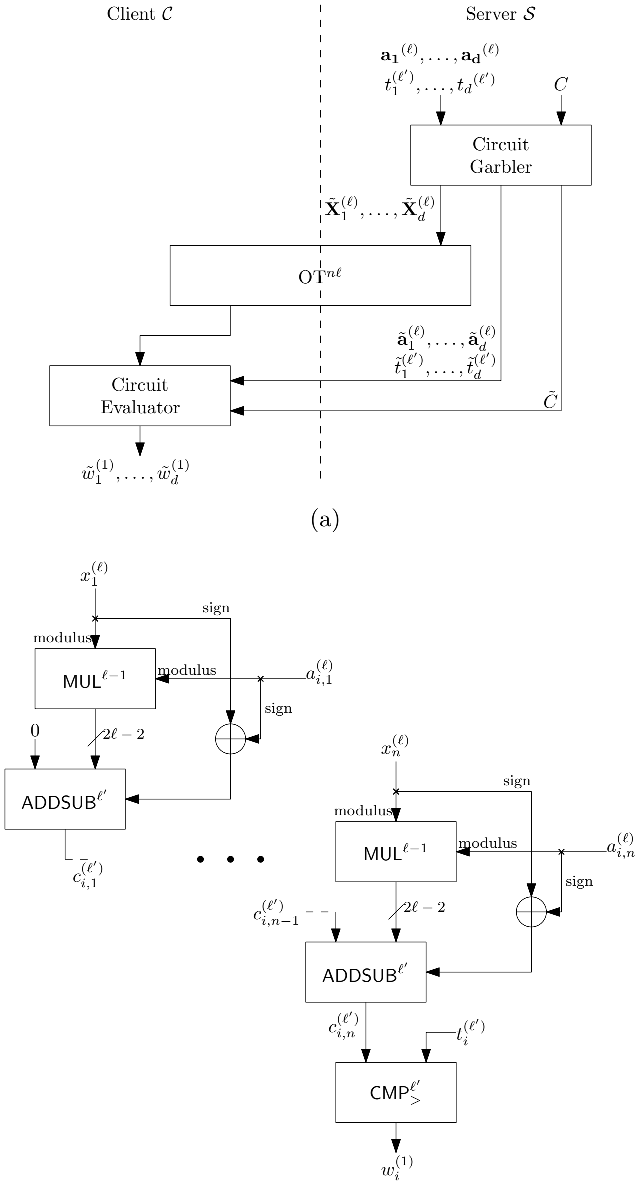
We here address two possible implementations for a generic LBP in the encrypted domain: a full-GC protocol and a hybrid protocol. Considering the high number of bits needed for the representation of the scalar products and that it is used in a comparison function, we discarded a full-HE implementation a priori.

8.1.1 Full-GC implementation

A straightforward instantiation can be obtained by evaluating a garbled circuit whose size depends on the number of attributes n. The linear branching program can be instantiated based on the secure evaluation of a garbled circuit as shown in Figure 8.1(a). First, $\mathcal{S}$ creates a boolean circuit C with ℓ-bit inputs $x_1^{(\ell)}, \ldots, x_n^{(\ell)}$ and output bits $w_1, \ldots, w_d$ that obliviously computes the intended functionality as described below.

The circuit is evaluated securely with Yao's garbled circuit protocol, i.e., $\mathcal{S}$ creates a garbled circuit $\tilde{C}$ which is sent to $\mathcal{C}$ along with the garbled inputs corresponding to $\mathcal{C}$'s inputs $x_1^{(\ell)}, .., x_n^{(\ell)}$ in a $OT_t^{n\ell}$ protocol, and finally $\mathcal{C}$ evaluates $\tilde{C}$ on these garbled inputs to obtain the garbled output values $\tilde{w}_1, \ldots, \tilde{w}_d$.

The circuit C needs to compute $w_i = (\mathbf{a_i}^{(\ell)} \mathbf{x}^{(\ell)^\top} > t_i^{(\ell')}) = (\sum_{j=1}^{n} a_{i,j}^{(\ell)} \cdot x_j^{(\ell)} > t_i^{(\ell')})$, $1 \leq i \leq d$. As shown in Figure 8.1(b), an efficient circuit construction can be obtained by first multiplying the magnitudes of $x_j^{(\ell)}$ and $a_{i,j}^{(\ell)}$ with an unsigned integer multiplication circuit ($\mathsf{MUL}^{\ell-1}$). Afterwards, the sign is determined by combining the sign bits of $x_j^{(\ell)}$ and $a_{i,j}^{(\ell)}$ with an XOR gate ($\oplus$). Depending on this sign, the multiplied value is added or subtracted from the intermediate result $c_{i,j-1}^{(\ell')}$ with an integer addition/subtraction circuit ($\mathsf{ADDSUB}^{\ell'}$). Hence, the intermediate values $c_{i,J}^{(\ell')}$ carry the sum of the first J summands, i.e., $c_{i,J}^{(\ell')} = \sum_{j=1}^{J} a_{i,j}^{(\ell)} \cdot x_j^{(\ell)}$. In the end, the final value $c_{i,n}^{(\ell')} = \sum_{j=1}^{n} a_{i,j}^{(\ell)} \cdot x_j^{(\ell)}$ is compared with the threshold value $t_i^{(\ell')}$ using an integer comparison circuit ($\mathsf{CMP}_{>}^{\ell'}$). Finally a d-input gate is used to provide the final output given the results of the d comparisons. The resulting garbled circuit


Client $\mathcal{C}$
Server $\mathcal{S}$
$\mathbf{a_1}^{(\ell)}, \ldots, \mathbf{a_d}^{(\ell)}$
$t_1^{(\ell')}, \ldots, t_d^{(\ell')}$
C
Circuit Garbler
$\tilde{\mathbf{X}}_1^{(\ell)}, \ldots, \tilde{\mathbf{X}}_d^{(\ell)}$
$\mathrm{OT}^{n\ell}$
$\tilde{\mathbf{a}}_1^{(\ell)}, \ldots, \tilde{\mathbf{a}}_d^{(\ell)}$
$\tilde{t}_1^{(\ell')}, \ldots, \tilde{t}_d^{(\ell')}$
Circuit Evaluator
$\tilde{C}$
$\tilde{w}_1^{(1)}, \ldots, \tilde{w}_d^{(1)}$
$x_1^{(\ell)}$
sign
modulus
$\mathsf{MUL}^{\ell-1}$
modulus
$a_{i,1}^{(\ell)}$
sign
0
$2\ell-2$
$\mathsf{ADDSUB}^{\ell'}$
$c_{i,1}^{(\ell')}$
$x_n^{(\ell)}$
sign
modulus
$\mathsf{MUL}^{\ell-1}$
modulus
$a_{i,n}^{(\ell)}$
sign
$c_{i,n-1}^{(\ell')}$
$2\ell-2$
$\mathsf{ADDSUB}^{\ell'}$
$c_{i,n}^{(\ell')}$
$t_i^{(\ell')}$
$\mathsf{CMP}_{>}^{\ell'}$
$w_i^{(1)}$

(a)

(b)

Figure 8.1: Full-GC solution (a) and Linear Selection Circuit (part of C) of a node (b).

has size $|\tilde{C}| = d(n(|\mathsf{MUL}^{\ell-1}|+|\oplus|+|\mathsf{ADDSUB}^{\ell'}|)+|\mathsf{CMP}_{>}^{\ell'}|) = d(n([2(\ell-1)^2-(\ell-1)]+[0]+[\ell'])+\ell'+1) = 2nd\ell^2-5nd\ell+3nd+nd\ell'+d\ell' \approx 2nd\ell^2+nd\ell'$ non-XOR 2-input gates and a d-input gate.

The circuit-based instantiation of LBP needs the same number of rounds as the underlying OT protocol since the garbled circuit can be sent with the last message of the OT protocol. The asymptotic communication complexity of the full-GC protocol is that of the $OT_t^{n\ell}$ protocol (OT) plus the size of the garbled circuit given above (GC). Considering only the precomputation of OT and transmitting the circuit during the online phase, we obtain the complexity shown in Table 8.1.

BP. In case of BPs, a substantially smaller circuit C' can be constructed to compute the functionality $w_i = (x_{\alpha_i}^{(\ell)} > t_i^{(\ell)})$. This circuit first obliviously selects the input $x_{\alpha_i}^{(\ell)}$ from the inputs $x_1^{(\ell)}, .., x_n^{(\ell)}$. This can be achieved by using selection blocks S_d^n as follows: An S_d^n selection block is a circuit which can obliviously select for each of its d outputs any of its n inputs. By using ℓ such selection blocks in parallel (one for each bit of the ℓ bits), the circuit can obliviously select $x_{\alpha_1}^{(\ell)}, \dots, x_{\alpha_d}^{(\ell)}$ from $x_1^{(\ell)}, \dots, x_n^{(\ell)}$. Afterwards, the selected values are compared with the respective threshold $t_i^{(\ell)}$ using an integer comparison circuit ($\mathsf{CMP}_{>}$). Using the efficient selection block constructions of [KS08b] together with the comparison blocks implemented in [PSS09] this results in a garbled circuit of size $|\tilde{C}'|_{bits} = \ell|S_d^n| + d|\mathsf{CMP}_{>}^{\ell}| = (\ell[4(n+3d)\lceil\log d\rceil + 4n - 16d + 12] + d[4\ell])t \approx 4(n\log d + 3d\log d)\ell t$ bits.

8.1.2 Hybrid implementation

In this instantiation of LBP (see Protocol 24 for an overview), an hybrid protocol is used to implement the LBP. The scalar product is computed by using an homomorphic protocol while comparison is computed by using GC.

$\mathcal{C}$ generates a key-pair for the additively homomorphic encryption scheme and sends the public key $pk_{\mathcal{C}}$ together with the homomorphically encrypted attributes $[\![x_1^{(\ell)}]\!], \dots, [\![x_n^{(\ell)}]\!]$ to $\mathcal{S}$. Using the additively homomorphic property, $\mathcal{S}$ can compute the linear combination of these ciphertexts with the private coefficients $\mathbf{a_i}^{(\ell)}$ as $[\![y_i^{(\ell')}]\!] = [\![\sum_{j=1}^{n} a_{i,j}^{(\ell)} x_j^{(\ell)}]\!]$, $1 \le i \le d$.

Afterwards, the encrypted values $[\![y_i^{(\ell')}]\!]$ are obliviously compared with the

HYBRID LBP PROTOCOL

inputs of $\mathcal{C}$: $\mathbf{x} = [x_1^{(\ell)}, \ldots, x_n^{(\ell)}]$
inputs of $\mathcal{S}$: $\mathbf{a_i} = [a_{i,1}^{(\ell)}, \ldots, a_{i,n}^{(\ell)}], t_i^{(\ell')}$; $i = 1 \ldots d$
output for $\mathcal{C}$: classification result
output for $\mathcal{S}$: nothing

client $\mathcal{C}$ server $\mathcal{S}$

encrypts $x_j^{(\ell)}$ $\forall j \in 1 \ldots n$;

$$\xrightarrow{\llbracket x_j^{(\ell)} \rrbracket \ \forall j \in 1 \ldots n}$$

(scalar products:)
$\llbracket y_i^{(\ell')} \rrbracket = \llbracket \sum_{j=1}^n a_{i,j}^{(\ell)} x_j^{(\ell)} \rrbracket =$
$= \prod_{j=1}^n \llbracket x_j^{(\ell)} \rrbracket^{a_{i,j}^{(\ell)}}, 1 \leq i \leq d;$
(packing:)
$\llbracket y \rrbracket = \llbracket \sum_{i=1}^d 2^{\ell'(i-1)} (y_i^{\ell'} + 2^{\ell'-1}) \rrbracket =$
$= \prod_{i=1}^d (\llbracket 2^{\ell'-1} \rrbracket \llbracket y_i^{\ell'} \rrbracket)^{2^{\ell'(i-1)}}$ (blinding:)
chooses $r \in_R \mathbb{Z}_N$;
$\llbracket z \rrbracket = \llbracket y \rrbracket \llbracket r \rrbracket = \llbracket x + r \rrbracket$;

$$\xleftarrow{\llbracket z \rrbracket}$$

decrypts $\llbracket z \rrbracket$;
$\downarrow z_{d\ell'-1}, \ldots, z_0$ $\downarrow r_{d\ell'-1}, \ldots, r_0$ $\downarrow t_1^{(\ell')} \ldots t_d^{(\ell')}$

GC($\mathsf{SUB}^{d\ell'}$, d $\mathsf{CMP}^{\ell'}$, d-input gate)

$\downarrow$ Classification result

Protocol 24: Hybrid LBP protocol. For simplicity we assume that all the y_i values can be packed in a single cyphertext.

thresholds $t_i^{(\ell')}$ in a comparison protocol. This protocol allows $\mathcal{C}$ to obliviously obtain the garbled values corresponding to the comparison of $y_i^{(\ell')}$ and $t_i^{(\ell')}$, i.e., $\tilde{w}_i^0$ if $y_i^{(\ell')} \leq t_i^{(\ell')}$ and $\tilde{w}_i^1$ otherwise. The secure comparison has to ensure that neither $\mathcal{C}$ nor $\mathcal{S}$ learns anything about the plaintexts $y_i^{(\ell')}$ from which they could deduce information about the other party's private function or inputs. Hence a "HE to GC" interface is needed. $\mathcal{S}$ blinds the encrypted value $\llbracket y_i^{\ell'} \rrbracket$ in order to hide the encrypted plaintext from $\mathcal{C}$ obtaining $\llbracket \gamma_i \rrbracket$. To achieve this,

$\mathcal{S}$ adds a randomly chosen value $R \in_R \mathbb{Z}_N$ [1] under encryption before sending them to $\mathcal{C}$ who can decrypt but does not learn the plain value. Afterwards, a garbled circuit C is evaluated which obliviously takes off the blinding value R and compares the result (which corresponds to $y_i^{(\ell')}$) with the threshold value $t_i^{(\ell')}$. We improve the communication complexity of this basic protocol which essentially corresponds to the protocol of [BPSW07] with the following two technical tricks:

Packing. Usually, the plaintext space of the HE scheme $\mathbb{Z}_N$ is substantially larger than the encrypted values $y_i^{(\ell')}$. Hence, multiple encryptions, say d', can be packed together into one ciphertext before blinding and sending it to $\mathcal{C}$. This reduces the communication complexity and the number of decryptions that need to be performed by $\mathcal{C}$ by a factor of d'. For this, the encrypted values $-2^{\ell'-1} < y_i^{\ell'} < 2^{\ell'-1}$ are shifted into the positive range $(0, 2^{\ell'})$ first by adding $2^{\ell'-1}$ and afterwards are concatenated by computing $[\![y]\!] = [\![\sum_{i=1}^{d'} 2^{\ell'(i-1)}(y_i^{\ell'} + 2^{\ell'-1})]\!] = \prod_{i=1}^{d'}([\![2^{\ell'-1}]\!][\![y_i^{\ell'}]\!])^{2^{\ell'(i-1)}}$. The packed ciphertext $[\![y]\!]$ encrypts a $L' = d'\ell'$ bit value now.

Minimizing Circuit Size. As described before, the garbled circuit obliviously takes off the blinding value R. Instead of computing in the plaintext space of the homomorphic cryptosystem $\mathbb{Z}_N$, it is beneficial to compute over integers since circuit sizes are substantially smaller (inspired by the BITREP gate of [ST06b]). For this, we ensure that no overflow occurs when blinding the L'-bit value y by adding $R \in_R \mathbb{Z}_N$. This can be achieved by choosing L' such that it is κ bits less than the bitlength of N, where κ is a statistical correctness parameter (e.g., $\kappa = 80$): $L' \leq T - \kappa$. Now, an overflow occurs only if the κ topmost bits of R are all ones which – as R was chosen randomly – occurs with probability $2^{-\kappa}$ which is negligible in κ. The circuit subtracts the lowest L' bits of R from those of γ to obliviously reconstruct the value y before comparing it componentwise with the thresholds. Then, each $y_i^{\ell'}$, $1 \leq i \leq d$ is compared with its corresponding threshold $t_i^{\ell'}$ with an integer comparison circuit ($\mathsf{CMP}_>$). Finally a d-input gate is used to provide

---

[1] We choose R from the full plaintext space in order to protect against malicious behavior of $\mathcal{C}$, instead than $\{0,1\}^{\ell'+\kappa}$.

Classification Protocol	Private Function	Moves	Asymptotic Communication Complexity		
			GC	OT	HE
[BPSW07]	BP	4	$12z\ell(t+\kappa)$	$OT_t^{z\ell}$	$(n+z)2T$
	LBP		$12z\ell'(t+\kappa)$	$OT_t^{z\ell'}$	
Full-GC	BP	2	$4(n\log d + 3d\log d)\ell t$	$OT_t^{n\ell}$	
	LBP		$(2nd\ell^2 + nd\ell')4t$		
Hybrid	BP	4	$8d\ell t$	$OT_t^{d\ell}$	$(n + \frac{\ell}{T-\kappa}d)2T$
	LBP		$8d\ell' t$	$OT_t^{d\ell'}$	$(n + \frac{\ell'}{T-\kappa}d)2T$

Table 8.1: Protocols for Secure Evaluation of Private BPs/LBPs with parameters z: #nodes, d: #decision nodes, n: #attributes, ℓ: bitlength of attributes, ℓ': bitlength of thresholds (for LBPs), t: symmetric security parameter, T: asymmetric security parameter, κ: statistical correctness parameter.

the output given the results of the d comparisons. The circuit can be generated automatically with the compiler of [PSS09] into a garbled circuit of size $|\tilde{C}| = |\mathsf{SUB}^{L'}| + d'|\mathsf{CMP}_{>}^{\ell'}| = (L' + d'(\ell')) \approx 2L'$ non-XOR 2-input gates and 1 d-input gate.

Complexity. From $L' = d'\ell' \leq T - \kappa$ we can infer $d' = \frac{T-\kappa}{\ell'}$ as the maximum number of packed ciphertexts. The comparison protocol needs to be run for d inputs which can be achieved by running the comparison protocol with d' inputs $\lceil \frac{d}{d'} \rceil$ times in parallel. The asymptotic communication complexity of the hybrid LBP protocol as shown in Table 8.1 consists of $n + \frac{d}{d'} = n + \frac{\ell'}{T-\kappa}d$ Paillier ciphertexts of size $2T$ bits each (HE), garbled circuits of size $\frac{d}{d'} \cdot 8L't + 2^d t = 8d\ell' t + 2^d t \approx 8d\ell' t$ bits (GC), and a $\frac{d}{d'} \cdot L' = d\ell'$ parallel OT protocol $OT_t^{d\ell'}$. The number of moves is two for sending the homomorphic encryptions plus those of the underlying OT protocol ($\tilde{C}$ can be sent with the last message of the OT protocol).

8.2 Application of LBP to ECG classification

Before describing the privacy-preserving ECG classification protocol, we define the players of the protocol and the data that needs to be protected. A first requirement is that the server $\mathcal{S}$, who is running the classification algorithm on client's ECG signal, learns neither information about the ECG signal nor the final result of the classification. At the same time, the client $\mathcal{C}$ should not get any information about the algorithm used by $\mathcal{S}$, except for the output of the classification. The latter point deserves some explanation. We assume that the general form of the classifier used by $\mathcal{S}$ is known, however the parameters of the classifier need to be kept secret. By referring to the description given in Chapter 7, the algorithm parameters that $\mathcal{S}$ aims at keeping secret are the matrix $\mathbf{B}$ and the classification tree of Figure 7.7. This is a reasonable assumption since the domain specific knowledge needed to classify the ECGs and the knowledge got from the training, a knowledge that $\mathcal{S}$ may want to protect, reside in the classification tree and the matrix $\mathbf{B}$.

In order to introduce the privacy-preserving ECG classifier, we observe that the classification algorithm based on the QDF functions and the classification tree (described in Chapter 7) is nothing but an LBP with $\mathbf{f}^{\mathbf{c},(\ell)}$ as attribute vector, and 6 nodes $P_i = \left\langle \boldsymbol{\beta}_i^{(\ell)}, 0 \right\rangle, i = 1, .., 6$, where $\mathbf{f}^{\mathbf{c},(\ell)}$ and $\boldsymbol{\beta}_i^{(\ell)}$ are ℓ-bit representations of the features and projection vectors. In this way, the general scheme for the privacy-preserving implementation of the classifier assumes the form given in Figure 8.2. The generic $\mathbf{x}^{(\ell)}$, ${\mathbf{a_i}}^{(\ell)}$, $t_i^{(\ell')}$ are replaced here by the specific ${\mathbf{f^c}}^{(\ell)}$, ${\boldsymbol{\beta}_\mathbf{i}}^{(\ell)}$, 0 respectively.

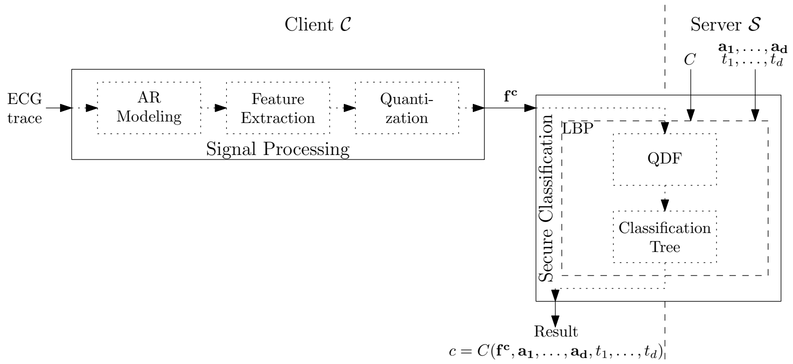


Figure 8.2: Privacy-preserving ECG diagnosis.

All steps until the computation of the composite feature vector are performed by $\mathcal{C}$ on the plain data. Such a choice does not jeopardize the security of the system from the server's point of view, since $\mathcal{S}$ is not interested in keeping the structure of the classifier secret, but only in preventing users from knowing the matrix $\mathbf{B}$ and the classification tree. On the contrary, all the steps from the projection onto the directions $\boldsymbol{\beta}_i$'s, through the final classification are carried out securely. Note that with respect to the overall architecture depicted in Figure 7.4, we added a quantization step before the encryption of the composite feature vector. The need for such a step stems from the observation that the parameters $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ resulting from the AR model estimation procedure are usually represented as floating point numbers, a representation that is not suitable for s.p.e.d. protocols which can compute on numbers represented as integers only. For this reason the elements of the composite feature vector $\mathbf{f^c}$ are quantized and represented in integer arithmetic for subsequent processing[2]. Note that the choice of the quantization step, and consequently the number of bits used to represent the data (ℓ in the LBP terminology), is crucial since on one side it determines the complexity of the overall secure protocol and on the other side it has an impact on the accuracy of the ECG classification.

Differently from the protocol described in Chapter 7, we omit the sixth feature obtaining the feature vector $\mathbf{f} = [\alpha_1, \alpha_2, \alpha_3, \alpha_4, n_1]$ and hence the composite feature vector $\mathbf{f^c}$ is composed by 21 values. This omission is due to the fact that the feature removed can be expressed as $n_2 = 300 - n_1$, where 300 is the number of samples in each chunk, and in the QDFs each product involving n_2 (its square values and its products with other features) can be expressed as a function of n_1.

8.2.1 Quantization error analysis

In this section we estimate the impact that the quantization error introduced passing from $\mathbf{f^c}$ to $\mathbf{f}^{\mathbf{c},(\ell)}$ and from $\boldsymbol{\beta}_i$ to $\boldsymbol{\beta}_i^{(\ell)}$ has on the classification accuracy. Such an analysis will be used to determine the minimum number of bits (ℓ) needed to represent the attribute vector and the linear combination vectors of the LBP. The value of ℓ influences the complexity of the secure classification protocol for two main reasons. As already outlined in Chapter 3 and Chapter 5, the main ingredients of the protocols for secure evaluation of private LBPs are garbled circuits and additively homomorphic encryption.

---

[2] In the same way the coefficients of matrix $\mathbf{B}$, representing the combination vectors of the LBP, are represented as integer numbers.

In the case of garbled circuits, the input of the protocol are the single bits used to represent $\mathbf{f}^{\mathbf{c},(\ell)}$ and $\boldsymbol{\beta}_i^{(\ell)}$. It is obvious, then, that the greater the number of bits, the more complex the resulting protocol will be. With regard to computing on homomorphically encrypted data, we observe that after each multiplication carried out in the encrypted domain, the number of bits necessary to represent the output of the multiplication increases[3] (it approximately doubles). Since it is not possible to carry out truncations in the encrypted domain, it is necessary that the size of the ring used by the homomorphic cryptosystem is large enough to contain the output of the computations without an overflow which would cause an invalid result. Augmenting the number of bits used to represent the inputs of the LBP may require to increase the size of the needed cryptosystem ring which results in an increased protocol complexity.

To start with, we observe that quantization is applied to the composite feature vector $\mathbf{f}^{\mathbf{c}}$, that is used to compute the vector $\mathbf{y}$, through multiplication with the matrix $\mathbf{B}$. After such a step, only the signs of vector $\mathbf{y}$ are retained, hence it is sufficient to analyze the effect of quantization until the computation of the sign of $\mathbf{y}$. As to the processing steps carried out by the client prior to quantization, we assume that all the blocks until QDF are carried out by using a standard double precision floating point arithmetic. In order to simplify the notation, we consider only the computation of one coefficient of the vector $\mathbf{y}$. The function to be computed is a simple inner product: $y = \mathbf{f}^{\mathbf{c}}, \boldsymbol{\beta}^\top = \sum_j \beta_j f_j^c$ where the index i has been omitted, and β_j and f_j^c are real numbers. The quantized version of the above relationship can be expressed as follows:

$$\begin{aligned} \beta_{q,j} &= \rho_1 \beta_j + \varepsilon_{1,j} = \lfloor \rho_1 \beta_j \rceil \\ f_{q,j}^c &= \rho_2 f_j^c + \varepsilon_{2,j} = \lfloor \rho_2 f_j^c \rceil \end{aligned} \tag{8.2}$$

where ρ_1 and ρ_2 are positive integers and $\varepsilon_{1,j}$ and $\varepsilon_{2,j}$ are the quantization errors affecting $\beta_{q,j}$ and $f_{q,j}^c$ respectively. By using the above relations it is

---

[3] The same observation holds for additions, however additions have a negligible effect with respect to multiplications.

possible to evaluate the final error due to quantization:

$$\begin{aligned}
&\sum_{j=0}^{N-1}\left(\rho_1\beta_j+\varepsilon_{1,j}\right)\left(\rho_2 f_j^c+\varepsilon_{2,j}\right)= \\
=\ &\sum_{j=0}^{N-1}\rho_1\rho_2\beta_j f_j^c+\rho_1\beta_j\varepsilon_{2,j}+\rho_2 f_j^c\varepsilon_{1,j}+\varepsilon_{1,j}\varepsilon_{2,j}= \\
=\ &\rho_1\rho_2\Bigg(y+\underbrace{\sum_{j=0}^{N-1}\frac{\beta_j\varepsilon_{2,j}}{\rho_2}+\sum_{j=0}^{N-1}\frac{f_j^c\varepsilon_{1,j}}{\rho_1}+\sum_{j=0}^{N-1}\frac{\varepsilon_{1,j}\varepsilon_{2,j}}{\rho_1\rho_2}}_{\varepsilon}\Bigg) \qquad (8.3)
\end{aligned}$$

where ε indicates the error on the scalar product once the scaling factor $\rho_1\rho_2$ is canceled out. By letting $\max(|\beta_j|)=M_b$, $\max(|f_j^c|)=M_f$ and by noting that $\max(|\varepsilon_{1,j}|)=\max(|\varepsilon_{2,j}|)=\frac{1}{2}$, we have:

$$\varepsilon\leq\frac{N}{2\rho_1\rho_2}\left(\rho_1 M_b+\rho_2 M_f+\frac{1}{2}\right)\leq\varepsilon^* \qquad (8.4)$$

where ε^* is a target maximum error that we do not want to exceed. Given ε^*, choosing the optimum values of ρ_1 and ρ_2 is equivalent to a constrained minimization problem in which the function to be minimized is $\rho_1\rho_2$ (since this is equivalent to minimize the number of bits necessary to represent the output of the scalar product) and the constraint corresponds to equation (8.4), that is:

$$\rho_1\geq\frac{N(2\rho_2 M_f+1)}{4\rho_2\varepsilon^*-2NM_b}. \qquad (8.5)$$

To ensure that ρ_1 is a positive integer, we require $2\rho_2\varepsilon^*-NM_b>0$, yielding the following minimization problem:

$$\min_{\rho_2>\frac{NM_b}{2\varepsilon^*}}\ \rho_2\frac{N(2\rho_2 M_f+1)}{4\rho_2\varepsilon^*-2NM_b}. \qquad (8.6)$$

By solving (8.6) we obtain the solutions:

$$\begin{aligned}
\rho_2 &= \frac{1}{2M_f\varepsilon^*}\left(NM_bM_f+\sqrt{NM_bM_f(\varepsilon^*+NM_bM_f)}\right), \qquad (8.7)\\
\rho_1 &= \frac{1}{2M_b\varepsilon^*}\left(NM_bM_f+\sqrt{NM_bM_f(\varepsilon^*+NM_bM_f)}\right). \qquad (8.8)
\end{aligned}$$

Special Case: $M_b = M_f$ We now consider the case in which $\boldsymbol{\beta}$ and $\mathbf{f^c}$ are represented with the same number of bits, that is we let $\max(|\beta_j|) = \max(|f_j^c|) = M$ yielding:

$$\rho_1 = \rho_2 = \rho = \frac{1}{2\varepsilon^*}\left(NM + \sqrt{N(\varepsilon^* + NM^2)}\right). \tag{8.9}$$

At this point we are ready to define the size of the ring usedto represent the composite feature vector and the matrix $\mathbf{B}$, in fact the maximum y that can be obtained from (8.3) is

$$\max(|y|) = \rho^2 M^2 N + \rho MN + \frac{N}{2} \tag{8.10}$$

so it is necessary to use a ring $\mathbb{Z}_n$ with n at least:

$$n \geq 2\left(\rho^2 M^2 N + \rho MN + \frac{N}{2}\right) \tag{8.11}$$

The ring size is a function of M, N and ε^*, that is:

$$\begin{aligned} n &\geq 2\left(\rho^2 M^2 N + \rho MN + \frac{N}{2}\right) \\ &= 2\left[\left(\frac{1}{2\varepsilon^*}\left(NM + \sqrt{N(\varepsilon^* + NM^2)}\right)\right)^2 M^2 N \right. \\ &\quad \left. + \left(\frac{1}{2\varepsilon^*}\left(NM + \sqrt{N(\varepsilon^* + NM^2)}\right)\right) MN + \frac{N}{2}\right] \\ &= \frac{N}{2(\varepsilon^*)^2}\left[2\varepsilon^* + M^2 N(2M^2 N + 3\varepsilon^*)\right. \\ &\quad \left. + (2M^3 N + 2M\varepsilon^*)\sqrt{N(\varepsilon^* + M^2 N)}\right] \\ &\approx \frac{2M^4 N^3}{(\varepsilon^*)^2} \end{aligned} \tag{8.12}$$

In our case $N = 21$, however the value of M is not known. In principle, in fact, the parameters of the AR model are not bounded[4] and hence M should be ∞. Of course this is not an acceptable solution, all the more that

---

[4] The solutions of the linear system $\mathbf{Ra} = \mathbf{r}$ grows without any bound as the determinant of the matrix $\mathbf{R}$ tends to zero.

in practical applications the AR model coefficient for ECG signals are rather small (lower than 10 in most cases). It is then reasonable to assume that the maximum value of the composite feature vector is determined by the features n_1 whose maximum value is 300 (i.e., the number of samples of each an ECG sequence corresponding to one heart beat), yielding $M_f = 90000$ (recall that $\mathbf{f^c}$ contains the squared valued of the features $\mathbf{f}$). A similar argument holds for the elements of the matrix $\mathbf{B}$, a rigorous upper bound does not exist, however in practice very large values are never encountered, hence we can safely set $M_b = M_f = M = 90000$. By inserting the above values within equation (8.12), we obtain the values of n as a function of ε^* as plotted in Figure 8.3. Some numerical results for typical values of ε^* are given in Table 8.2.

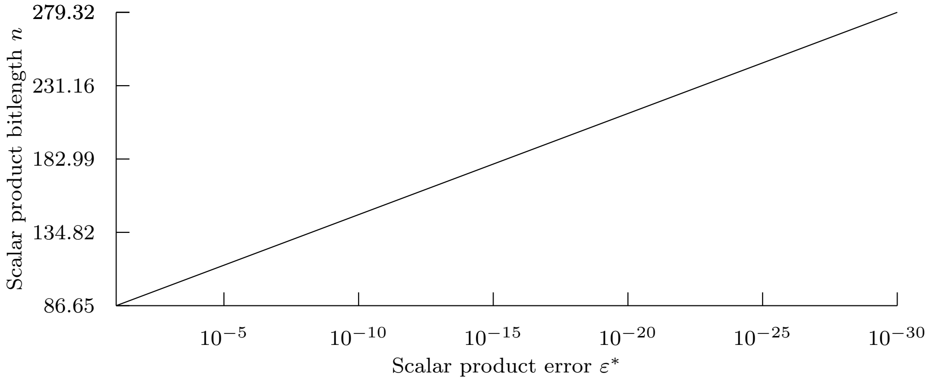


Figure 8.3: Ring bit length.

ε^*	$\log_2(n)$	ρ	$\log_2(\max_i\{\lvert\beta_i\rvert\})$	$\log_2(\max_i\{\lvert f_i^c\rvert\})$
10^{-5}	114	1.9×10^{11}	54	54
10^{-10}	147	1.9×10^{16}	71	71
10^{-20}	213	1.9×10^{26}	104	104
10^{-30}	280	1.9×10^{36}	137	137

Table 8.2: Numerical results

By considering the numerical results reported in Table 8.2 of [ASSK07] and the precision of a typical ECG database (e.g. the MIT-BIH database available on PhysioBank that has a precision of 10^{-4}), we choose a target final error of 10^{-5}, resulting in a representation of the feature vector $\mathbf{f^c}$ and the matrix **B** of 54 bits for each coefficient.

Speeding up the system

The analysis reported above is mainly based on worst case assumptions. In practice, we may expect that the number of bits necessary for a good classification accuracy is lower than 54. To investigate this aspect, we implemented a simulator to exactly understand which is the minimum number of bits that can be used. The results we obtained by running the simulator on the MIT Database of ECG signals are shown in Figure 8.4. This figure shows the accuracy of the system as a function of ℓ. As we can see $\ell = 44$ is sufficient to guarantee the same performance of a non-quantized implementation.

In order to further speed up the system, we tested a version of the ECG classifier with a reduced number of features. Specifically, we reduced $\mathbf{f}$ by eliminating even the feature n_1. In this way, we obtain a 15-coefficient $\mathbf{f^c}$. Obviously the reduction of the feature space results also in a reduction of the accuracy, but this reduction is quite negligible: our experiments, in fact, indicate that the accuracy decreases only from 88.33% to 86.30%. On the other hand, as it will be shown later, by removing one feature we gain a lot from a complexity point of view. Such a gain is already visible in Figure 8.4, where we can see that with the reduced set of features a value of ℓ as low as 24 is enough to obtain the same performance of a non-quantized version of the classifier.

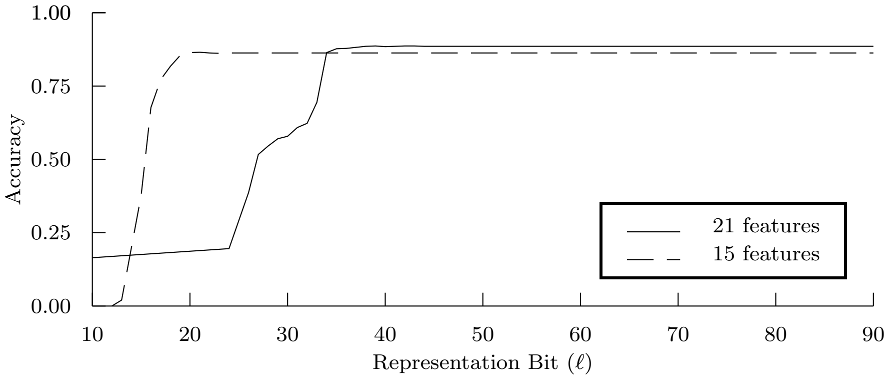


Figure 8.4: Classification accuracy of dataset using 21 and 15 features

8.2.2 Analysis

To evaluate the communication and computation complexity of the Hybrid and the GC protocols, we estimated the communication complexity according to previous analysis and implemented both protocols in C++. The security parameters in the protocols of [BFK+09a] are denoted by T for the bitlength of the RSA modulus for Paillier encryption [Pai99] in the Hybrid protocol, and t for the symmetric security parameter which determines the performance of the GC protocol using an elliptic-curve based OT protocol. For the tests, we chose the following security parameters according to common recommendations [GQ09]: $T = 1248,\ t = 80$ for short-term security (recommended use up to 2010[5]), $T = 2432,\ t = 112$ for medium-term security (up to 2030) and $T = 3248,\ t = 128$) for long-term security.

To analyze the protocols efficiency, we considered the parameter sizes proposed in Section 8.2.1:

- In scenario #1, we represent the features of $\mathbf{f}^{\mathbf{c},\ell}$ with $n = 21$ and $\ell = 56$ bits, as obtained from the theoretical estimations.
- In scenario #2, the features are represented with $n = 21$ and $\ell = 44$ bits, the lower value obtained from the practical tests.
- In scenario #3, we measure how the optimizations of Section 8.2.1 increase the efficiency of the protocols. Here $n = 15$ and $\ell = 24$ bits.

We recall that $d = 6$ and ℓ' is obtained according to (8.1). The estimated communication complexity of the protocol for secure classification of ECG data is given in Table 10.2. We consider the elliptic curve implementation of *OT* of Section 4.1 and we do not use offline precomputation of OT and transmission of the circuit. Note that since in the worst case ℓ'=92 and $d = 6$, only one pack is transmitted from $\mathcal{S}$ to $\mathcal{C}$.

To evaluate the real communication and computation complexity of the Hybrid and the GC instantiation of the protocol, we implemented both protocols in C++ using the Miracl library[6]. The measured communication and computation complexities for short-term security are shown in Table 8.4. The

---

[5] Tests were performed in 2009. Anyway these values are still widely used today.

[6] `http://www.shamus.ie`

Test #	Features	n	ℓ	Protocol Type	Short-term security	Medium-term security	Long-term security
1	5	21	54	GC	28.1 MByte	39.3 MByte	45.0 MByte
				Hybrid	105 kByte	151 kByte	175 kByte
2	5	21	44	GC	18.6 MByte	26 MByte	29.7 MByte
				Hybrid	88 kByte	126 kByte	147 kByte
3	4	15	24	GC	3.9 MByte	5.4 MByte	6.2 MByte
				Hybrid	50 kByte	72 kByte	84 kByte

Table 8.3: Estimated Communication Complexity

tests were performed on two PCs with 3 GHz Intel Core Duo processor and 4GB memory connected via Gigabit Ethernet. The third scenario has been tested even for medium-term security. We underline that the performance of the two software has not be stressed and not all the improvements described for GC and OT have been implemented.

Test #	Features	N	ℓ	Protocol Type	Computation			
					Client [s]		Server [s]	
					cpu	total	cpu	total
1	5	21	54	Hybrid	2.3	35.4	5.4	34.2
				GC	7.2	64.5	17.3	64.7
2	5	21	44	Hybrid	2.0	29.0	4.8	27.6
				GC	4.7	48.5	11.5	48.8
3	4	15	24	Hybrid	1.3	18.7	3.3	16.2
				GC	1.3	17.5	3.1	19.2
3*	4	15	24	Hybrid	6.5	40.5	16.3	30.9
				GC	3.0	20.4	4.6	20.8

* medium-term security

Table 8.4: Performance of protocols for secure ECG classification

Table 8.4 reports the computation complexity for the client and the server (separated into CPU time and total time which additionally includes data transfer and idle times). From these measurements we draw the following conclusions:

Parameter Sizes: The performance of both protocols in test #2 is slightly better than that of test #1 due to smaller size of ℓ. Reducing the number of features in test #3 results in substantially improved protocols while the classification accuracy is only slightly decreased as discussed in Section 8.2.1.

COMMUNICATION COMPLEXITY: The data transmitted in the GC protocol (MBytes) is an order of magnitude larger than in the Hybrid protocol (kBytes). However, we can easily observe that the biggest part of data is transmitted from $\mathcal{S}$ to $\mathcal{C}$. This asymmetric communication complexity of the GC protocol matches today's asymmetric network connections (e.g., ADSL or mobile networks), where the upstream is limited, while tens of MBytes can be downloaded easily. Future research should concentrate on further reducing the communication complexity of GC's.

COMPUTATION COMPLEXITY (SHORT-TERM SECURITY): For the test cases #1 and #2 the computation complexity of the Hybrid protocol is better by a factor of three in CPU time and factor two in total time, whereas for the optimized test case #3 both protocols have approximately the same computation complexity. Hence, for short-term security, the Hybrid protocol is better than the GC protocol with respect to computation and communication complexity (see also 'Communication Complexity' above).

COMPUTATION COMPLEXITY (MEDIUM-TERM SECURITY): Increasing the security parameters has a more dramatic effect on the computation complexity of the Hybrid protocol than on that of the GC protocol (see test #3 vs. #3*). This effect results from the asymmetric security parameter T being almost doubled, whereas the symmetric security parameter t is only slightly increased. We stress that this loss in performance of additively homomorphic encryption for realistic security parameter sizes is often neglected in literature or hidden by choosing relatively small moduli sizes of $T = 1024$ bit. For medium-term security, the GC protocol is substantially better than the Hybrid protocol.

By representing the quantized features with a lower number of bits (the minimum that permits a correct evaluation) and omitting the feature related to the AR model error, the communication and computation is decreased significantly. Working with 4 features and quantizing the features with 24 bits, we have a slightly loss of accuracy of the protocol, opposed to an important complexity decrease. This is acceptable because an ECG may be processed in a very short time, opening the way to real time processing of ECG signals in the encrypted domain. As expected, we can see that the Hybrid version of the protocol has a much lower communication complexity, while the computation complexity of the two protocols is nearly the same (still with some advantage for the hybrid protocol).

Toward a more efficient LBP

We conclude the chapter showing how a more efficient LBP implementation can be obtained by introducing precomputation on the Hybrid-LBP. For the comparison we consider the third scenario with short-term security.

OFFLINE PHASE In the offline phase, the garbled circuits are generated by $\mathcal{S}$ $(2\cdot d\ell'+1 = 2\cdot 306+1$ garbled tables, where $\ell' = 2*24+\lceil\log_2 15-1\rceil = 51))$ and transferred to $\mathcal{C}$ $[(2\cdot 306\cdot 4+(2^d))\cdot t \approx 2512t$ bits$]$. Additionally, 306 parallel OTs need to be pre-computed. Being $306 > 3t$ the extended OT protocol described in Section 4.4 is used, resulting in the evaluation of an OT_t^t having complexity $\approx 6t^2$ bits and an extension to $OT_t^{d\ell'}$ that requires the transmission of other $\approx 4d\ell' t = 1224t$ bits. The offline phase requires 4 rounds.

ONLINE PHASE The communication complexity in the online phase is approximately 15 cyphertexts from $\mathcal{C}$ to $\mathcal{S}$ and one from $\mathcal{S}$ to $\mathcal{C}$. Moreover an online OT of $d\ell' = 306$ values is performed, while $\mathcal{S}$ inputs $d\ell'$ secrets for the least significant bits of r and $d\ell'$ secrets for the thresholds. The bandwidth necessary for the online phase is $16 * 2T + 306 * 2t + 306 * t + 306 * t = 32T + 1224t$ bits. The online phase requires 4 rounds.

The resulting complexity of the LBP-based ECG classification protocol is summarized in Table 8.5.

Optimization If $\mathcal{S}$ does not want to hide to $\mathcal{C}$ that indeed all thresholds $t_i^{\ell'}$ are zero, we can omit the comparison and directly use the most significant bit instead. By doing this, we no longer need the comparison circuits and save

Phase	Rounds	Data [bits]
Setup	4	$3736t + 6t^2$
Online	4	$32T + 1224t$

Table 8.5: Communication complexity of LBP-based ECG classification protocol.

306 non-XOR gates. This yields a more efficient protocol where in the setup phase $4 \cdot 306 = 1,224$ less invocations of H by $\mathcal{S}$ and $4 \cdot 306t = 1224t$ less communication are needed and in the online phase $\mathcal{C}$ needs 306 less invocations of H.

Chapter 9

Privacy preserving Classification by using Neural Network

In this chapter we address the classification of ECG signals through artificial Neural Networks (NN). In fact, neural networks are well-know machine learning structures used in many different fields ranging from approximation to classification. NNs are widely used as classifiers and, in general, they give good results if the set used to train the network is representative of all the considered classes and the generalization grade is good enough (see [KB95] or [AB01]).

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

In Section 9.1 the development of a Neural Network implementing the ECG classification in the plain domain is described. Then we show how the NN can be adapted to work with integer values (Section 9.2), as necessary for a s.p.e.d. implementation and finally we describe and analyze its implementation in the encrypted domain (Section 9.3), also providing a comparison with the LBP-based solution of Chapter 8.

9.1 Neural Network design

Before going on, in order to ease the description of the s.p.e.d. protocol based on a NN classifier, we review the details of the operations carried out by a NN. Generally speaking each neuron in a NN performs only two simple operations: a scalar product and a function evaluation. As shown in Figure 9.1, a single neuron, or *perceptron*, consists of a number of weighted connections $\mathbf{w} = (w_1, w_2, ..., w_n)^T$, a bias b and an activation function $f(\cdot)$. When a new input vector $\mathbf{x} = (x_1, x_2, ..., x_n)^T$ is provided, the perceptron performs a scalar product among the weights and the input vector and adds up the bias:

$$b + \sum_{i=1}^{n} w_i \; x_i = b + \mathbf{w}^T \; \mathbf{x} \tag{9.1}$$

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

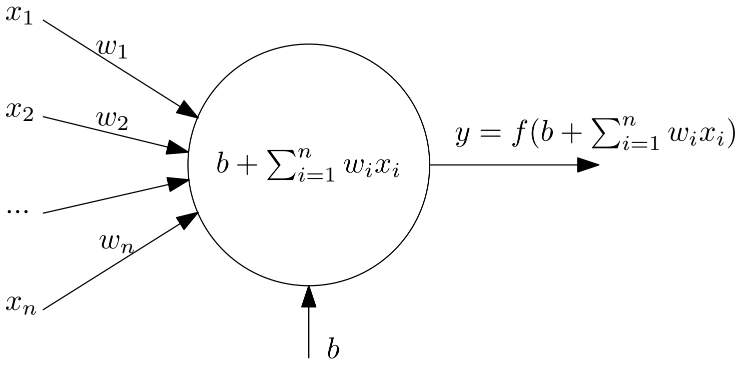

x_1
w_1
x_2
w_2
...
w_n
x_n
$b + \sum_{i=1}^{n} w_i x_i$
$y = f(b + \sum_{i=1}^{n} w_i x_i)$
b

Figure 9.1: A perceptron.

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

- n is the number of inputs of the NN, and $\mathbf{x} = (x_1, x_2, ..., x_n)^T$ is the input vector,
- n_h is the number of neurons in the hidden layer (the first NN layer),
- n_o is the number of neurons in the output layer,
- $\mathbf{W}_h$ is a matrix of size $n_h \times n$ whose elements are the weights of the connections between the inputs and the hidden layer, that is: $\mathbf{W}_h(i, j) = w_{h;i,j}$ is the coefficient used to weight the connection between the j-th input and the i-th node of the hidden layer,

- $\mathbf{W}_o$ is a matrix of size $n_o \times n_h$ whose elements are the weights of the connections between the hidden and the output layers, that is: $\mathbf{W}_o(i,j) = w_{o;i,j}$ is the coefficient used to weight the connection between the output of the j-th node of the hidden layer and the i-th node of the output layer,
- $\mathbf{y}_h$ is a vector of length n_h with the outputs of the neurons of the hidden layer,
- $\mathbf{y}_o$ is a vector of length n_o with the outputs of the neurons of the output layer,
- $\mathbf{b}_h$ is a vector of length n_h that contains the biases of the neurons of the hidden layer,
- $\mathbf{b}_o$ is a vector of length n_o that contains the biases of the neurons of the output layer.

With the above notation, the output of the i-th neuron in the hidden layer is:

$$y_{h;i} = f(b_{h;i} + \sum_{j=1}^{n} w_{h;i,j}\ x_j) \tag{9.2}$$

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

$$\mathbf{y}_h = f(\mathbf{b}_h + \mathbf{W}_h\ \mathbf{x}), \tag{9.3}$$

where the activation function $f(\cdot)$ is applied component-wise to all the values of the input vector. Similarly the output of the k-th neuron of the output layer is

$$y_{o;k} = b_{o;k} + \sum_{i=1}^{n_h} w_{o;k,i} y_{h;i} \tag{9.4}$$

that in matrix form can be written as:

$$\mathbf{y}_o = \mathbf{b}_o + \mathbf{W}_o\ \mathbf{y}_h, \tag{9.5}$$

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

---

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

an output neuron returns a positive value. If more neurons have a positive output, the one returning the highest value is chosen for classification.

In our scenario, the number of inputs is dictated by the number of features the classifier relies on, while the number of output layers corresponds to the number of diseases the NN should distinguish, so we have $n = 4$ and $n_o = 6$. The degrees of freedom we have, then, are n_h, the number of neurons in the hidden layer, and the activation function.

To choose n_h, we carried out some tests trying to reach the same accuracy provided by the LBP-based classifier. There is not a rule to choose the correct number of neurons for the hidden layer. Considering that the network has 4 inputs and 6 outputs, we can suppose that the hidden layer may have from 4 to 24 neurons. Hardly a higher number of neurons is necessary. Being the complexity of the network related to the number of neurons, the research has been restricted to a maximum of 15 neurons. A training set was built by using as samples the pair: $(\mathbf{f}, \mathbf{o})$ where, as said before, $\mathbf{f}$ is the feature vector $(f_1, f_2, f_3, f_4)^\top = (a_1, a_2, a_3, a_4)^\top$ and $\mathbf{o}$ is a six-component vector, having value equal to 1 for the index of the class the ECG signal belongs to and -1 elsewhere[2]. In our experiments we used a dataset of 200 ECG signals (and the corresponding 200 feature vectors) taken from PhysioBank [GAG+00]. Specifically, we split the dataset into a training set (containing 140 ECG sequences) and a test set (with the remaining 60 signals)[3]. While the size of the dataset may not be realistic for real life applications, we decided to use it since this dataset is often used in relevant literature on ECG classification; for instance it is the same used in [ASSK07]. As it will be clear from the following discussion, the size of the dataset does not have a direct impact on the structure of the proposed protocols (while it surely has a great impact on the training phase), however it is possible that for larger datasets a larger number of features is needed thus impacting on the complexity of the overall protocols.

Together with the number of nodes in the hidden layer, the activation function has to be chosen. The natural choice is to use the Hyperbolic Tangent

---

[2] This kind of Neural Network is often called NN with *fired* output.

[3] NN network training require a large dataset, hence we had the necessity to choose a bigger training set than the one used for the LBP implementation, where we chose the training and test dataset size according to [GSK02].

Sigmoid transfer function ($tansig(\cdot)$) widely used in NNs (Figure 9.2(a)):

$$tansig(x) = \frac{2}{1 + e^{-2x}} - 1 \tag{9.6}$$

As opposed to the test phase, during training we have the necessity to use an activation function that returns values in the range $[-1, 1]$, the values used for the vector $\mathbf{o}$, hence during training the $tansig(\cdot)$ function is used even in the output layer. Unluckily, the $tansig(\cdot)$ function is difficult to implement in a s.p.e.d. protocol, hence a simpler solution is necessary. Being simple linear approximations of the $tansig(\cdot)$ function and considering their easy implementation in the encrypted domain, the preferable transfer functions are the Hard-Limit ($hardlim(\cdot)$) transfer function (Figure 9.2(b)) or the Symmetric Hard-Limit ($hardlims(\cdot)$) transfer function(Figure 9.2(c)), whose implementation relies on a `CMP` circuit, but unfortunately training a Neural network with a continuous and non-derivable transfer function is difficult and training with a non-continuous function is quite impossible.

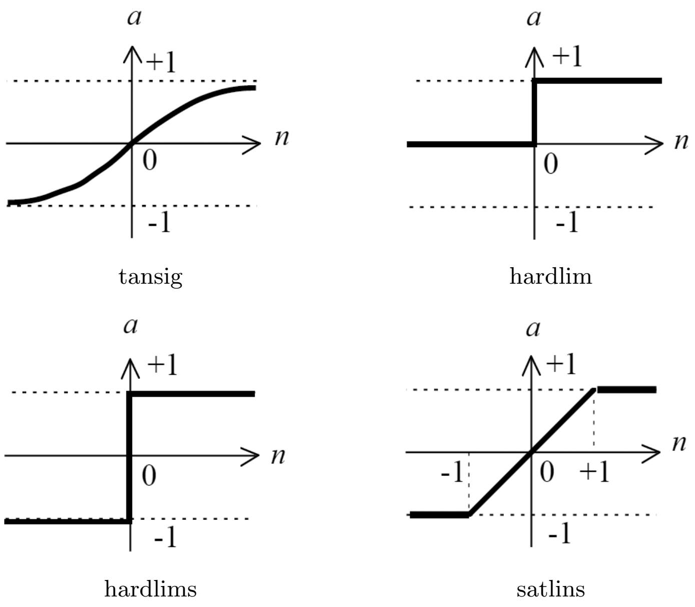


Figure 9.2: Transfer functions

We exploited two approaches to train the NN. First of all the neural network was trained by using the *tansig*$(\cdot)$ function and then other functions, easy to implement in the encrypted domain, were usedduring the tests, checking how accuracy changes with the number of neurons in the hidden layer. Figure 9.3 shows the classification accuracy as a function of the number of neurons of the hidden layer obtained by using the *tansig* function during tests and the accuracy obtained substituting it with the *hardlims*$(\cdot)$ transfer function.

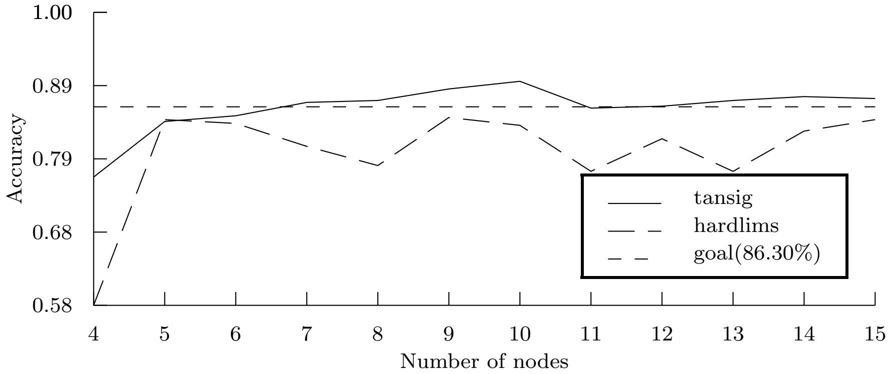


Figure 9.3: Classification accuracy as a function of the node number with the tansig or hardlims transfer functions in the same Neural networks.

Being the classification accuracy not sufficient, we tried to substitute the training function with the Symmetric Saturating Linear (*satlins*$(\cdot)$) transfer function (Figure 9.2(d)), defined by:

$$f(x) = \mathsf{SATLIN}(x) = \begin{cases} 1 & \text{if } x > 1 \\ x & \text{if } -1 < x < 1 \\ -1 & \text{if } x < -1, \end{cases} \tag{9.7}$$

obtaining the results shown in Figure 9.4.

As second solution, we tried to directly train the NN with the *satlins* function. Since *satlins*$(\cdot)$ does not have a continuous derivative, training a NN with it is quite difficult, hence we used different training functions. We obtained the best results with the Levenberg-Marquardt regularized learning algorithm [HM94]. Using this setup and 100 epochs for training, we obtained the classification accuracy shown in Figure 9.5. Considering the results of

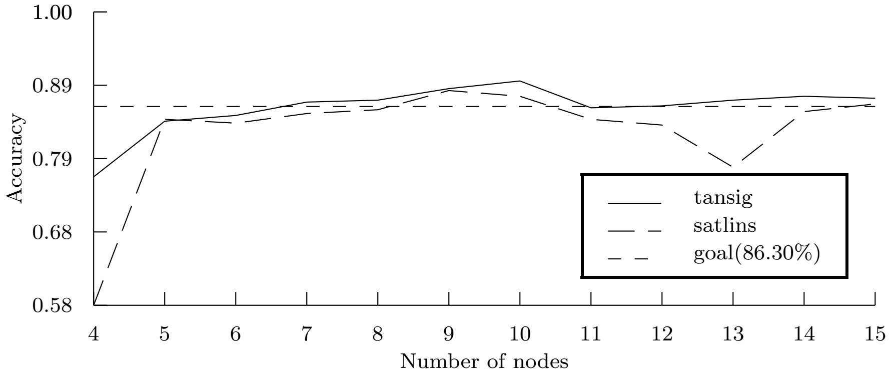


Figure 9.4: Classification accuracy as a function of the node number with the tansig or satlins transfer functions in the same Neural networks.

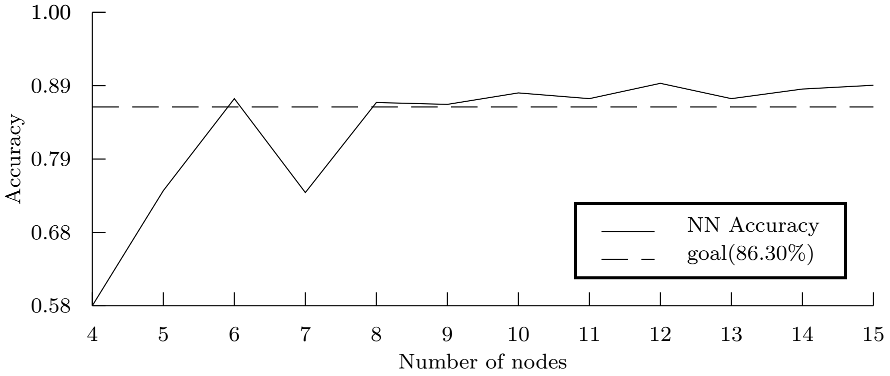


Figure 9.5: Classification accuracy as a function of the number of nodes in the hidden layer and SATLIN as activation function.

Figure 9.5, we developed the NN described in Figure 9.6 (for the sake of simplicity the figure shows only a part of the involved variables).

The NN is the smallest giving a classification accuracy larger than 86.30%. It has $n_h = 6$ neurons in the hidden layer, trained and evaluated with the *satlins*$(\cdot)$ function. Note that in the final classification no activation function is used in the output layer (that is equivalent to use $f(x) = x$ as activation function) and the final classification is obtained according to the position of the maximum among the outputs of the NN.

A compact representation of the NN is given in Figure 9.7 where the

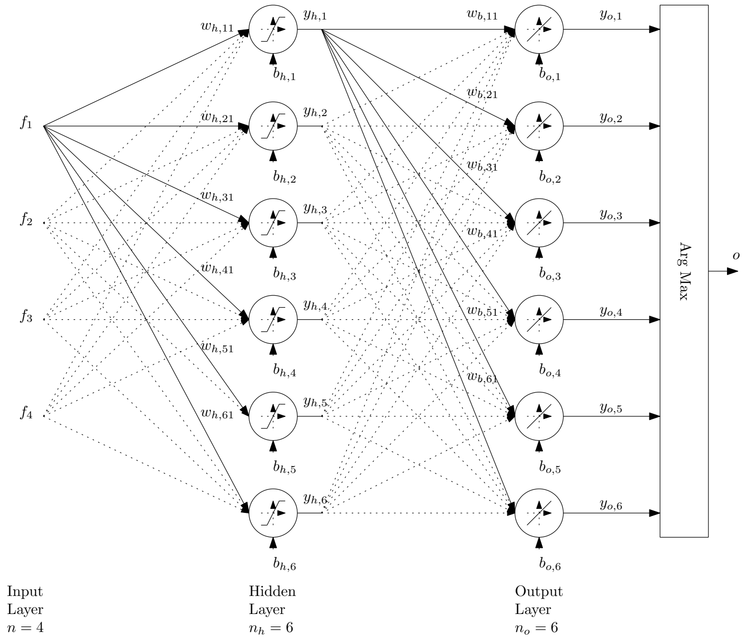


Figure 9.6: NN topology.

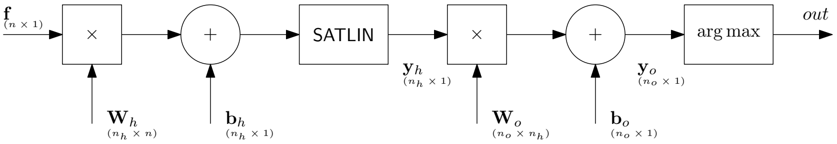

$\mathbf{f}$ $(n \times 1)$
×
+
SATLIN
×
+
arg max
out
$\mathbf{W}_h$ $(n_h \times n)$
$\mathbf{b}_h$ $(n_h \times 1)$
$\mathbf{y}_h$ $(n_h \times 1)$
$\mathbf{W}_o$ $(n_o \times n_h)$
$\mathbf{b}_o$ $(n_o \times 1)$
$\mathbf{y}_o$ $(n_o \times 1)$

Figure 9.7: Chain blocks to compute the output of a NN.

entire flow needed to evaluate the NN is shown. In particular we recall that in formulas we have:

$$\mathbf{y}_o = \mathbf{b}_o + \mathbf{W}_o \ \mathsf{SATLIN}\left(\mathbf{b}_h + \mathbf{W}_h \ \mathbf{f}\right), \tag{9.8}$$

with the finally classification result computed as:

$$o = \arg\max \mathbf{y}_o. \tag{9.9}$$

9.2 Quantized Neural Network classifier

In this section we introduce the quantized version of the NN classifier described so far, i.e. a version of the classifier that works only with integer numbers.

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

Working with quantized values, the quantized output vector $\mathbf{y}_{h_q}$ of the hidden layer is:

$$\mathbf{y}_{h_q} = \mathsf{QSATLIN}\left(\lfloor q_i q_h \mathbf{b}_h \rceil + \lfloor q_i q_h \mathbf{W}_h\ \mathbf{f} \rceil\right), \tag{9.10}$$

where the biases $\mathbf{b}_h$ have been multiplied by both q_i and q_h to make the bias homogeneous with the term $q_i q_h \mathbf{W}_h\ \mathbf{f}$. As to the SATLIN function of Eq. 9.7, we have replaced it with its quantized version defined as follows:

$$\mathsf{QSATLIN}(x) = \begin{cases} q_i q_h & \text{if } x \geq q_i q_h \\ x & \text{if } -q_i q_h < x < q_i q_h \\ -q_i q_h & \text{if } x \leq -q_i q_h \end{cases} \tag{9.11}$$

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

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

$$\mathbf{y}_{o_q} = \lfloor q_o q_h q_i \mathbf{b}_o \rceil + \lfloor q_o \mathbf{W}_o \ \mathbf{y}_{h_q} \rceil \tag{9.12}$$

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

$$o = \arg\max\{\mathbf{y}_{o_q}\} \tag{9.13}$$

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

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

We can summarize the operation of a quantized NN with the required

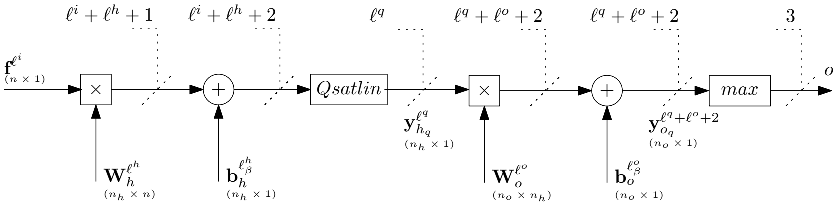

$\ell^i + \ell^h + 1$
$\ell^i + \ell^h + 2$
ℓ^q
$\ell^q + \ell^o + 2$
$\ell^q + \ell^o + 2$
3
$\mathbf{f}^{\ell^i}_{(n \times 1)}$
×
+
Qsatlin
×
+
max
o
$\mathbf{W}_h^{\ell^h}$ $(n_h \times n)$
$\mathbf{b}_h^{\ell^h_\beta}$ $(n_h \times 1)$
$\mathbf{y}_{h_q}^{\ell^q}$ $(n_h \times 1)$
$\mathbf{W}_o^{\ell^o}$ $(n_o \times n_h)$
$\mathbf{b}_o^{\ell^o_\beta}$ $(n_o \times 1)$
$\mathbf{y}_{o_q}^{\ell^q+\ell^o+2}$ $(n_o \times 1)$

Figure 9.8: Quantized NN. In our case we have $n = 4, n_h = 6$ and $n_o = 6$.

bit-lengths at each step, with the following formula (see also Figure 9.8):

$$\mathbf{y}_{o_q} = \underbrace{\mathbf{b}_o^{(\ell^o_\beta)} + q_o \mathbf{W}_o^{(\ell^o)} \underbrace{\mathsf{QSATLIN}\left(\underbrace{\mathbf{b}_h^{(\ell^h_\beta)} + \underbrace{\mathbf{W}_h^{(\ell^h)}\ \mathbf{f}^{(\ell^i)}}_{\ell^i+\ell^h+1}}_{\ell^i+\ell^h+2} \right)}_{\ell^q}}_{\ell^q+\ell^o+2} \tag{9.14}$$

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

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

9.2.1 Representation vs. Classification Accuracy

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

In order to determine the minimum number of bits necessary to reach the same classification accuracy of the LBP classifier, we run a simulator that evaluates the classification accuracy of the NN in the case $\ell^i = \ell^h = \ell^o$, obtaining the results shown in Figure 9.9. To guarantee the same classification obtained by the LBP classifier, we must let $\ell^i = \ell^h = \ell^o = 13$ bits. These numbers of bits (namely ℓ^i and ℓ^h) are those necessary to correctly represent the integer part of the input features and NN parameters multiplied by the quantization factors q_i and q_h. Specifically, according to the experiments we carried out with the PhysioBank database, they correspond to $q_i = 511.87$ and $q_h = 255.94$.

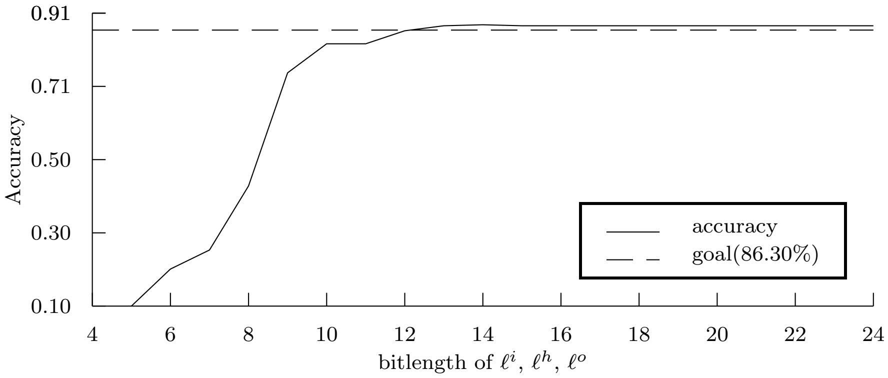


Figure 9.9: Classification accuracy as a function of ℓ^i, ℓ^h, ℓ^o.

The results of the scalar products in the hidden layer are used as input in the $\mathsf{QSATLIN}(\cdot)$ function that gives an output with magnitude bounded by $q_i q_h$, hence it can be represented with $\ell^q = \lceil \log_2 q_i q_h \rceil + 1$ bits. From the values given above we obtain $q_i q_h = 131008$ yielding $\ell^q = 18$ bits, including the sign bit. As a further test we tried to reduce the number of bits used for the output layer once the bitlength of the parameters of the hidden layer has been fixed. As shown in Figure 9.10, letting $\ell^o = 8$ is sufficient to guarantee the same accuracy. A summary of the parameters we used for the NN classifier are given in Table 9.1.

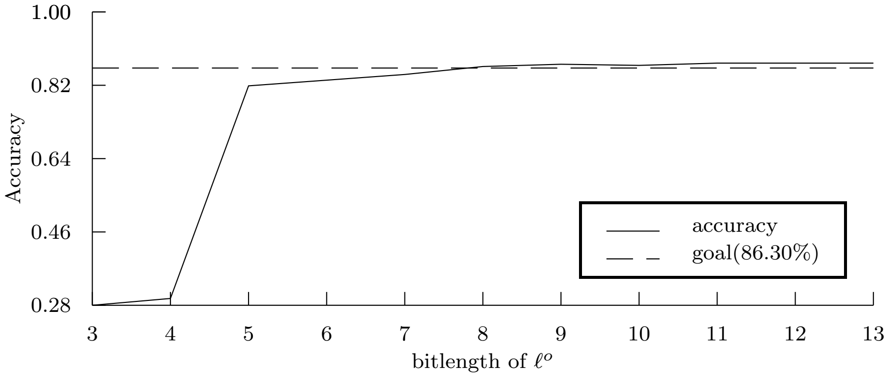


Figure 9.10: Classification accuracy in function of ℓ^o, with $\ell^i = \ell^h = 13$.

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

Table 9.1: NN Parameters.

9.3 Privacy-Preserving NN classifier

With respect to previous works in NN computation in a SFE framework (e.g. [OPB07] where the activation functions are implemented by introducing a rather security-critical multiplicative blinding step), our solution is provably secure and computationally efficient. Specifically, the general structure of our protocol follows the approach of [SS08] where the quantized NN for classification (in our case the NN shown in Figure 9.8) is represented as a boolean circuit which is evaluated securely with a garbled circuit protocol. While [SS08] gives only concrete circuit instantiations for threshold functions, our NN uses the QSATLIN function for which we give an efficient circuit instantiation.

We now describe the garbled circuit implementing the NN according to the parameters determined in the previous section:

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

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

In total $\mathcal{C}$ has 52 input wires and $\mathcal{S}$ has $312 + 150 + 288 + 150 = 900$ input wires.

Circuit: The circuit is constructed by instantiating the blocks of Figure 9.8 with circuit building blocks as follows.

The inputs $\mathbf{f}^{(\ell^i)}$ and $\mathbf{W}_h^{(\ell^h)}$ are given in sign-magnitude representation s.t. we can easily compute their component-wise product: the magnitude is the product of the input magnitudes using $n \cdot n_h$ MUL blocks for 12-bit unsigned integers $(4 \cdot 6 \cdot (2 \cdot 12^2 - 12) = 6{,}624$ non-XOR gates); the sign is computed "for free" by XORing the input signs. Now, depending on the sign, the magnitudes of the products are added to or subtracted from $\mathbf{b}_h^{(\ell^h_\beta)}$ by using ADDSUB blocks (at most $n_h \cdot (n \cdot (\ell^i + \ell^h + 1)) = 6 \cdot 4 \cdot 27 = 648$ non-XOR gates). The output $\mathbf{b}_h^{(\ell^h_\beta)} + \mathbf{W}_h^{(\ell^h)} \mathbf{f}^{(\ell^i)}$ is a vector with $n_h = 6$ components of $\ell^i + \ell^h + 2 = 28$-bit signed values in 2's complement representation.

Afterwards, for each of the $n_h = 6$ neurons the QSATLIN activation function is evaluated: first, the 28-bit input (let us call it x) is converted from 2's complement into sign/magnitude representation with an ADDSUB block (27 non-XOR gates). Afterwards, the QSATLIN function is computed according to Eq. (9.11) as $\mathsf{QSATLIN}(x) = \mathsf{sign}(x) \cdot \mathsf{min}(\mathsf{abs}(x), q_i q_h)$. The minimum is computed by comparing the magnitude of x with 1 using a CMP block (27 non-XOR gates). Depending on the outcome of this comparison, the 17-bit magnitude of the outcome is either the magnitude of x or $q_i q_h$ selected with a MUX block [17 non-XOR gates]. Overall, the conversion and computation of the $n_h = 6$ $\mathsf{QSATLIN}$ functions requires $6 \cdot (2 \cdot 27 + 17) = 426$ non-XOR gates. The output $\mathbf{y}_{h_q}$ is a vector of $n_h = 6$ components of $\ell^q = 18$-bit signed values in sign/magnitude representation.

The value $\mathbf{y}_{o_q} = \mathbf{b}_o^{(\ell^o_\beta)} + \mathbf{W}_o^{(\ell^o)} \mathbf{y}_{h_q}$ is computed similarly to the computation of $\mathbf{b}_h^{(\ell^h_\beta)} + \mathbf{W}_h^{(\ell^h)} \mathbf{f}^{(\ell^i)}$ described before and requires a circuit of at most $n_o n_h (2(\ell^q - 1)(\ell^o - 1) - (\ell^q - 1)) + n_o n_h (\ell^q + \ell^o + 2) = 6 \cdot 6 \cdot (2 \cdot 17 \cdot 7 - 17) + 6 \cdot 6 \cdot 28 = 7{,}956 + 1{,}008$ non-XOR gates. The output $\mathbf{y}_{o_q}$ is a vector with $n_o = 6$ components of $\ell^q + \ell^o + 2 = 28$-bit signed values in 2's complement representation.

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

OUTPUTS: The output of the circuit for $\mathcal{C}$ is $\mathbf{o}$ (3 wires).

In total, the circuit has 52 input wires of $\mathcal{C}$, $312 + 150 + 288 + 150 = 900$ input wires of $\mathcal{S}$, at most $6{,}624 + 648 + 426 + 7{,}956 + 1{,}008 + 259 \lessapprox 17{,}000$ non-XOR 2-input gates and 3 output wires for $\mathcal{C}$.

As the NN classifier can be represented as a reasonably small boolean circuit C, it can be evaluated securely with Yao's garbled circuit (GC) protocol as described in Chapter 5. The inputs to the protocol are the quantized inputs of $\mathcal{C}$: $\mathrm{in}_{\mathcal{C}} = (\mathbf{f}^{(\ell^i)})$ and $\mathcal{S}$: $\mathrm{in}_{\mathcal{S}} = (\mathbf{W}_h^{(\ell^h)}, \mathbf{b}_h^{(\ell^h_\beta)}, \mathbf{W}_o^{(\ell^o)}, \mathbf{b}_o^{(\ell^o_\beta)})$. These plain inputs are converted into their corresponding garbled inputs $\tilde{\mathrm{in}}_{\mathcal{C}}, \tilde{\mathrm{in}}_{\mathcal{S}}$ provided to $\mathcal{C}$ who uses them to evaluate a garbled circuit $\tilde{C}$ created by $\mathcal{S}$ to obtain the garbled output $\tilde{z} = \tilde{C}(\tilde{\mathrm{in}}_{\mathcal{C}}, \tilde{\mathrm{in}}_{\mathcal{S}})$. Finally, the garbled output is converted into the plain value $\mathbf{o} = z$ output to C.

The costs for this NN-based ECG classification protocol with symmetric parameter t are summarized in Table 9.2:

SETUP PHASE the garbled circuit $\tilde{C}$ is generated by $\mathcal{S}$ and transferred to $\mathcal{C}$ $(17{,}000 \cdot 4(t + 1) \approx 68{,}000t$ bits). Additionally, $|\mathrm{in}_{\mathcal{C}}| = 52$ parallel OTs need to be pre-computed to convert $\mathcal{C}$'s input $\mathrm{in}_{\mathcal{C}}$ into its garbled version $\tilde{\mathrm{in}}_{\mathcal{C}}$. Being the number of OT lower than t, the extended OT protocol is not used. The setup phase requires 3 moves.

Online Phase: In the online phase, $\mathcal{C}$ obtains the garbled inputs $\tilde{\mathrm{in}}_{\mathcal{S}}$ corresponding to $\mathcal{S}$'s input $\mathrm{in}_{\mathcal{S}}$ ($900 \cdot (t+1) \approx 900t$ bits) and executes the online phase of 52 parallel OTs ($\approx 104t$ bits) which requires 2 moves.

Phase	Moves	Data [bits]
Setup	3	$68,312t$
Online	2	$1,004t$

Table 9.2: Complexity of NN-based ECG classification protocol.

9.3.1 Comparison with the LBP solution

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

For the length of the security parameters, we consider short-term security ($t = 80$ and $T = 1248$ bits) and long-term security ($t = 128$ and $T = 3248$ bits). In both cases, the NN protocol requires approximately 16 times more bandwidth than the LBP solution in the setup phase (e.g., 667 kByte compared to 41 kByte for short-term security). For the short-term security, the NN requires 33% less bandwidth than LBP (14.9 kByte for LBP and 9.8 kByte for NN), and for long-term security the difference between the complexity of the protocols grows to 45% less bandwidth for the NN solution (15.7 kByte compared to 28.7 kByte).

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

In summary we can state that the LBP classifier is preferable from a communication complexity point of view when considering the total amount of data sent in setup plus online phase. The reason is that, as shown in Chapter 6, the 12 bit values to be multiplied are so small that the HE and GC implementations are roughly equivalent. An hybrid implementation of the

NN would need the use of homomorphic protocols for the scalar products in both the layer, the first one followed by a circuit implementing the QSATLIN function in the hidden layer and the second one followed by the ARGMIN circuit. The small benefit eventually introduced by the use of HE is widely canceled out by the 3 interfaces needed.

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

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

We conclude our discussion by observing that the complexity of both protocols depend on the number of features used to classify the ECG signals. In the NN case the dependence of the size of the classifier on the number of features is not easy to determine. On one side it results in an increase of the size of the input layer of the NN, with a linear impact on the complexity of the part of the protocol corresponding to the computation of the input of the hidden layer. On the other side it is likely that the number of neurons in the hidden layer will have to increase as well thus resulting in a superlinear dependence of the complexity on the number of features. The overall complexity increases, however, is likely to be less than quadratic, given that the size of the output layer will remain constant. In the LBP case the dependence is at least quadratic due to the inclusion within the composite feature vector of quadratic terms[4]. For this reason, we expect that the NN structure is going

---

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

to become even more advantageous if the number of features considered by the classifier increases.

Part III

Quality Evaluation

In a remote healthcare scenario, a patient applying electrodes for the ECG recording is not an expert and can connect the electrodes wrongly, while the hospital or the company involved in the 2PC protocol is not able to see the signal and evaluate its quality, with the risk to perform a wrong analysis. The application of a technique for signal quality evaluation becomes then necessary.

We describe a methodology which allows the service provider to obtain a quality measure so that the quality of the input to later computations is guaranteed and the corresponding output (the actual analysis performed on the data) complies with certain confidence criteria. In particular, we propose a protocol that allows evaluating the quality of ECG signals and communicates to the user whether the input signal is good enough to continue with the analysis or not [LGB12]. This protocol is based on the evaluation of the Signal-to-Noise-Ratio (SNR) between the originally measured signal and a filtered version of it.

The quality evaluation can be efficiently performed in a s.p.e.d. scenario by using an hybrid protocol that needs the transmission of less than 4MBytes for 30 seconds analysis and guarantees classification results close to 85%.

"If the ECG isn't broken then we have problem"

©*Goldsmith*

Chapter 10

SNR computation

In the previous part of the thesis we addressed the problem of biomedical signal processing in a s.p.e.d. scenario, demonstrating that it is possible to protect the privacy of the patients in a remote healthcare scenario. Considering that the patient has to apply electrodes for the ECG recording by himself or with the help of another person that is not necessary an expert, some electrodes could be connected wrongly. On the other side the hospital or the company involved in the 2PC protocol is not able to see the signal and evaluate its quality. A signal with poor quality can produce a wrong result, so that a healthy signal can be classified as sick or, worst, some disease symptom are not identified. Hence a technique for the signal quality evaluation becomes necessary. Even if quality evaluation can be embedded in the recording device available to the user, a privacy preserving quality evaluation protocol may be useful whenever the service provider is interested to change the protocol parameters without upgrading the software or hardware of the devices[1]. Otherwise if the service provider and the company producing the device are two distinct elements and the former is interested to protect his property also by the company. Moreover we can imagine quality evaluation as part of an analysis chain, and not necessarily the beginning of the chain, where the input signal is available only in its encrypted form. Also device calibration can be considered a major issue, but if performed in presence of trusted stuff, as we here suppose, we can relax the privacy and security constrains related to.

Measuring signal quality is a difficult problem that has captured the attention of researchers in many fields, as diverse as audio, image, video and medical signal analysis. Generally speaking, quality evaluation techniques can be split into two main categories: full reference or no-reference techniques. The former class refers to a situation in which the quality of a signal has to

---

[1] Software upgrade can cause security loss, while hardware upgrade is expensive.

be judged by referring to an *ideal* signal. This is the case in lossy compression applications wherein the quality of the compressed signal has to be judged by considering its perceptual *distance* from the original signal. In contrast, no-reference techniques have to measure the quality of a signal without making any reference to an ideal signal that is supposed to represent the maximum possible quality.

The medical scenario addressed in this thesis requires that no-reference quality measures are adopted, because only the noisy version of the ECG signal is available. This is a difficult problem for which only a few (plain domain) solutions are available (see [LMC08, ADAI$^+$09] for example). In order to be as general as possible, we focus on a very simple scheme in which the quality of a signal is given by the Signal-to-Noise-Ratio (SNR) between the originally measured signal $\mathbf{y}$ and a filtered version of the signal, denoted by $\mathbf{x}$, as shown in [BGL10]. The rationale behind this choice is that the filtered (denoised) signal represents the ideal signal corresponding to $\mathbf{x}$. Furthermore, the removed noise, i.e. the distance of $\mathbf{y}$ from $\mathbf{x}$, represents a measure of the quality of the measurement $\mathbf{y}$. A natural choice is to represent the SNR, defined as the ratio between the energy of $\mathbf{x}$ and the energy of the noise $\mathbf{y}-\mathbf{x}$, in dB, i.e. by using a logarithmic scale. The reason for such a choice is twofold: first of all this is a consolidated practice in signal processing, moreover we will show that by replacing the computation of a ratio with the computation of the difference of two logarithms, we can considerably reduce the complexity of a secure protocol for the evaluation of the SNR.

We are, thus, interested in developing a s.p.e.d. protocol that allows a client $\mathcal{C}$ and a server $\mathcal{S}$ to compute the signal quality of $\mathbf{y}$ as shown in Figure 10.1, accessing only the encrypted version of $\mathbf{y}$ and allowing interaction. By using a s.p.e.d. protocol, the client $\mathcal{C}$ protects the privacy of his signal while he is not allowed to know the intermediate results of the computation and the parameters of the algorithms used by $\mathcal{S}$, which may be interested to protect the details of the quality evaluation algorithm (specifically the coefficients of the denoising filter) since this may be a proprietary algorithm. Moreover the computation of the quality of the signal may be only an intermediate step of a longer processing chain whose details need to be kept secret, and whose security could be compromised if some intermediate results were disclosed.

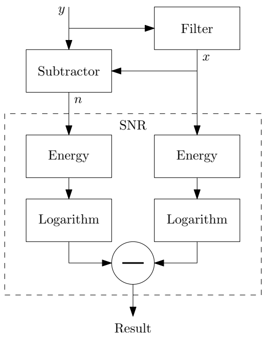


Figure 10.1: Scheme to compute the SNR.

For the privacy preserving implementation we propose an hybrid protocol and, in order to base the analysis on real data and evaluate the actual efficiency of the proposed solution, we apply our protocol to the evaluation of the quality of ECG signals, by referring to the MIT-BIH Arrhythmia Database[2] available in PhysioBank archives [GAG+00], containing recordings of 30-minute long two lead ECGs. Despite our choice to apply the signal quality evaluation to ECG signals, it is important to underline that our approach is rather general and can also be applied to different signals.

The remainder of this chapter is organized as follows. Noise affecting ECG signals are presented in Section 10.1. In Section 10.2, we introduce the protocols to compute signal quality based on SNR in the encrypted domain. We analyze the size of the operands, the complexity of the protocol, and sketch its security in Section 10.3. The analysis of the operands is general but figures for the ECG case are also provided. We also report on the error incurred in the logarithm computation in Section 10.4 demonstrating that we can ignore it in practice.

10.1 Noise affecting ECG recording

Before describing the protocol for the evaluation of the quality of an ECG signal, it is necessary to understand which are the main noise sources. An ECG can be contaminated by different kinds of noise, depending on the elec-

---

[2] http://www.physionet.org/physiobank/database/mitdb/

trical components of the system, patient movement (intentional or not), bad electrodes connection, etc. The noise usually contaminating ECG signals are[3]:

Power Line Interference: this noise consists of 60/50 Hz pickup and harmonics that can be modeled as sinusoids and combination of sinusoids. According to Friesen et al [FJJ+90], the frequency content of this kind of noise is 60/50 Hz with harmonics and the amplitude is 50% of peak-to-peak ECG amplitude.

Electrode Contact: this noise is a transient interference caused by loss of contact between the electrode and the skin, which can be permanent or intermittent. The switching action can result in large artifacts since the ECG signal is usually capacitively coupled to the system. This type of noise can be modeled as a randomly occurring rapid baseline transition that decays exponentially to the base line and has a superimposed 60 Hz component. According to [FJJ+90], the duration of the noise signal is 1 sec and the amplitude is the maximum-recorded output with the frequency of 60 Hz.

Motion Artifacts: this noise is due to transient base line changes in the electrode skin impedance with electrode motion. The shape of the base line disturbance caused by the motion artifacts can be assumed to be a biphasic signal resembling one cycle of a sine wave. The peak amplitude and duration of the artifacts are variables. The duration of this kind of noise signal is 100-500 ms with amplitude of 500% peak-to-peak ECG amplitude.

Muscle Contraction: the muscle movements cause generation of artifactual millivolt level potentials. It can be assumed to be transient burst of zero mean band limited Gaussian noise. The variance of the distribution may be estimated from the variation and duration of the bursts. Standard deviation of this kind of noise is 10% of peak-to-peak ECG amplitude with duration of 50 ms with a frequency content from dc to 10 kHz.

---

[3] We refer to [ASSK07, ch.2] for a detailed description

Base Line Wander: the baseline wander of the ECG signals causes problems in the detection of peaks. For example, due to the wander, the T peak could be higher than the R peak, and be detected as an R peak. Low frequency wander of the ECG signal can be caused by breathing or patient movement. The drift of the baseline with breathing can be represented as a sinusoidal component and the frequency of respiration added to the ECG signal. The variation could be reproduced by amplitude modulation of the ECG by the sinusoidal component that is added to the base line. The amplitude variation is 15% of peak-to-peak ECG amplitude and the base line variation is 15% of ECG amplitude at 0.15 to 0.3 Hz. These noise should be removed from ECG before extracting the characteristic features. Noise removal is accomplished by passing the cardiovascular signals through a filter whose cutoff frequency is a function of the noise frequency.

10.2 SNR Evaluation in the Encrypted Domain

In this section we describe the s.p.e.d. protocol implementing the quality evaluation scheme depicted in Figure 10.1. We remind that s.p.e.d. protocols are able only to perform integer operations, hence in the following all the functions that we evaluate operate on integers. In the following, we assume that the ECG signal has already been preprocessed to remove baseline wander and power line interference, for example, by using the filter proposed in [VAS85].

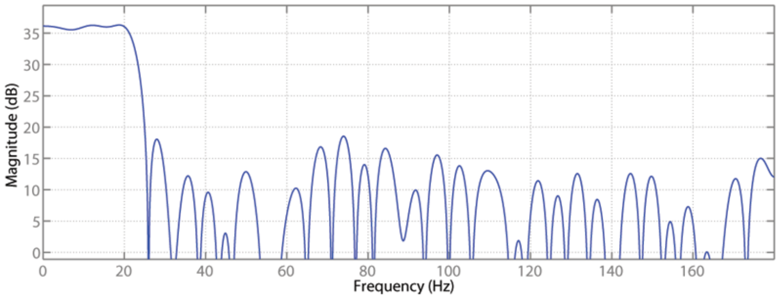
Filtering: Indeed in the ECG case the original signal is not even definable, however the amount of noise contained in frequency bands usually not occupied by a clean ECG signal may be considered as a measure of the quality of the signal itself. In a no-reference scenario the original signal is not available, as in the case of an ECG where only the recorded signal $\mathbf{y}$ is available. As we outlined in the introduction, our quality measure works by *estimating* the denoised signal $\mathbf{x}$ by applying a filter to the signal $\mathbf{y}$. A denoising filter is a linear transform that can be applied in the encrypted domain by using homomorphic encryption as proposed in [BPB08b, BPB08c, BPB10] and summarized in Section 3.5.2. According to the recommendations made by the American Heart Association for ECG recordings, in the lower frequency region, the frequency components between above 0.5 Hz and 20 Hz bring important information relative to patient health and hence should not be removed. The frequencies below 0.5 Hz are removed by the filter proposed in [VAS85]. In our scheme, we consider the amount of noise at frequencies greater than 20 Hz as a measure of the quality of the signal. For this reason, the filter in Figure 10.1 is an integer low-pass filter with cut off frequency $f_c = 20$ Hz. The filter has been optimized to obtain a small number of coefficients that can be represented with few bits. In the end, we obtained a $k_c = 82$ order integer FIR filter having amplification factor $amp = 64$ and 47 non-zero coefficients. The maximum filter coefficient is 8 and the coefficients assume values in the set $\mathcal{K} = \{0, 1, 2, 4, 5, 6, 7, 8\}$. The spectrum of the filter is shown in Figure 10.2, together with its impulse response.

Having the filter even real magnitude response, it has symmetric coefficients, i.e. $c_i = c_{k_c - i}$, hence a filtered sample can be computed as $x_i = c_0 y_i + \sum_{j=1}^{k_c/2}(c_j y_{i+j} + c_j y_{i-j})$. Note that only c_0 and other 23 coefficients are not null. To filter the signal we can first compute each product $[\![c_j y_i]\!]$ $\forall i = 1, \ldots, k$ $\forall j = 0, \ldots, k_c/2$ as

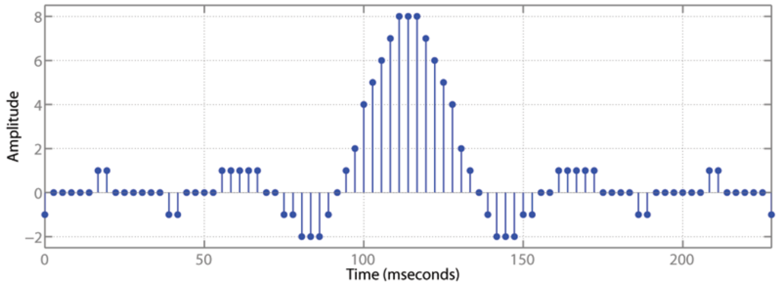
$$[\![y_i c_j]\!] = \begin{cases} 1 & \text{if } c_j = 0, \\ [\![y_i]\!] & \text{if } c_j = 1, \\ [\![y_i]\!]^{c_j} & \text{else.} \end{cases} \tag{10.1}$$

Since there are only 6 non-null and non-unitary coefficients, only $6k$ products are computed. Then the filtered samples are obtained as $[\![x_i]\!] = [\![c_0 y_i]\!] \prod_{j=1}^{k_c/2}([\![c_j y_{i+j}]\!][\![c_j y_{i-j}]\!])$ $\forall i = 1, \ldots, k$. If $i - j < 1$ we assume $y_{i-j} = y_1$, while if $i + j > k$ we assume $y_{i+j} = y_k$.

Noise Computation: The noise signal $\mathbf{n}$ can be computed by subtracting $\mathbf{y}$ and $\mathbf{x}$ sample-wise ($n_i = y_i - x_i$). The operation can be performed by the server in the encrypted domain: $[\![n_i]\!] = [\![y_i - x_i]\!] = [\![y_i]\!] * [\![x_i]\!]^{-1}$ $\forall i = 1..k$. Since the integer filter is applied in the encrypted domain, we have to consider that it may return a filtered signal amplified by a factor amp. Thus, to correctly derive the noisy part of the signal, we have



Spectrum of the filter(Magnitude response)



Impulse response

Figure 10.2: Filter plots

to multiply **y** by the same factor, obtaining $[\![n_i]\!] = [\![amp * y_i - x_i]\!] = [\![y_i]\!]^{amp} * [\![x_i]\!]^{-1}$ $\forall i = 1..k$. The amplification factor can be obtained observing the magnitude plot of the filter.

ENERGY COMPUTATION: To compute the SNR we must evaluate the energy of **x** and **n**. This can be done by using the interactive protocol shown in Section 3.4.10 that, applied to **x** and **n**, returns $[\![E_x]\!]$ and $[\![E_n]\!]$.

SNR COMPUTATION: The final step of the quality evaluation procedure is the computation of the SNR

$$SNR = 10 * \log_{10} \frac{E_x}{E_n} = \frac{10}{\log_2 10} * \log_2 \frac{E_x}{E_n}. \tag{10.2}$$

Notice that the factor $10/\log_2 10$ is just a constant and therefore it can be neglected. The main observation our scheme relies on is that in a s.p.e.d. framework, it is easier to compute the difference of two logarithms than the logarithm of a ratio, which would involve a division circuit. Hence we are going to compute

$$SNR = \log_2 \frac{E_x}{E_n} = \log_2 E_x - \log_2 E_n, \tag{10.3}$$

where E_x and E_n are available in encrypted form. Note that while (10.3) is an exact equality when dealing with real numbers, in our case this is only an approximated relationship since we are using an integer logarithm. Furthermore, observe that computing the integer logarithm (base 2) of a binary positive integer number is equivalent to detecting the minimum number of bits necessary to represent the number (in our case the energy is positive by definition). We decided to compute the SNR as given in (10.3) using a GC.

To switch from HE to GC, the interface described in Section 8.1.2 is used: $\mathcal{S}$ obfuscates E_x and E_n and sends them to $\mathcal{C}$ that, after decryption, uses them as inputs to a GC prepared by $\mathcal{S}$. The Boolean circuit starts by removing the obfuscations from the energies, then it evaluates the logarithms, as described in Section 5.3.16 (the integer logarithm has been chosen, instead of other solutions, for its low complexity) and finally computes the SNR by subtracting the result of the logarithms. We can optimize the protocol by observing that after having evaluated the energy of an obfuscated signal, $\mathcal{C}$ sends it to $\mathcal{S}$ to remove the total obfuscation r_x introduced during the energy computation and then applies a new obfuscation to switch to GC and sends back the value with a new obfuscation to $\mathcal{C}$. To reduce the number of communication rounds, $\mathcal{S}$ can transmit $[\![\sum_i 2x_i r_{x,i} + r'_x]\!] = [\![r'_x]\!] * \prod_i [\![x_i]\!]^{2r_{x,i}}$ to $\mathcal{C}$ during the computation of E_x, for an appropriate[4] value r'_x. $\mathcal{C}$ decrypts the value and subtracts it from the obfuscated energy, obtaining a energy still obfuscated by the value $\sum_i r_{x,i}^2 - r'_x$. This value can be considered

---

[4] The size of r'_x has to be larger than $\sum_i 2x_i r_{x,i}$ by at least a factor of 2^κ to guarantee statistical indistinguishability at the same level as the rest of the protocol.

the obfuscation that has to be removed by the GC. The term r'_x is introduced to avoid any leakage of information when $\sum_i 2x_i r_{x,i}$ is revealed to the client. Clearly, the reduction in round complexity comes at the cost of additional data being transmitted per round and at the cost of additional circuit complexity. The same optimization can be applied to the computation of $\log_2 E_n$.

As shown in Section 5.3.16 the logarithm computation is performed in two steps: in the first one the input value is processed so that it has zero values before the most significant 1 and a sequence of 1 in the remaining bits, in the second one the bits equal to 1 are counted. After having applied the protocol proposed to both the energy E_x and E_n it would be sufficient to compute the difference between $\log_2 E_x$ and $\log_2 E_n$ to obtain the SNR. This could be easily done still using a GC. Note that the protocol proposed returns $\lfloor \log_2 a \rfloor + 1$, but after the subtraction the two $+1$ terms are discarded. A more efficient implementation (using only one counter) can be obtained as follows. Given the two energies E_x and E_n we apply the first part of the protocol to both values obtaining $c(E_x)$ and $c(E_n)$. Evaluating $\mathsf{COUNT}(c(E_x) \oplus c(E_n))$ we obtain $|\log_2(E_x) - \log_2(E_n)|$. The result of the XOR is a binary string containing a number of 1's equal to the result of the difference.

As an example, let suppose to have the following inputs: $E_x = 22 = (00010110)_2$ and $E_n = 9 \;\; = (00001001)_2$ whose SNR results $SNR = \lfloor \log_2(22) \rfloor + 1 - \lfloor \log_2(9) \rfloor - 1 = 5 - 4 = 1$. Once $c(E_x) = (00011111)_2$ and $c(E_y) = (00001111)_2$ have been obtained, we can evaluate $SNR = \mathsf{COUNT}(c(E_x) \oplus c(E_n)) = 1$ (note that $c(E_x) \oplus c(E_n) = (00010000)_2$), instead of counting them and computing the difference.

The only thing that we still need to do is to compute the sign of the SNR for which it is sufficient to evaluate the relation $E_x < E_n$ which has the same complexity as a subtraction circuit.

The full protocol that, given a signal $\mathbf{x}$, filters it and then compute the SNR is shown in Protocol 25.

HYBRID SNR COMPUTATION PROTOCOL

inputs of $\mathcal{C}$: $\mathbf{y} = [y_1, \ldots, y_k]$
inputs of $\mathcal{S}$: $\mathbf{c} = [c_1, \ldots, c_{k_c}],\ c_j \in \mathcal{K},\ amp$
output for $\mathcal{C}$: *SNR*
output for $\mathcal{S}$: nothing

client $\mathcal{C}$ server $\mathcal{S}$

encrypts $y_i\ \forall i \in 1 \ldots k$;

$$\xrightarrow{[\![y_i]\!]\ \forall i \in 1 \ldots k}$$

[FILTERING]
$\forall i = 1, \ldots, k;\ \forall c_j \in \mathcal{K}:$
$[\![y_i c_j]\!]$ computed according to (10.1)
$\forall i = 1, \ldots, k:$
$$[\![x_i]\!] = [\![c_0 y_i + \sum_{j=1}^{k_c/2} (c_j y_{i+j} + c_j y_{i-j})]\!]$$
$$= [\![c_0 y_i]\!] \prod_{j=1}^{k_c/2} ([\![c_j y_{i+j}]\!][\![c_j y_{i-j}]\!])$$
$(y_{i-j} = y_1$ if $i - j < 1)$;
$(y_{i+j} = y_k$ if $i + j > k)$;
[NOISE COMPUTATION]
$\forall i = 1, \ldots, k:$
$$[\![n_i]\!] = [\![amp y_i - x_i]\!] = [\![y_i]\!]^{amp}[\![x_i]\!]^{-1};$$
[BLINDING]
$\forall i = 1, \ldots, k:$
choose $r_{x,i}, r_{n,i} \in \mathbb{Z}_{2^{\ell_n+\kappa}}$;
$[\![x'_i]\!] = [\![x_i + r_{x,i}]\!] = [\![x_i]\!][\![r_{x,i}]\!]$;
$[\![n'_i]\!] = [\![n_i + r_{n,i}]\!] = [\![n_i]\!][\![r_{n,i}]\!]$;
$[\![r'_x + \sum_{i=1}^{k} 2r_{x,i}x_i]\!] = [\![r'_x]\!] \prod_{i=1}^{k} [\![x_i]\!]^{2r_{x,i}}$;
$[\![r'_n + \sum_{i=1}^{k} 2r_{n,i}n_i]\!] = [\![r'_n]\!] \prod_{i=1}^{k} [\![n_i]\!]^{2r_{n,i}}$;

$$\xleftarrow{[\![x'_i]\!], [\![n'_i]\!]\ \forall i = 1, \ldots, k, [\![r'_x + \sum_{i=1}^{k} 2r_{x,i}x_i]\!], [\![r'_n + \sum_{i=1}^{k} 2r_{n,i}n_i]\!]}$$

decrypts cyphertexts;
[ENERGY COMPUTATION]
$E'_x = \sum_{i=1}^{k} {x'_i}^2 - r'_x - \sum_{i=1}^{k} 2r_{x,i}x_i$;
$E'_n = \sum_{i=1}^{k} {n'_i}^2 - r'_n - \sum_{i=1}^{k} 2r_{n,i}n_i$;

$\downarrow E'_x \quad \downarrow E'_n \qquad \downarrow \sum_i r_{x,i}^2 - r'_x \quad \downarrow \sum_i r_{n,i}^2 - r'_n$

$$\boxed{\text{GC(2SUB, } 2\ell_e - 2 \text{ OR, CMP, COUNT)}}$$

$\downarrow$ *SNR*

Protocol 25: Hybrid SNR computation protocol (packing and bitlengths are omitted for simplicity).

10.3 Protocol Analysis

In this section we analyze the complexity of the protocol proposed. Such complexity depends on the number of bits that are required to represent correctly the data to be analyzed, the final result and all the intermediate values involved in the computation. For this reason, we start our analysis by evaluating the number of bits necessary to carry out all the steps of the protocol as a function of the bitsize of the input data. Such an analysis is then applied to the case of ECG signals drawn from the MIT-BIH Arrhythmia Database. This allow us to derive a precise value for the communication complexity of our protocol in a real life scenario. Given that the protocol can run only on integer numbers, we assume that the input signal samples have already been quantized and hence are represented by integer numbers. Samples in the PhysioBank archives are already available in quantized form. Then we analyze the error introduced by the use of integer functions and we conclude with a simple analysis of the security.

10.3.1 Bitsize analysis

The number of bits used to represent the data influences the complexity of the secure protocol for two main reasons. In the case of garbled circuits, all values are encrypted bitwise. Hence the greater the number of bits, the more complex the resulting protocol will be. With regard to computing on homomorphically encrypted data, we observe that after each multiplication or square computation carried out in the encrypted domain, the number of bits necessary to represent the output of the multiplication doubles. Thus, it is necessary that the plaintext space supported by the HE cryptosystem is large enough to represent the output of the computations without an overflow, which would cause an invalid result. To start the analysis, let's assume that $\mathcal{S}$ has k integer samples of the signal $\mathbf{y} = [y_1^{(\ell_y+1)}, \dots, y_k^{(\ell_y+1)}]$ represented with ℓ_y bits for the magnitude and one for the sign. The maximum possible value that $\mathbf{y}$ may assume being $max_y = 2^{\ell_y} - 1$.

Filtering: As a first step $\mathbf{y}$ is filtered to produce $\mathbf{x}$. This is done by applying an integer filter with k_c coefficients $c_j^{(\ell_c+1)}$ each represented with ℓ_c bits for the magnitude and 1 for the sign. The result of each product during the computations needs $\ell_y + \ell_c + 1$ bits. Since the maximum value that a sample can assume is $max_y = 2^{\ell_y} - 1$, it follows that the maximum value that a filtered sample can assume is $max_x = \sum_{j=1}^{k_c} |c_j * max_y| = max_y * \sum_{j=1}^{k_c} |c_j|$. c_j can be considered instead of max_c because they are well known to $\mathcal{S}$ that is carrying out the computation by using homomorphic encryption. Thus, the number of bits necessary to represent a filtered sample is $\ell_x = \ell_y + \lceil \log_2(\sum_{j=1}^{k_c} |c_j|) \rceil$ bits for the magnitude and one bit for the sign. Since $x_i^{(\ell_x+1)}$ is amplified with respect to $y_i^{(\ell_y+1)}$ by a factor amp, we have to amplify $y_i^{(\ell_y+1)}$ by the same factor to proceed with the computation. The amplified $y_i^{(\ell_y+1)}$ requires $\ell_y + \lceil \log_2 amp \rceil$ bits for the magnitude and one for the sign, but the amplification factor scales $y_i^{(\ell_y+1)}$ to have a value with the same order of magnitude as $x_i^{(\ell_x+1)}$, hence we can represent it with $\ell_x + 1$ bits.

NOISE: We can observe that $|n_i| = |amp * y_i - x_i| \leq amp * |y_i| + |x_i|$, hence the maximum value that the noise can assume is $max_n = amp * max_y + max_x$ and the representation of the noise needs $\ell_n = \ell_x + 1$ bits for the magnitude and the sign.

ENERGY: For the energy, the worst case is when we have a signal that assumes only the maximum value. In this case the maximum value of the energy is $max_{E_n} = \sum_{i=1}^{k} max_n^2 = k * max_n^2$. From the previous considerations we obtain that the number of bits necessary to the energy representation is ℓ_e where $\ell_e = \lceil \log_2 max_{E_n} \rceil$. The sign of the energy is positive, hence the additional sign bit for it is not needed. Being max_x smaller than max_n, max_{E_x} is smaller than max_{E_n}. Anyway we can set $max_{E_x} = max_{E_n} = max_E$ for simplicity.

SNR COMPUTATION: Finally, the SNR is computed as $\log_2 E_x - \log_2 E_n$. Observe that since both E_x and E_n are positive integers, the maximum value of the difference is (in magnitude) the same as the logarithm of the two inputs ($max_{SNR} = \lceil \log_2(max_E) \rceil$), hence the number of bits necessary to represent the difference is given by $\ell_{SNR} = \lceil \log_2(\ell_e) \rceil$ bits for the magnitude and one for the sign.

Application to real ECG signals In the Physiobank database each sample y_i is represented with $\ell_y + 1 = 11$ bits, hence the maximum value it can assume is $max_y = 2^{10} - 1 = 1023$. The filter we designed has a maximum absolute value $max_c = 8$ that can be represented with $\ell_c + 1 = 5$ bits. The sum of the coefficients is $\sum_{j=1}^{k_c} |c_j| = 112$ and $amp = 64$. We consider to compute the SNR of 30 seconds of signal, sampled at 360 HZ (totally $k = 10800$ samples)[5]. Given these parameters, we can compute the bitsize of all the data involved in the computation as shown in Table 10.1.

Variable name	Maximum value	Magnitude bitlength
Original sample y_i	1023	$\ell_y = 10$
Filter Coefficients c_j	8	$\ell_c = 4$
Filtered sample x_i	114576	$\ell_x = 17$
Noise Signal n	180048	$\ell_n = 18$
Signal Energy $E_x,\ E_n$	350106648883200	$\ell_E = 49$
SNR SNR	49	$\ell_{SNR} = 6$

Table 10.1: Number of bits necessary to represent the values obtained by a worst case analysis.

10.3.2 Communication complexity

To evaluate the complexity of the protocol, we focus on the part that requires interaction. At the beginning $\mathcal{C}$ transmits k cyphertexts to $\mathcal{S}$ one for each sample. Computed $[\![x_i]\!]$ and $[\![n_i]\!]$, $\forall i = 1..k$, the computation of the SNR requires 1 move for the HE part plus those of the underlying OT protocol. We denote the asymmetric security parameter used in the HE section with T and the symmetric security parameter used in the GC section with t.

In the interface between HE and GC, $\mathcal{S}$ has to transmit the obfuscated $[\![x_i + r_{x,i}]\!], [\![n_i + r_{n,i}]\!]$, $\forall i = 1..k$ and the two terms $[\![r'_x + 2\sum_i x_i r_{x,i}]\!]$, $[\![r'_n + 2\sum_i n_i r_{n,i}]\!]$. If we assume to represent the blinding terms $r_{x,i}$ and $r_{n,i}$ with

---

[5] This choice is motivated by the fact that 30 seconds are sufficient for the SNR evaluation and this interval does not introduce a big delay in further computation.

$\ell_o + 1 = \ell_n + 1 + \kappa$ bits, each term $x_i + r_{x,i}$ and $n_i + r_{n,i}$ can be represented with $\ell_o + 2$ bits and $m = \lfloor T/(\ell_o + 2) \rfloor$ values can be packed together, hence the protocol needs to transmit

$$n_p = \lceil 2 * k/m \rceil$$

cyphertexts of size $2T$ bits. 4 rounds are necessary in total.

As we can easily observe, a high number of cyphertexts is transmitted. The communication complexity can be reduced by using the composite signal representation of Section 3.5.2, but all the improvements introduced in Protocol 25 to decrease the number of products to compute can not be used. The samples recorded by the electrocardiograph can be packed in n_p samples so that the final result of the filtering and obfuscation is correctly represented. In this way the bandwidth of the HE part is reduced from $k + n_p + 1$ to $2n_p + 1$ cyphertexts. In the following we assume that packing is used.

The two terms $[\![2 \sum_{i=1}^{k} r_{x,i} x_i]\!]$,$[\![2 \sum_{i=1}^{k} r_{n,i} n_i]\!]$ require $\ell_o + \ell_n + \lceil \log_2 k \rceil + 2$ bits for their representation. The complexity of the GC is linear. From Section 10.2 we can see that it has complexity $\mathcal{O}(\ell_e)$. Now we want to analyze the exact amount of bits transferred from the protocol. By generating r'_x and r'_n to hide statistically the other values, r'_x needs

$$\ell_r = \ell_o + \ell_n + \lceil \log_2 k \rceil + \kappa + 1 + 1$$

bits and we obtain two obfuscation values for the energies that require $\ell_o + \ell_n + \lceil \log_2 k \rceil + 3$ bits, and they can be packed together in another cyphertext if $\ell_r < T/2$.

In the GC part, $\mathcal{S}$ has to send the secrets relative to the ℓ_e bits of the obfuscated E_x and E_n values to $\mathcal{C}$ through the OT and the secrets relative to the ℓ_e bits of the obfuscation values $\sum_i r_{x,i}^2 + r'_x$ and $\sum_i r_{n,i}^2 + r'_n$ together with the Garbled Circuit[6]. Considering that $2\ell_e > t$ the protocol of Section 4.4 is used. t OTs are performed offline and then extended to $2\ell_e$ OTs with offline communication complexity $6t^2 + 8\ell_e t$ bits. The online phase of the OT needs the transmission of $2 * (2\ell_e)t$ bits. $\mathcal{S}$ uses the two obfuscation values

---

[6] After the subtraction of the ℓ_e least significant bits the magnitude of each energy is obtained. Considering that energy is positive, continuing with the evaluation to obtain the carry (sign) bit is not necessary.

as server input to the GC and has to transmit ℓ_e secrets of size t to $\mathcal{C}$ for each of them. The two subtraction circuits that remove the obfuscation have $2\ell_e$ non-XOR gates. Moreover the circuit computing the SNR is composed by $2(\ell_e-1)+\ell_e-\lfloor\log_2(\ell_e+1)\rfloor+\ell_e$ non-XOR gates. Hence the GC that $\mathcal{S}$ has to transmit (online or offline) to $\mathcal{C}$ is characterized by $6\ell_e-\lfloor\log_2(\ell_e+1)\rfloor-2$ tables of size $4t$.

Table 10.2 shows the amount of data transfered by the protocol in a real case, where we are going to evaluate the SNR of 30 seconds ($k=10800$ samples) of ECG, considering short term security ($t=80$, $T=1248$ and $\kappa=80$), offline circuit transmission and that in the HE section the two obfuscation values require $\ell_r=212$ bits and can be packed together.

Section	Offline		Online	
	Complexity	Bits	Complexity	Bits
HE			$(2n_p+1)2T$	$8,988,096$
$\mathcal{C}$ input secrets (OT)	$6t^2+8\ell_e t$	70,400	$4\ell_e t$	$16,000$
$\mathcal{S}$ input secrets			$2\ell_e t$	$8,000$
Garbled Tables	$6\ell_e-\lfloor\log_2(\ell_e+1)\rfloor-2$	$94,080$		
Total		164,480		9,012,096

Table 10.2: Communication complexity of SNR protocol.

10.4 Error Analysis

Working with an integer filter applied to integer samples, errors in the protocol are introduced only by the quantization process inherent in the measurement and by the use of the integer logarithm. We now analyze the magnitude of such an error by considering the choice we made to compute the logarithm of a ratio as the difference of two logarithms, that is we compute the error we make when we substitute $\log_2(a/b)$ with $\lfloor\log_2 a\rfloor-\lfloor\log_2 b\rfloor$, where $a,b\in\mathbb{N}^+$. We opted for such a choice since computing the inversion in the encrypted domain is very expensive: we now evaluate the appropriateness of such a choice from an accuracy point of view. Let $\epsilon\in[0,1)$ be the error introduced by each floor operation with respect to the real logarithm. The total error results:

$$\begin{aligned}\epsilon_{tot} &= |\log(a/b) - \lfloor \log_2 a \rfloor + \lfloor \log_2 b \rfloor| \\ &= |\log(a/b) - \log_2 a + \epsilon_a + \log_2 b - \epsilon_b| = |\epsilon_a - \epsilon_b| < 1.\end{aligned}$$

To compare our scheme with a protocol based on the application of the logarithm directly to the ratio a/b , we suppose to have an efficient implementation of the division in the encrypted domain that returns $\lfloor a/b \rfloor$. Considering that $a, b \in \mathbb{N}^+$ then $\lfloor \log_2 \lfloor a/b \rfloor \rfloor = \lfloor \log_2 a/b \rfloor$ yielding

$$\epsilon_{tot} = \log(a/b) - \lfloor \log_2 \lfloor a/b \rfloor \rfloor = \log(a/b) - \lfloor \log_2 a/b \rfloor \in [0, 1).$$

Note that in this case we have a biased error whose mean value is different than zero. We evaluated experimentally the errors in both cases by performing actual tests on the ECG signals from the Physionet database in the plain domain and simulating the proposed protocol as well as a protocol based on division. The error between the integer and the real implementation is distributed as in Figure 10.3. The mean absolute error in the proposed protocol is 0.33, while the mean absolute relative error obtained using the division-based protocol is 0.5. Evidently, the application of the floor operator twice in the division-based protocol leads to a larger error.

We can conclude that the implementation of the integer SNR by using the difference of the logarithm is easy to compute and introduces a smaller error.

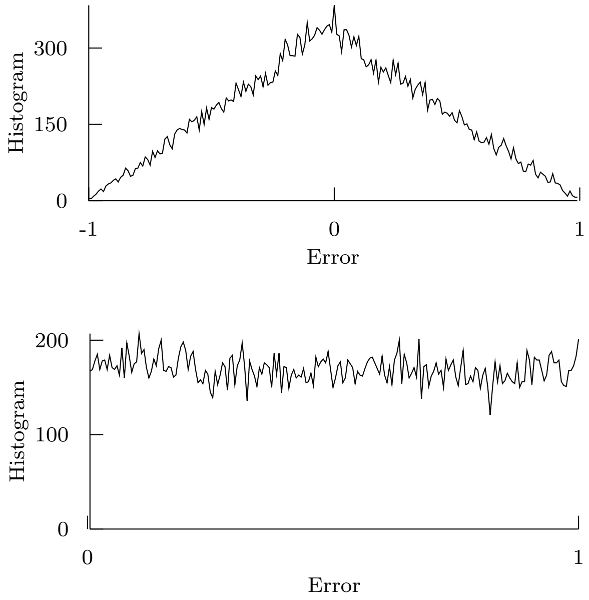


Figure 10.3: Error incurred by using the integer logarithm instead of the real logarithm in the proposed protocol (a) and in the protocol based on division (b). Tests performed on ∼30000 random couples of values.

Chapter 11

Privacy Preserving Quality Evaluation

As already said in Chapter 7, different types of information can be extracted from different parts of the ECG. Moreover the noise affecting the signal can permit some evaluations, while impeding others. For example, to evaluate the heart rate or some arrhythmias, it is sufficient to identify the R peaks (the highest peaks of the QRS complexes). This is a simple operation that can fail only in the presence of very strong noise. On the contrary, atrial flutter can be identified by a "saw-tooth" effect and to detect an ischemia it is necessary to evaluate the slope of the ST segment and the T wave. In these examples even a small quantity of noise can compromise the evaluation.

In this chapter, we describe a methodology which allows the service provider to obtain a quality measure guaranteeing that the quality of the input to later computations is good enough to allow a reliable analysis. In particular, we propose a protocol that allows to evaluate the quality of ECG signals and communicates to the user whether the input signal is good enough to continue with the analysis or not [LGB12]. In case it is not, a new measurement and re-sending of the signal is requested. The protocol has been designed so that it can be easily implemented in a privacy preserving manner that permits the client $\mathcal{C}$ not to reveal his data to the server $\mathcal{S}$, yet allowing the server to perform the analysis of the ECG quality without revealing the private parameters (e.g., the filter coefficients) of its algorithm. Both the client and the server can be interested to obtain as much information as possible, but they are not interested to deviate from the protocol, being the patient's health involved. Hence the semi-honest model seems appropriate.

The system we are proposing evaluates ECG signal quality by identifying the presence of electrode contact noise. Our purpose is to provide a solution allowing a simple implementation in the encrypted domain, even if not optimal from the accuracy point of view.

Let consider that an ECG signal may have enough quality to allow the evaluation of a particular function or the extraction of particular information, but it may also have bad quality not allowing information extraction, or sufficient quality for certain analysis/applications but not for others. Thus, the quality of an ECG signal is application-dependent. Existing solutions are generally too complex to allow an efficient implementation in the encrypted domain.

In Chapter 10 we have already presented an hybrid protocol evaluating the SNR of a signal. The problem is that usually the correlation of SNR with the quality is very poor and it is of little interest as a general objective measure of signal quality.

The main idea is to extend the method proposed by using a segmented SNR instead of the overall signal SNR. In doing so, we extend the approach proposed in Chapter 10, following the idea described in [HP98], where a segment based SNR is used to assess *speech* signal quality. In particular, the same idea can be applied to many other types of signals having a *fixed range of frequencies* that can be affected by burst of noise, such as videos, ECGs, etc. The method in [HP98] subdivides the signal into small segments and, for each of them, the SNR is computed. Finally, the mean and variance of the segmented SNR are computed. The use of the variance is justified by the observation that while the electrode contact noise has a minor impact on the mean SNR, an occasional burst of noise having small time length can be better detected by examining the SNR variance. Hence both the mean and the variance of the segmented SNR can be used to evaluate the quality of a signal, together with other features (here we consider the SNR of the whole signal) to obtain a more accurate analysis. This methodology is validated through multiple tests in which we observe that the segmental SNR reflects the signal quality better than the SNR alone. Notice that in [HP98], the authors propose additional “plain” techniques to assess the quality of a speech signal, but these would be too computationally intensive to be efficiently implemented in the encrypted domain.

The chapter is organized in the following way: in Section 11.1 the protocol is described, while in Section 11.2, we analyze the dimension of the data involved in the computation and the communication complexity of the protocol, moreover the results of experiments conducted on ECG signals from the MIT-Arrhythmia database in terms of classification accuracy are presented.

11.1 Protocol description

In the following, we describe the steps necessary to reach a decision based on a set of features (SNR, segment SNR mean, segment SNR variance) extracted during the privacy preserving protocol. The plain implementation of the protocol is summarized in Figure 11.1. In the remainder of this chapter, we assume to evaluate τ seconds of an ECG signal $\mathbf{s} = \{s_1, \dots, s_{\tau f_s}\}$, where f_s is the sampling frequency and each sample is represented with $\ell_s + 1$ bits (ℓ_s bits for the magnitude and 1 for the sign).

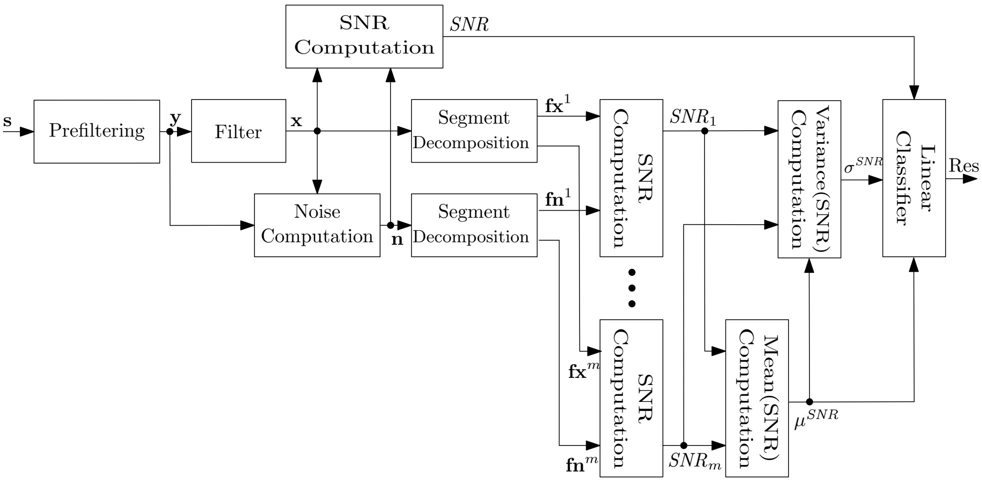


Figure 11.1: Sequence of steps performed to evaluate the quality of an ECG signal.

Filtering and noise computation The protocol starts acting as described in Chapter 10. The client first removes the base-line wander and the power-line interference by using the filter proposed in [VAS85], producing a filtered signal $\mathbf{y} = \{y_1, \dots, y_{\tau f_s}\}$ that, after encryption, is transferred to $\mathcal{S}$. The signal $[\![\mathbf{y}]\!]$ is filtered again by $\mathcal{S}$ to produce the signal $[\![\mathbf{x}]\!]$. This HE subprotocol involves the filter developed by $\mathcal{S}$ that is interested in protecting his intellectual property. Finally $\mathcal{S}$ evaluates the noise signal $[\![\mathbf{n}]\!]$, i.e. the encryption of the difference between the signals $\mathbf{x}$ and $\mathbf{y}$, still using HE.

Energy and SNR Evaluation We subdivide the signals $[\![\mathbf{x}]\!]$ and $[\![\mathbf{n}]\!]$ into segments of w samples, obtaining $m = \lfloor \tau f_s / w \rfloor$ segments $\mathbf{fx} = \{\mathbf{fx}^1 \dots \mathbf{fx}^m\}$ and $\mathbf{fn} = \{\mathbf{fn}^1 \dots \mathbf{fn}^m\}$, where each segment is composed by the encryption of w samples. For simplicity we assume that $\tau f_s = mw$. Otherwise, if $\tau f_s \neq mw$, the last segment, having less than w samples, is discarded. For each pair of signal and noise segments $(\mathbf{fx}^i, \mathbf{fn}^i)$ the SNR should be evaluated as the power ratio between the signal and the noise in the logarithmic decibel scale:

$$SNR_i^f = 10 \log_{10} \frac{\frac{\left(\sum_{j=1}^{w} (fx_j^i)^2\right)}{w}}{\frac{\left(\sum_{j=1}^{w} (fn_j^i)^2\right)}{w}} = \frac{10}{\log_2 10} \log_2 \frac{\sum_{j=1}^{w} (fx_j^i)^2}{\sum_{j=1}^{w} (fn_j^i)^2}.$$

As already said, $10/\log_2 10$ only amplifies the result and therefore it can be neglected. Hence the SNRs we are going to evaluate in the encrypted domain are

$$\begin{aligned} SNR_i^f &= \log_2 \frac{\sum_{j=1}^{w} (fx_j^i)^2}{\sum_{j=1}^{w} (fn_j^i)^2} = \log_2 \frac{E_{fx^i}}{E_{fn^i}} \\ &= \log_2 E_{fx^i} - \log_2 E_{fn^i}, \end{aligned}$$

where E indicates the signal energy. Starting from the encryption of the samples composing the segment $\mathbf{fx}^i$ and $\mathbf{fn}^i$, the energies and SNR of the i-th segment are evaluated by using the hybrid protocol described in Chapter 10. The SNR^f values are not the final result, in fact they have to be used in later computation and hence kept secret. To avoid their disclosure to $\mathcal{C}$, the GC blinds them by adding random values r_{SNR_i} each $\ell_{SNR_f} + \kappa$ bits long, where ℓ_{SNR_f} is the bitlength of the segment SNR.

Note that the computation of the SNR of the whole signal can be obtained starting from the energy of the segments previously computed.

In fact

$$SNR = \log_2 \frac{\sum_{j=1}^{mw}(fx_j)^2}{\sum_{j=1}^{mw}(fn_j)^2} = \log_2 \frac{\sum_{i=1}^{m}\sum_{j=1}^{w}(fx_j^i)^2}{\sum_{i=1}^{m}\sum_{j=1}^{w}(fn_j^i)^2}$$
$$= \log_2(\sum_{i=1}^{m} E_{fx^i}) - \log_2(\sum_{i=1}^{m} E_{fn^i}).$$

$\mathcal{C}$ adds together all the segment energies E'_{fx^i} and all the segment energies E'_{fn^i}, obtaining the energies E'_x and E'_n of the whole signals, obfuscated by a value that is the sum of the obfuscation introduced in each segment energy, still known by $\mathcal{S}$. Finally, SNR is obtained by another circuit for SNR evaluation having a larger input bit-length and hence a larger number of gates.

Once all the obfuscated SNR_i^f and SNR values are obtained, $\mathcal{C}$ encrypts them and transmits the ciphertexts to $\mathcal{S}$, who can remove the obfuscation by using the homomorphic properties of the cryptosystem.

MEAN AND VARIANCE COMPUTATION Once $\mathcal{S}$ has obtained the encrypted values $[\![\mathbf{SNR^f}]\!] = \{[\![SNR_1^f]\!], \ldots, [\![SNR_m^f]\!]\}$, he can compute their mean and variance by using HE.

The computation of the SNRs mean would require division by m. If m is public (or known), this is a cheap operation. However, if it is a private value, the division operation can be expensive and requires an interactive protocol [Veu10, LB11]. An additional disadvantage is that this protocol would introduce a rounding error. An alternative is to avoid the division by m. Then, the mean is amplified by a factor of m, the accuracy is preserved and the complexity of the protocol is reduced since interaction with $\mathcal{C}$ is not necessary. The amplified mean is simply computed by $\mathcal{S}$ as $[\![\mu^{SNR}]\!] = [\![\sum_{i=1}^{m} SNR_i^f]\!] = \prod_{i=1}^{m}[\![SNR_i^f]\!]$.

The computation of the variance can be performed by using an interactive protocol. Given the amplified mean, the SNR^f values have to be amplified by the same factor before computing the variance. Moreover, following the same approach we used for the mean, division by m is avoided and thus, the resulting variance is amplified by a factor m^3. Considering that $\mathcal{C}$ already has the SNR_i^f values obfuscated with a random value r_{SNR_i}, $\mathcal{S}$ can send the obfuscated SNR mean $[\![\mu^{SNR} + r_\mu]\!]$ (where $r_\mu \in \mathbb{Z}_{\ell_\mu+\kappa}$, given the bitlength ℓ_μ of μ^{SNR}) to $\mathcal{C}$ who decrypts it and computes

$$\sum_{i=1}^{m} \Big(m(SNR_i^f + r_{SNR_i}) - (\mu^{SNR} + r_\mu) \Big)^2 .$$

Finally $\mathcal{C}$ encrypts the result and sends it back to $\mathcal{S}$ who removes the obfuscation value $\sum_i (m\, r_{SNR_i} - r_\mu)^2 - 2\sum_i (m\, SNR_i^f - \mu^{SNR})(m\, r_{SNR_i} - r_\mu)$ by using homomorphic properties, thus, obtaining the encrypted amplified variance $[\![\sigma^{SNR}]\!]$.

Classification Each feature previously computed (the overall SNR, the segment SNR mean and the segment SNR variance) can be used in a single-feature classifier that compares the feature with a threshold obtained by training. In the following, we show that by combining them, one can improve the accuracy of the whole classification by using a simple linear classifier. To do so, we developed a linear classifier that uses σ^{SNR}, μ^{SNR} and the SNR computed on the whole signal as input. In practice the signal quality is classified by evaluating the following inequality:

$$a + b\ \sigma^{SNR} + c\ \mu^{SNR} + d\ SNR > 0$$

where the coefficients a, b, c, d are obtained by training. If the inequality is verified, the signal is classified as noisy, otherwise as clean.

Training is carried out in the plain domain independently for each subject by using ECG recorded in a supervised environment so that the weight vector $\boldsymbol{\beta} = [a, b, c, d]$ can be obtained. Assume that a training set of k segments has been recorded from a patient and that from each segment j, a vector $\alpha_{\mathbf{j}} = [1, \sigma_j^{SNR}, \mu_j^{SNR}, SNR_i]$ is created in the plain domain, including an additional bit γ_j, indicating whether the signal has been classified as clean ($\gamma_j = -1$), or noisy ($\gamma_j = +1$). Then, we obtain,

$$a + b\ \sigma_j^{SNR} + c\ \mu_j^{SNR} + d\ SNR_j + \epsilon_j = \boldsymbol{\alpha}_j\ \boldsymbol{\beta} + \epsilon_j = \gamma_j$$

where ϵ_j is the error made by the classifier.

> By considering all the signals available for training, we compute the matrix $\mathbf{A} = [\alpha_1, \ldots, \alpha_k]^T$ and the vector $\mathbf{G} = [\gamma_1, \ldots, \gamma_k]^T$. The goal of the training is finding the vector $\boldsymbol{\beta}$ that minimizes the mean square error sum $\sum_{i=1}^{k} \epsilon_i^2$, which can be obtained as
>
> $$\boldsymbol{\beta} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{G}.$$
>
> Because $[a, b, c, d]$ are real values, they are quantized and represented with integer numbers to be used in the encrypted domain. Observe that the scalar product can be implemented by resorting to HE only. This assumes that the coefficients are known in plaintext form to $\mathcal{S}$, alternatively, a two party multiplication protocol can be performed similar to the computation of the variance. It is important to point out that the number of bits required to represent the data depends on many factors and it can change with each training set. In particular, the bitsize for the coefficients can be chosen so that the computation precision is similar to that of a plain implementation. In short, the classification is computed using HE:
>
> $$\begin{aligned} &[\![a + b\ \sigma^{SNR} + c\ \mu^{SNR} + d\ SNR]\!] = \\ &= [\![a]\!]\ [\![\sigma^{SNR}]\!]^b\ [\![\mu^{SNR}]\!]^c\ [\![SNR]\!]^d. \end{aligned}$$
>
> The sign of the scalar product is used for the classification, while the magnitude gives an indication of its reliability. High values are more reliable than small ones. Noticing that both sign and magnitude are useful for the classification and depending on the privacy requirements of $\mathcal{S}$ and $\mathcal{C}$, the protocol can choose to disclose the scalar product result to $\mathcal{C}$. Hence, $\mathcal{S}$ sends the encrypted magnitude and sign values to $\mathcal{C}$ who decrypts them and performs the final classification (determines to which class the particular measurement belongs). If $\mathcal{S}$ prefers to keep the magnitude secret, it can obfuscate the scalar product result by using a random value and then transmit a comparison GC having the obfuscated value and the random value as inputs.

The full privacy preserving protocol that evaluates the signal quality is shown in Protocol 26.

HYBRID PROTOCOL FOR ECG QUALITY EVALUATION

inputs of $\mathcal{C}$: $\mathbf{y} = [y_1, \dots, y_{\tau f_s}]$
inputs of $\mathcal{S}$: $\mathbf{c} = [c_1, \dots, c_{k_c}],\ c_j \in \mathcal{K},\ amp; [a, b, c, d]$
output for $\mathcal{C}$: cl
output for $\mathcal{S}$: nothing

client $\mathcal{C}$ server $\mathcal{S}$

[FILTERING, NOISE COMPUTATION]
computed according to Protocol 25;
[SEGMENT DECOMPOSITION]

[SEGMENTS SNR COMPUTATION]
$\downarrow r_{SNR_i} \forall i = 1 \dots m$

SNRs computation according to Protocol 25

$\downarrow SNR + r \downarrow SNR_i^f + r_{SNR_i} \forall i$
encrypts $y_i\ \forall i \in 1 \dots \tau f_s$;

$$\xrightarrow{\llbracket SNR + r \rrbracket, \llbracket SNR_i^f + r_{SNR_i} \rrbracket\ \forall i \in 1 \dots m}$$

$\llbracket SNR \rrbracket = \llbracket SNR + r \rrbracket \llbracket -r \rrbracket$;
[MEAN COMPUTATION]
$\llbracket \mu^{SNR} \rrbracket = \prod_{i=1}^{m} \llbracket SNR_i^f \rrbracket$;
[VARIANCE COMPUTATION]
choose $r_\mu \in \mathbb{Z}_{\ell_\mu + \kappa}$;
$\llbracket \mu^{SNR'} \rrbracket = \llbracket \mu^{SNR} \rrbracket \llbracket r_\mu \rrbracket$;

$$\xleftarrow{\llbracket \mu^{SNR'} \rrbracket}$$

decrypts $\mu^{SNR'}$;
$\sigma^{SNR'} =$
$= \sum_{i=1}^{m} \left(m(SNR_i^f + r_{SNR_i}) - \mu^{SNR'} \right)^2$;
encrypts $\sigma^{SNR'}$;

$\llbracket r^{SNR} \rrbracket =$
$\llbracket \sum_i (m\ r_{SNR_i} - r_\mu)^2 \rrbracket *$
$\prod_i (\llbracket SNR_i^f \rrbracket^m \llbracket \mu^{SNR} \rrbracket^{-1})^{-2(m\ r_{SNR_i} - r_\mu)}$;

$$\xrightarrow{\llbracket \sigma^{SNR'} \rrbracket}$$

$\llbracket \sigma^{SNR} \rrbracket = \llbracket \sigma^{SNR'} \rrbracket \llbracket r^{SNR} \rrbracket^{-1}$;
[CLASSIFICATION]
$\llbracket cl \rrbracket = \llbracket a \rrbracket\ \llbracket \sigma^{SNR} \rrbracket^b\ \llbracket \mu^{SNR} \rrbracket^c\ \llbracket SNR \rrbracket^d$;

$$\xleftarrow{\llbracket cl \rrbracket}$$

decrypt cl.

Protocol 26: Hybrid protocol for ECG quality evaluation (packing and bitlengths are omitted for simplicity).

11.2 Analysis

In a real implementation, the system has to be trained on patient's data in a supervised environment. A nurse or a doctor prepares a training set with intervals of clean ECG obtained with electrodes correctly and wrongly placed by them and by the patient respectively. To increase the data set, simulated noise can be added with different power levels to the whole clean signals or only in small random intervals. Finally, the nurse splits the ECG signals in clean and noisy signals. The subdivision can be made as a function of subjective parameters (an expert decides if the quality is sufficient or not) or testing the segments with the software that will be later used for the analysis of the ECG, when the software returns the answer expected by the expert, the signal is classified as clean, otherwise as noisy. The obtained data set is used to train a classifier that is then used to evaluate the signals that the patient will record at home connecting the electrodes without the presence of an expert.

We used the MIT-BIH Arrhythmia Database[1] for our experiments. The signals were divided into intervals and classified as noisy or clean according to the annotation available in the database, even if the type of noise was not specified explicitly. An interval is considered clean if no samples are affected by noise, otherwise it is considered noisy. To extend the data set we added artificial electrode contact noise stored in the MIT-BIH Noise Stress Test Database[2] to whole clean segments and partially (only to a randomly chosen section of the segment).

11.2.1 Data Dimension

To evaluate the quality of the signal we propose to analyze $\tau = 30$ seconds of the signal. This interval length is chosen because it is long enough to allow a correct evaluation and not so long to delay the beginning of the subsequent analysis, especially if the patient places the electrodes in a wrong way and has to evaluate the quality of the signal several times.

As already said in Chapter 10, the signals in the MIT-BIH Arrhythmia

---

[1] http://www.physionet.org/physiobank/database/mitdb/

[2] http://www.physionet.org/physiobank/database/nstdb/

database have a sample frequency $f_s = 360$Hz and are recorded in the U.S.A. (with a power-line frequency $f_p = 60$Hz). Each sample is represented with $\ell_s = 10$ bits for the magnitude and one bit for the sign, hence the maximum value it can assume is $max_s = 1023$. Since the signal $\mathbf{y}$ is obtained by processing $\mathbf{s}$ in the plain domain as proposed by Van Alsté and Schidler [VAS85], the signal is not amplified with respect to $\mathbf{s}$. Hence we can assume that the maximum value is $max_y = max_s = 1023$ and $\ell_y = 10$.

The filter owned by $\mathcal{S}$ is the integer low pass filter, described in Section 10.2. We recall that the amplification introduced by the integer filter is $amp = 64$, while the sum of the coefficients is $|c_0| + \sum_{j=1}^{k} |2 * c_j| = 112$. This sum is used to estimate the maximum value a filtered sample can assume.

In the following, we provide an analysis of the bit-length of the data involved in the computation by using a worst case analysis. The maximum value that each variable can assume may be easily determined by a logarithm computation.

Since the maximum value that a sample can assume is $max_y = 2^{\ell_y} - 1$, it follows that the maximum value that a filtered sample can assume is $max_x = |c_0 * max_y| + \sum_{j=1}^{k} |c_j * 2 * max_y| = max_y * (|c_0| + \sum_{j=1}^{k} |2 * c_j|)$, and its representation needs ℓ_x bits for the magnitude and 1 for the sign. The maximum value that a noise sample can assume is $max_x - (-amp max_y)$. The magnitude of each noise sample can be represented with ℓ_n bits and another bit is required for the sign.

After some experiments, we decided to subdivide the 30 seconds of signal ($t * f_s = 10800$ samples) into $m = 30$ segments, each having length one second ($w = 360$ samples). In the worst case, all filtered (or noise) samples assume the maximum value. The maximum value of the energy of a segment of the filtered signal is $max_{E_x} = \sum_{i=1}^{w} max_x^2 = w * max_x^2$ and the maximum energy of the noise segments is $max_{E_n} = w max_n^2$. For simplicity, we represent both max_{E_x} and max_{E_n} with the same number of bits ℓ_e, obtained by the logarithm of max_{E_n}. Since the energy is positive, it is not necessary to add a bit for the sign. Similarly, the maximum value that both the energies of the whole filtered signal and noise signal can attain is $mw(max_x + amp max_y)^2$, which can be represented with ℓ_E bits.

By using max_{E_n} as an upperbound for both the filtered and noise signal energies, the highest magnitude of the SNR is obtained when the energy of one of them is maximum and the other is zero (this scenario is not possible, in practice). Thus, the maximum value that the segment SNR can assume is $max_{SNR_f} = \lceil \log_2(max_{E_{fxi}}) \rceil = \pm \lceil \log_2(w * (2^{\ell_s} - 1)^2 * (|c_0| + \sum_{j=1}^{k} |2 * c_j|^2) \rceil$ and it can be represented with $\ell_{SNR_f} = \lceil \log_2 \ell_e \rceil$ bits for the magnitude and one bit for the sign. The SNR of the whole signal needs $\ell_{SNR} = \lceil \log_2 \ell_E \rceil + 1$ bits for its representation.

The maximum value that the mean of the SNR can assume is $m * max_{SNR_f}$ and needs ℓ_μ bits for the magnitude and one for the sign. The differences between the SNR values amplified by m and the mean require another bit.

Finally we can obtain the maximum value for the variance by considering that

$$\begin{aligned} \sigma^{SNR} &= \sum_{j=1}^{m} (mSNR_{f_j} - \mu^{SNR})^2 \\ &< \sum_{j=1}^{m} (mmax_{SNR_f} + mmax_{SNR_f})^2 \\ &= m^3 max_{SNR_f}^2 = max_\sigma \end{aligned}$$

and needs ℓ_σ bits for its representation. Being the variance positive, the sign bit is not needed.

We do not impose any limitation to the representation of the classification parameters because their values depend on the training set and different bit-lengths can be necessary for different people. The correct bit-length is hence chosen so that classification with quantized parameters is sufficiently similar to the classification involving real parameters, paying attention that the scalar product result does not exceed the maximum value allowed by the ciphertext. In some cases, shortest bit-lengths will mean significant lower computational complexity. The results of the above analysis are summarized in Table 11.1.

11.2.2 Communication Complexity

This section provides an analysis of the communication complexity of the protocol (summarized in Table 11.2). For our analysis, we assume the use of the following parameters (short term security): $T = 1024$, $t = 80$, $\kappa = 80$ bits.

Variable name	Maximum value	Magnitude bitlength
Original Signal s	1023	$\ell_s = 10$
Pre-filtered Signal y	1023	$\ell_y = 10$
Filter Coefficients c	8	$\ell_c = 4$
Filtered Signal x	114576	$\ell_x = 17$
Noise Signal n	180048	$\ell_n = 18$
Segment Energy $E_{fx},\ E_{fn}$	11670221629440	$\ell_e = 44$
Segment SNR SNR_f	44	$\ell_{SNR_f} = 6$
Full Energy $E_x,\ E_n$	350106648883200	$\ell_E = 49$
SNR SNR	49	$\ell_{SNR} = 6$
SNR Mean μ^{SNR}	1320	$\ell_\mu = 11$
SNR Variance σ^{SNR}	52272000	$\ell_\sigma = 26$

Table 11.1: Maximum value and number of bits necessary for the magnitude representation of the variables involved in the computation by worst case analysis. Another bit is needed for the sign, except for energies and SNR variance.

The protocol starts with the transmission of $\tau f_s = 10800$ cyphertexts having size $2T$ from $\mathcal{C}$ to $\mathcal{S}$. During energy computation, $\mathcal{S}$ transmits the filtered and noise samples to $\mathcal{C}$, together with the obfuscation values $r' - \sum_i 2r_i\hat{s}_i$, where $\hat{s}_i$ is used to indicate a generic x_i or n_i sample, that $\mathcal{C}$ has to remove from the energy. The $2\tau f_s$ samples can be packed in

$$\left\lceil \frac{2\tau f_s}{\lfloor T/(\ell_n + \kappa)\rfloor} \right\rceil$$

ciphertexts, while the $2m$ obfuscation values can be packed in

$$\left\lceil \frac{2m}{\lfloor T/(\max\{2\ell_n + \kappa + 1, \ell_e + \kappa\} + 1)\rfloor} \right\rceil$$

ciphertexts. At this point $\mathcal{C}$ needs to evaluate the GC. The circuit can be transmitted offline and is composed by m sub-circuits computing the segment SNR and one sub-circuit computing the whole SNR.

Since the energy values can be represented with ℓ_e bits, the GC that computes each SNR^f is composed by 2 subtraction circuits (ℓ_e non XOR gates), the circuit implementing the SNR computation ($2(\ell_e - 1) + \ell_e - \lceil \log_2(\ell_e + 1)\rceil + \ell_e = 4\ell_e - \lceil \log_2(\ell_e + 1)\rceil - 2$ non-XOR gates) and a controlled addition/subtraction circuit [BFL+11] that blinds the result ($\ell_{SNR_f} + \kappa$ non-XOR

Section	Offline	Online
HE	0	26,626,048
Circuit	3,402,240	0
$\mathcal{C}$ secrets to GC (OT)	914,560	438,080
$\mathcal{S}$ secrets to GC	432,320	0
Total	4,749,120	27,064,128

Table 11.2: Online and offline bandwidth (bits) required by the protocol

gates). Similarly the SNR of the whole filtered and noise signal energies (having bitlength ℓ_E) is computed and the circuit related is composed by $6\ell_E - \lceil \log_2(\ell_E + 1) \rceil - 2 + \ell_{SNR} + \kappa$ non-XOR gates. The whole circuit is hence composed by $m(6\ell_e - \lceil \log_2(\ell_e + 1) \rceil - 2 + \ell_{SNR_f} + \kappa) + 6\ell_E - \lceil \log_2(\ell_E + 1) \rceil - 2 + \ell_{SNR} + \kappa$ non-XOR gates having size $4t$ bits each. $\mathcal{C}$'s inputs to GC are $2m$ energies represented with ℓ_e bits each and 2 energies represented by ℓ_E bits. The secrets relative to $\mathcal{C}$'s input are transmitted by using OT. Since the number of input bits is greater than $3t$, the complexity of the OT is reduced to the transmission of $\approx 6t^2 + 4(2m\ell_e + 2\ell_E)t$ bits offline and $2(2m\ell_e + 2\ell_E)t$ bits online. $\mathcal{S}$'s inputs to the GC are the secrets corresponding to the random values used to blind the energies at the input and the SNR at the output. Notice that these values can be generated offline. Thus, they can also be transmitted offline together with the circuit, resulting in the transmission of $2m\ell_e + 2\ell_E + m(\ell_{SNR_f} + \kappa) + (\ell_{SNR} + \kappa)$ secrets of size t bits. $\mathcal{C}$ sends $\mathcal{S}$ m ciphertexts containing the obfuscated segment SNR, one ciphertext containing the SNR and, finally, two ciphertexts to compute the SNR variance.

11.2.3 Classification Performance

From each signal in the dataset, we obtained segments that can be subdivided into clean (c) or noisy (n). A segment is considered noisy if classified as such in the Physiobank databases. In a real implementation an expert decides if the signals used for training are clean or noisy.

Due to the short length of the signals in the database (30 minutes), only 60 segments can be extracted from each signals. After classifying them as noisy or clean according to database notations, signals having less than 10 clean segments or less than 10 noisy segments are discarded, because the small number of segments for a type can compromise a good training. We performed three different tests in which we trained and evaluated the classifiers on three different data sets. The first data set (c/n) was built by using only real clean and noisy sections. The second (c/a) data set was built by using clean sections and sections where artificial noise was added to whole clean sections (a), considering them as noisy sections. The last data set (c/p) is similar to the second one, but, instead of completely noisy sections, we used clean sections where the noise was added only to a randomly chosen interval (p).

For each signal, 60% of the segments of the different types were randomly chosen for the training and the others were used for the testing. A different $\boldsymbol{\beta}$ vector was obtained for each individual (signal). Table 11.3 shows the results obtained by using the linear classifier. The table also shows the performance that we can obtain by classifying the signal using only the mean of the segment SNR, the variance of the segment SNR or the whole SNR. The table contains the mean of the results of all the signals used for the tests. It is clear from these results that the combination of all derived values with a simple linear classifier significantly improves the classification results.

Table 11.3: Performance of the protocol using the linear classifier or a single feature.

Test type	Linear classifier	σ^{SNR}	μ^{SNR}	SNR
c/n	0.8490	0.8158	0.7061	0.7325
c/a	0.8365	0.8005	0.8368	0.8234
c/p	0.7377	0.6729	0.6695	0.6666

Therefore our protocol for remote ECG quality evaluation guarantees quite good classification results ($\approx$ 85%) with an online bandwidth of less than 4MBytes for a 30-seconds analysis.

Chapter 12

Conclusions

The need for privacy protection is steadily increasing in our society due to the wide diffusion of on-line distributed services offered by non-trusted parties having a potential access to private information like users' preferences or other personal data. This need is even more pressing in settings where the information to be protected is related to the health of the users: with the appearance of more and more online medical repositories, it is simple to imagine that in a few years the approach to remote healthcare will be completely different from the current one and it is of the outmost importance that manipulation of sensitive data does not compromise the privacy of users.

The availability of signal processing algorithms that work directly on the encrypted data is of great help for application scenarios where biomedical signals must be produced, processed or exchanged in digital format. In this thesis, we have broadly referred to this new class of signal processing techniques operating in the encrypted domain as s.p.e.d. (Signal Processing in the Encrypted Domain) techniques. While, in principle, the evaluation of any functionality in the encrypted domain is always possible, the development of efficient schemes that minimize the computational and communication complexity is not trivial, since it requires a joint design of the signal processing (SP) and cryptographic aspects of the system.

The first part of this thesis has been devoted to the presentation of the existing literature concerning the technologies related to s.p.e.d. . This results in a common background, that can help whoever is interested in this field. In particular the first part describes the necessary properties of the encryption primitives, to highlight the limits of current solutions, and to clarify the differences between signals and other kinds of data like alphanumeric or bit strings that impact on processing in the encrypted domain. The general cryptographic tools that allow to process encrypted signals are Homomorphich Encryption, Oblivious Transfer and Garbled Circuits, since they allow to perform computations on the encrypted data.

The definition of a GC-based proposal for the evaluation of the integer logarithm, having a low complexity counter circuit as main component, is very relevant. Moreover the definition of hybrid protocols permits to develop efficient protocols by dividing them in subparts each one implemented by using the most efficient HE or GC solution.

In the second part of the thesis we have shown how s.p.e.d. technology may help to achieve the twofold goal of allowing the processing of biomedical signals while ensuring the privacy of the signal's owner with reasonable communication and computation complexity. The proposed protocols address the classification of ElectroCardioGram (ECG) signals and rely on Secure Function Evaluation (SFE) constructions and on a proper design of the classification algorithms so to ease their implementation in a s.p.e.d. framework. More specifically, we have described two alternative ECG classification protocols, the former is based on a Linear Branching Program (LBP) classifier and is implemented by relying on a hybrid approach wherein homomorphic encryption and garbled circuit theories are used together, the latter implements a Neural Network (NN) classifier and relies only on garbled circuit constructions. The optimization of the signal processing part substantially improved the performance of both the solutions. Regarding the LBP solution, we experimentally compared two different implementations of the system, one relying on garbled circuits (GC) and one on a hybrid combination of the homomorphic Paillier cryptosystem and GCs. While from a communication complexity perspective the Hybrid protocol is clearly better, the computational complexity of the two protocols is similar for short-term security parameters, whereas for medium-term security the GC based protocol is preferable.

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

In the third part of the thesis, we presented a protocol to evaluate the quality of an ECG signal specifically geared towards remote health monitoring applications. Our solution is novel especially from a signal processing and privacy preserving points of view. In particular, our new technique analyzes the amount of noise in a biomedical signal based on analysis of the Signal-to-Noise ratio in small segments of the encrypted signal rather than the whole measurement [BGL10]. The analysis is based on the statistics of the raw signal segment and a corresponding filtered signal. The signal is divided into segments and the signal-to-noise ratio is computed between each segment and its denoised version. Given the SNRs, the first and second order moments (mean and variance) of them are obtained. These values are then used together with the SNR of the whole signal to classify it as clean or noisy depending on the result of a linear classifier, whose training set is specific to each individual. By using the integer logarithm, the SNR is computed by introducing a very small (accuracy) error.

For both the classification and quality evaluation protocols, a reader may wonder whether a running time in the order of a few seconds is affordable in real life applications. The ultimate answer to this question depends on the application at hand, however we can observe that a running time of less than one second would be enough for applications wherein heart beats are classified at the same pace at which they are produced. While our protocols are not that fast, their performance are not far away from the above so-to-say real time requirements thus witnessing the validity of our solutions.

Security issues have been overlooked in this thesis, focusing principally on signal processing aspects. Anyway security is a fundamental aspect of this topic and it is important to remember that the security of the proposed protocols is always guaranteed by the cryptographic primitives and by the additive obfuscation, so that each party is able to observe only its own input or output data, while intermediate values are kept secret and are not revealed to both the parties.

12.1 Tracks for future works

Preserving the privacy of the patients in a remote healthcare scenario is really important and, as demonstrated in this thesis, also feasible. We are working to propose new applications of s.p.e.d. to the health diagnosis and we hope that many other researchers decide to come abreast of us in this research.

The promising results of our research pave the way to a number of interesting research directions that can be exploited in the future.

First of all, new more efficient implementations of the underlying building blocks can be sought for to further improve the efficiency of the classifiers. In particular, the ECG quality evaluation protocol can still be improved by choosing the discrimination threshold according to a Relative Operating Characteristic (ROC) curve. It is important to note that a noisy signal classified as clean can result in wrong further classification. On the other side a clean signal classified as noisy causes only the request of a new placement of electrodes. Hence the two cases can be weighted differently in the analysis. Moreover in the protocol evaluating the quality we carried out only a worst case analysis, while tests aiming at choosing the lowest data dimension, still providing the same classification accuracy, could lead to better performance.

The results we obtained for the particular case of ECG classification could be extended to more general set-ups with the goal of deriving some general conclusions about the suitability of the LBP and the NN approaches to classification in a s.p.e.d. framework.

Different scenarios involving other biomedical signals, such as Electroencephalography (EEG), Electromyography (EMG), blood pressure, etc. can be considered. Moreover, similarly to the ECG case, a signal quality evaluation problem can be addressed for each of them, not necessarily relying on the protocol proposed here. Also the device calibration could be addressed in the encrypted domain.

Much work can still be done in this field. We are already working for a solution that allows us to evaluate a generic function in the encrypted domain through a secure implementation of a linear piece-wise approximation [PLB12]. This allows us to evaluate, for example, root and sinusoidal functions. Moreover an approximation of the logarithm can be used in the quality evaluation protocol. This guarantees higher precision than the current solution, to the detriment of the complexity. A difficult challenge consists in finding a solution that allows MPC protocols to work directly with floating point numbers. Moreover many new applications can still be exploited, such as smart metering, data fusion, game theory, etc. Also security models more stringent than the semi-honest model should be considered, attempting to develop efficient constructions in the presence of malicious adversaries, even if for security against stronger covert adversaries (who can be caught in a cheating attempt with a fixed probability) or malicious adversaries (who are caught cheating with overwhelming probability) the expected runtime grows considerably.

Last but not least, some new advances in cryptography, like the recently proposed fully homomorphic encryption (FHE) systems [Gen09, DGHV10, CNT, AGH10] could open new research directions, finally leading to a brand new class of efficient s.p.e.d. protocols. By using FHE, $\mathcal{S}$ can evaluate any functionality without the interaction with $\mathcal{C}$, that will perform only the decryption of the final result. In FHE protocols the values are bitwise encrypted and the functionalities represented by boolean circuits, similarly to GC solution. Unluckily efficient FHE schemes guaranteeing the same security level of GC has not been devised so far, making GC solution more used than FHE's.

12.2 Final considerations

Indeed in the last years research in Signal Processing in the Encrypted Domain produced many important results. At the beginning of our experience Yao's garbled circuits seemed to be only a theoretic result, not applicable in real applications. A fully homomorphic encryption scheme was supposed to be unrealistic. On the contrary efficient GC protocols and full homomorphic encryption are reality. Many important functionalities, such as logarithm, linear branching programs, neural networks, have found an implementation and Secure Multi-Party Computation has been applied in many different scenarios: biomedical, biometric, auctions, electronic voting, etc.

Actually privacy is protected only by weak laws, while the user inclination is to share any personal information through social networks and give full control on personal data to phone applications, such as the GPS position or personal preferences. Anyway we believe that, in a not so far future, people will rediscover the privacy importance and s.p.e.d. protocols will be the solution to major privacy concerns. In the meanwhile research keeps going on, funded by public agencies and some companies, to be ready when the privacy will be needed by the population.

Bibliography

[AB01] G. Arulampalam and A. Bouzerdoum. Application of shunting inhibitory artificial neural networks to medical diagnosis. In *Intelligent Information Systems Conference, The Seventh Australian and New Zealand 2001*, pages 89–94, 2001.

[ABF$^+$08] E. Aimeur, G. Brassard, J.M. Fernandez, F.S.M. Onana, and Z. Rakowski. Experimental demonstration of a hybrid privacy-preserving recommender system. In *Proceedings of the 2008 Third International Conference on Availability, Reliability and Security*, pages 161–170. IEEE Computer Society, 2008.

[ADAI$^+$09] M. Alfaouri, K. Daqrouq, I.N. Abu-Isbeih, E.F. Khalaf, A.R. Al-Qawasmi, and W. Al-Sawalmeh. Quality Evaluation of Reconstructed Biological Signals. *American Journal of Applied Sciences*, pages 187–193, 2009.

[AGH10] C. Aguilar, P. Gaborit, and J. Herranz. Additively homomorphic encryption with d-operand multiplications. In *Crypto 2010*, LNCS. Springer, 2010.

[AIR01] B. Aiello, Y. Ishai, and O. Reingold. Priced oblivious transfer: How to sell digital goods. *Advances in Cryptology – EUROCRYPT 2001*, pages 119–135, 2001.

[ALN87] N. Ahituv, Y. Lapid, and S. Neumann. Processing encrypted data. *Communications of the ACM*, 20(9):777–780, Sep. 1987.

[AS00] R. Agrawal and R. Srikant. Privacy-preserving data mining. *ACM Sigmod Record*, 29(2):439–450, 2000.

[ASSK07] U. R. Acharya, J. Suri, J. A. E. Spaan, and S. M. Krishnan. *Advances in Cardiac Signal Processing*. Springer, 2007.

[Bao03] F. Bao. Cryptanalysis of a provable secure additive and multiplicative privacy homomorphism. In *International Workshop on Coding and Cryptography (WCC)*, 2003.

[Bar04] W. C. Barker. Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher. NIST special publication 800-67, May 2004.

[BAS$^+$09] M. Barni, F. Armknecht, T. Schneider, A-R. Sadeghi, A. Piva, T. Bianchi, R. Lazzeretti, and P. Failla. Final report on s.p.e.d. theory, December 2009. Signal Processing in the EncryptEd Domain (SPEED): deliverable 2.5.

[BB04] D. Boneh and X. Boyen. Short signatures without random oracles. In *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 56–73. Springer-Verlag, 2004.

[BBC$^+$10a] M. Barni, T. Bianchi, D. Catalano, R. Di Raimondo, R. Donida Labati, P. Failla, D. Fiore, R. Lazzeretti, V. Piuri, A. Piva, and F. Scotti. A privacy-compliant fingerprint recognition system based on homomorphic encryption and fingercode templates. In *Biometrics: Theory, Applications and Systems, 2010. BTAS 2010. IEEE Fourth International Conference on*, 2010.

[BBC$^+$10b] M. Barni, T. Bianchi, D. Catalano, R. Di Raimondo, R. Donida Labati, P. Failla, D. Fiore, R. Lazzeretti, V. Piuri, A. Piva, and F. Scotti. Privacy-preserving fingercode authentication. In *Multimedia and Security, 2010. MM&Sec 2010. 12th ACM Workshop on*, 2010.

[BC08] J. Bringer and H. Chabanne. An authentication protocol with encrypted biometric data. *Progress in Cryptology–AFRICACRYPT 2008*, pages 109–124, 2008.

[BCA$^+$93] F. M. Bennett, D. J. Christini, H. Ahmed, K. Lutchen, J. M. Hausdorff, and N. Oriol. Time series modeling of heart rate dynamics. In *Computers in Cardiology 1993. Proceedings.*, pages 273–276, 1993.

[BCI$^+$07] J. Bringer, H. Chabanne, M. Izabachene, D. Pointcheval, Q. Tang, and S. Zimmer. An application of the Goldwasser-Micali cryptosystem to biometric authentication. In *The 12th Australasian Conference on Information Security and Privacy (ACISP 07)*, 2007.

[BDCOP04] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano. Public key encryption with keyword search. In *Advances in Cryptology-Eurocrypt 2004*, pages 506–522. Springer, 2004.

[Bea95] D. Beaver. Precomputing oblivious transfer. In *Advances in Cryptology – CRYPTO'95*, volume 963 of *LNCS*, pages 97–109. Springer, 1995.

[Ben88] J. D. C. Benaloh. *Verifiable Secret-Ballot Elections.* PhD thesis, Yale University, 1988.

[BFK$^+$09a] M. Barni, P. Failla, V. Kolesnikov, R. Lazzeretti, A.-R. Sadeghi, and T. Schneider. Secure evaluation of private linear branching programs with medical applications. In *14th European Symposium on Research in Computer Security (ESORICS'09)*, LNCS. Springer, 2009. Full version available at http://eprint.iacr.org/2009/195.

[BFK$^+$09b] M. Barni, P. Failla, V. Kolesnikov, R. Lazzeretti, A.R. Sadeghi, and T. Schneider. Combining signal processing and cryptographic protocol design for efficient ecg classification. In *Proceedings of the 1st International Workshop on Signal Processing in the EncryptEd Domain-SPEED 2009*, pages 42–61, 2009.

[BFL$^+$09] M. Barni, P. Failla, R. Lazzeretti, A. Paus, A.R. Sadeghi, T. Schneider, and V. Kolesnikov. Efficient privacy-preserving classification of ECG signals. In *Information Forensics and Security, 2009. WIFS 2009. First IEEE International Workshop on*, pages 91–95. IEEE, 2009.

[BFL$^+$11] M. Barni, P. Failla, R. Lazzeretti, A.R. Sadeghi, and T. Schneider. Privacy-preserving ECG classification with branching programs and neural networks. *Information Forensics and Security, IEEE Transactions on*, (99):1–1, 2011.

[BGL10] M. Barni, J. Guajardo, and R. Lazzeretti. Privacy preserving evaluation of signal quality with application to ECG analysis. In *Information Forensics and Security (WIFS), 2010 IEEE International Workshop on*, pages 1–6. IEEE, 2010.

[BGN05] D. Boneh, E.J. Goh, and K. Nissim. Evaluating 2-DNF formulas on ciphertexts. *Theory of Cryptography*, pages 325–341, 2005.

[BL96] D. Boneh and R. Lipton. Searching for elements in black box fields and applications. In *Proceedings of Crypto'96*, volume 1109 of *LNCS*, pages 283–297. Springer-Verlag, 1996.

[Bla08] D. Blankenhorn. Microsoft HealthVault is nothing like Google Health, 2008.

[BLS01] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the Weil pairing. In *Advances in Cryptology – ASIACRYPT*, volume 2248 of *Lecture Notes in Computer Science*, pages 514–532. Springer-Verlag, 2001.

[BM90] M. Bellare and S. Micali. Non-interactive oblivious transfer and applications. In *Advances in Cryptology - CRYPTO'89 Proceedings*, pages 547–557. Springer, 1990.

[BOGW88] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 1–10. ACM, 1988.

[Bon98] D. Boneh. The decision Diffie-Hellman problem. *Algorithmic Number Theory*, pages 48–63, 1998.

[BPB08a] T. Bianchi, A. Piva, and M. Barni. Comparison of different FFT implementations in the encrypted domain. In *Proc. 16 th European Signal Process. Conf*, 2008.

[BPB08b] 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.

[BPB08c] T. Bianchi, A. Piva, and M. Barni. Implementing the discrete Fourier transform in the encrypted domain. In *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*, pages 1757–1760. IEEE, 2008.

[BPB10] T. Bianchi, A. Piva, and M. Barni. Composite signal representation for fast and storage-efficient processing of encrypted signals. *Information Forensics and Security, IEEE Transactions on*, 5(1):180–187, 2010.

[BPSW07] J. Brickell, D. E. Porter, V. Shmatikov, and E. Witchel. Privacy-preserving remote diagnostics. In *ACM Conference on Computer and Communications Security (CCS'07)*, pages 498–507. ACM, 2007.

[BR93] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM CCS 1993: 1st Conference on Computer and Communications Security*, pages 62–73. ACM Press, 1993.

[BR94] M. Bellare and P. Rogaway. Optimal asymmetric encryption – how to encrypt with rsa. In *Euroccrypt'94*, volume 950 of *Lecture Notes in Computer Science*, pages 92–111, 1994.

[BR96] M. Bellare and P. Rogaway. The exact security of digital signatures: How to sign with RSA and Rabin. In *Advances in Cryptology – EUROCRYPT 1996*, volume 1070 of *Lecture Notes in Computer Science*, pages 399–416. Springer-Verlag, 1996.

[BY88] E.F. Brickell and Y. Yacobi. On privacy homomorphisms. In D. Chaum et al., editor, *Advances in Cryptology – Eurocrypt'87*, pages 117–125. Springer, 1988.

[Cam13] P. Campisi. *Security and Privacy in Biometrics*, chapter R. Lazzeretti and P. Failla and M. Barni. Privacy–Aware Processing of Biometric Templates by Means of Secure Two-Party Computation. Springer, 2013.

[CCD88] D. Chaum, C. Crépeau, and I. Damgård. Multiparty unconditionally secure protocols. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, page 19. ACM, 1988.

[CGH04] R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited. *Journal of the ACM (JACM)*, 51(4):557–594, 2004.

[CKN06a] J. Cheon, W. Kim, and H. Nam. Known-plaintext cryptanalysis of the Domingo-Ferrer algebraic privacy homomorphism scheme. *Inf. Process. Lett.*, 97(3):118–123, 2006.

[CKN06b] Jung Hee Cheon, Woo-Hwan Kim, and Hyun Soo Nam. Known-plaintext cryptanalysis of the domingo-ferrer algebraic privacy homomorphism scheme. *Information Processing Letters*, 97(3):118–123, Feb. 2006.

[CNT] J.S. Coron, D. Naccache, and M. Tibouchi. Public key compression and modulus switching for fully homomorphic encryption over the integers. *EuroCRYPT 2012*.

[Com11] European Commission. SHARE compact road map, October 2011. Available online at `http://googleblog.blogspot.com/2011/06/update-on-google-health-and-google.html`.

[Coo09] M. Cooney. IBM touts Encryption Innovation. Computerworld Newsletter, June 2009. `http://www.computerworld.com/s/article/9134823/IBM_touts_encryption_innovation?taxonomyId=152&intsrc=kc_top&taxonomyName=compliance`.

[CS99] R. Cramer and V. Shoup. Signature schemes based on the strong RSA assumption. In *ACM CCS 1999: 6th Conference on Computer and Communications Security*, pages 46–51. ACM Press, 1999.

[CT06] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory.* Wiley-Interscience, 2006.

[DCDZ05] S. J. Delany, P. Cunningham, D. Doyle, and A. Zamolotskikh. Generating estimates of classification confidence for a case-based spam filter. In *Case-Based Reasoning Research and Development*, volume 3620 of *LNCS*, pages 177–190. Springer, 2005.

[DF96] J. Domingo-Ferrer. Privacy homomorphisms for statistical confidentiality. *Qüestiió*, 20(3):505–521, Dec. 1996.

[DF02] J. Domingo-Ferrer. A provably secure additive and multiplicative privacy homomorphism. In *Proceedings of the 5th International Conference on Information Security (ISC 2002)*, number 2433 in LNCS, pages 471–483. Springer-Verlag, 2002.

[DGHV10] M. v. Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan. Fully homomorphic encryption over the integers. In *Advances in Cryptology – EUROCRYPT'10*, LNCS. Springer, 2010.

[DGK07] I. Damgård, M. Geisler, and M. Krøigaard. Efficient and secure comparison for on-line auctions. In *Information Security and Privacy*, pages 416–430. Springer, 2007.

[DGK09] I. Damgård, M. Geisler, and M. Kroigard. A correction to 'Efficient and secure comparison for on-line auctions'. *International Journal of Applied Cryptography*, 1(4):323–324, 2009.

[DH76] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on information Theory*, 22(6):644–654, 1976.

[DJ01] I. Damgård and M. Jurik. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In *Public Key Cryptography*, pages 119–136, 2001.

[DR99] J. Daemen and V. Rijmen. The Rijndael block cipher. AES Proposal, Mar. 1999.

[DSA00] Digital signature standard. National Institute of Standards and Technology, NIST FIPS PUB 186-2, U.S. Department of Commerce, 2000.

[dVMPT08] F. Levy dit Vehel, M. Marinari, L. Perret, and C. Traverso. *Gröbner Bases, Coding Theory, and Cryptography*, chapter A Survey on Polly Cracker Systems. RISC Book Series. Springer, Heidelberg, 2008.

[EBVL11] Z. Erkin, M. Beye, T. Veugeri, and R.L. Lagendijk. Efficiently computing private recommendations. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 5864–5867. IEEE, 2011.

[EFG$^+$09] 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, 2009.

[EGL85] S. Even, O. Goldreich, and A. Lempel. A randomized protocol for signing contracts. *Communications of the ACM*, 28(6):647, 1985.

[ElG85] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, IT-31(4):469–472, 1985.

[EPK$^+$07] Z. Erkin, A. Piva, S. Katzenbeisser, RL Lagendijk, J. Shokrollahi, G. Neven, and M. Barni. Protection and retrieval of encrypted multimedia content: when cryptography meets signal processing. *EURASIP Journal on Information Security*, 2007:17, 2007.

[EVL11] Z. Erkin, T. Veugen, and RL Lagendijk. Generating private recommendations in a social trust network. In *Computational Aspects of Social Networks (CASoN), 2011 International Conference on*, pages 82–87. IEEE, 2011.

[EVTL12] Z. Erkin, T. Veugen, T. Toft, and R.L. Lagendijk. Generating private recommendations efficiently using homomorphic encryption and data packing. *Information Forensics and Security, IEEE Transactions on*, 7(3):1053–1066, 2012.

[Fai10] P. Failla. Heuristic search in encrypted graphs. In *IARIA International Conference on Emerging Security Information, Systems and Technologies, SECURWARE*, volume 2010, 2010.

[FDH$^+$10] M. Franz, B. Deiseroth, K. Hamacher, S. Jha, S. Katzenbeisser, and H. Schroeder. Secure Computations on Non-Integer Values. *Information Forensics and Security, 2010. WIFS 2010. Second IEEE International Workshop on*, 2010.

[Fer96] J.D. Ferrer. A new privacy homomorphism and applications. *Information Processing Letters*, 60(5):277–282, 1996.

[FG07] C. Fontaine and F. Galand. A survey of homomorphic encryption for nonspecialists. *EURASIP J. Inf. Secur.*, 2007(1):1–15, 2007.

[FJJ+90] G.M. Friesen, T.C. Jannett, M.A. Jadallah, S.L. Yates, S.R. Quint, and H.T. Nagle. A comparison of the noise sensitivity of nine QRS detection algorithms. *Biomedical Engineering, IEEE Transactions on*, 37(1):85–98, 1990.

[FK] M. Fellows and N. Koblitz. Combinatorial cryptosystems galore! *Finite Fields: Theory, Applications, and Algorithms*, 168:51–61.

[FLK08] P. Flor-Henry, J. L. Lind, and Z. J. Koles. Quantitative EEG and source localization in fibromyalgia. *International Journal of Psychophysiology*, 69(3):142–142, 2008.

[FOP+01] E. Fujisaki, T. Okamoto, D. Pointcheval, , and J. Stern. RSA–OAEP is secure under the RSA assumption. In *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 260–275, 2001.

[FSW02] P.A. Fouque, J. Stern, and G.J. Wackers. Cryptocomputing with rationals. In *Proceedings of the 6th international conference on Financial cryptography*, pages 136–146. Springer-Verlag, 2002.

[GAG+00] A. L. Goldberger, L. A. N. Amaral, L. Glass, J. M. Hausdorff, P. Ch. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley. Physiobank, Physiotoolkit, and Physionet: Components of a new research resource for complex physiologic signals. *Circulation*, 101(23):e215–e220, June 2000.

[GAM85] RA GAMES. Complex approximations using algebraic integers. *IEEE Transactions on Information Theory*, 1985.

[GB11] The Official Google Blog. An update on google health and google power meter, 2011. Available online at `http://roadmap.healthGrid.org`.

[Gen09] C. Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the 41st annual ACM symposium on Theory of Computing*, pages 169–178. ACM, 2009.

[GHR99] R. Gennaro, S. Halevi, and T. Rabin. Secure hash-and-sign signatures without the random oracle. In *Advances in Cryptology – EUROCRYPT*

1999, volume 1592 of *Lecture Notes in Computer Science*, pages 123–139. Springer-Verlag, 1999.

[GHX07] D. F. Ge, B. P. Hou, and X. J. Xiang. Study of feature extraction based on autoregressive modeling in ECG automatic diagnosis. *Acta Automatica Sinica*, 33(5):462–466, 2007.

[GM84] S. Goldwasser and S. Micali. Probabilistic encryption. *J. COMP. SYST. SCI.*, 28(2):270–299, 1984.

[GMR88] S. Goldwasser, S. Micali, and R. L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, April 1988.

[GMW91] O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the ACM (JACM)*, 38(3):690–728, 1991.

[GQ09] D. Giry and J.-J. Quisquater. Cryptographic key length recommendation, March 2009. `http://keylength.com`.

[GSK02] D. F. Ge, N. Srinivasan, and S. M. Krishnan. Cardiac arrhythmia classification using autoregressive modeling. *BioMedical Engineering OnLine*, 1(1):5, 2002.

[Hel78] M. Hellman. An overview of public key cryptography. *IEEE Communications Magazine*, 16(6):24–32, 1978.

[HKL13] F. Hartung, T. Kalker, and S. Lian. *Digital Rights Management: Technology, Standards and Applications*, chapter M. Barni and R. Lazzeretti and C. Orlandi. Processing encrypted signals for DRM applications. CRC Press, 2013.

[HM94] M.T. Hagan and M.B. Menhaj. Training feedforward networks with the Marquardt algorithm. *IEEE transactions on Neural Networks*, 5(6):989–993, 1994.

[HP98] J.H.L. Hansen and B.L. Pellom. An effective quality evaluation protocol for speech enhancement algorithms. In *Fifth International Conference on Spoken Language Processing*. Citeseer, 1998.

[HRD$^+$07] J. Ha, C. J. Rossbach, J. V. Davis, I. Roy, H. E. Ramadan, D. E. Porter, D. L. Chen, and E. Witchel. Improved error reporting for software that uses black-box components. In *Programming Language Des. and Impl. (PLDI'07)*. ACM, 2007.

[IKNP03] Y. Ishai, J. Kilian, K. Nissim, and E. Petrank. Extending oblivious transfers efficiently. In *Advances in Cryptology – CRYPTO'03*, volume 2729 of *LNCS*. Springer, 2003.

[IR89] R. Impagliazzo and S. Rudich. Limits on the provable consequences of one-way permutations. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 44–61. ACM, 1989.

[KB95] MW Kattan and R. Beck. Artificial neural networks for medical classification decisions. *Archives of pathology & laboratory medicine(1976)*, 119(8):672–677, 1995.

[KJGB06] L. Kruger, S. Jha, E.-J. Goh, and D. Boneh. Secure function evaluation with ordered binary decision diagrams. In *ACM Conference on Computer and Communications Security (CCS'06)*, pages 410–420. ACM Press, 2006.

[KLC⁺08] S. Katzenbeisser, A. Lemma, M.U. Celik, M. van der Veen, and M. Maas. A buyer–seller watermarking protocol based on secure embedding. *Information Forensics and Security, IEEE Transactions on*, 3(4):783–786, 2008.

[KS08a] V. Kolesnikov and T. Schneider. Improved garbled circuit: Free XOR gates and applications. In *International Colloquium on Automata, Languages and Programming (ICALP'08)*, volume 5126 of *LNCS*, pages 486–498. Springer, 2008.

[KS08b] V. Kolesnikov and T. Schneider. A practical universal circuit construction and secure evaluation of private functions. In *Financial Cryptography and Data Security (FC'08)*, volume 5143 of *LNCS*, pages 83–97. Springer, 2008.

[KSS09a] V. Kolesnikov, A.R. Sadeghi, and T. Schneider. How to combine homomorphic encryption and garbled circuits. In *Signal Processing in the Encrypted Domain–First SPEED Workshop–Lousanne*, page 100, 2009.

[KSS09b] V. Kolesnikov, A.R. Sadeghi, and T. Schneider. Improved garbled circuit building blocks and applications to auctions and computing minima. *Cryptology and Network Security*, pages 1–20, 2009.

[KSS10] Vladimir Kolesnikov, Ahmad-Reza Sadeghi, and Thomas Schneider. From dust to dawn: Practically efficient two-party secure function evaluation protocols and their modular design. Cryptology ePrint Archive, Report 2010/079, 2010. http://eprint.iacr.org/.

[KTB+03] P. Kalra, J. Togami, G. Bansal, A. W. Partin, M. K. Brawer, R. J. Babaian, L. S. Ross, and C. S. Niederberger. A neurocomputational model for prostate carcinoma detection. *Cancer*, 98(9):1849–1854, 2003.

[LB08] R. Lazzeretti and M. Barni. Lossless compression of encrypted grey-level and color images. In *Proceeding of 16th European Signal Processing Conference (EUSIPCO 2008)*, 2008.

[LB11] R. Lazzeretti and M. Barni. Division between encrypted integers by means of garbled circuits. In *Information Forensics and Security (WIFS), 2011 IEEE International Workshop on*, pages 1–6. IEEE, 2011.

[LF12] R. Lazzeretti and P. Failla. Data Sorting in Encrypted Domain. In *Information Forensics and Security (WIFS), 2012 IEEE International Workshop on*. IEEE, 2012. Submitted.

[LGB12] R. Lazzeretti, J. Guajardo, and M. Barni. Privacy preserving ECG quality evaluation. In *Multimedia and Security, 2012. MM&Sec 2012. 14th ACM Workshop on*, 2012.

[LMC08] Q. Li, RG Mark, and GD Clifford. Robust heart rate estimation from multiple asynchronous noisy sources. *Physiological measurement*, 29:15–32, 2008.

[LP08] Y. Lindell and B. Pinkas. Privacy preserving data mining. *Journal of cryptology*, 15(3):177–206, 2008.

[LP09] Y. Lindell and B. Pinkas. A proof of Yao's protocol for secure two-party computation. *Journal of Cryptology*, 22(2):161–188, 2009. Cryptology ePrint Archive: Report 2004/175.

[LPS08] Y. Lindell, B. Pinkas, and N. Smart. Implementing two-party computation efficiently with security against malicious adversaries. In *Security and Cryptography for Networks (SCN'08)*, volume 5229 of *LNCS*, pages 2–20. Springer, 2008.

[LSP+12] Y. Luo, S.C. Samson, T. Pignata, R. Lazzeretti, and M. Barni. An efficient protocol for private iris-code matching by means of garbled circuits. In *submitted to Special Session on Emerging Topics in Cryptography and Image Processing, International Conference on Image Processing (ICIP)*, 2012.

[McB08] M. McBride. Google Health: Birth of a giant. *Health Management Technology*, 29:8–10, 2008.

[MGH08] C. Melchor, P. Gaborit, and J. Herranz. Additive homomorphic encryption with t-operand multiplications. Cryptology ePrint Archive, Report 2008/378, 2008. `http://eprint.iacr.org/`.

[MNPS04] D. Malkhi, N. Nisan, B. Pinkas, and Y. Sella. Fairplay — a secure two-party computation system. In *USENIX*, 2004. `http://fairplayproject.net`.

[NP01] M. Naor and B. Pinkas. Efficient oblivious transfer protocols. In *ACM-SIAM Symposium On Discrete Algorithms (SODA'01)*, pages 448–457. Society for Industrial and Applied Mathematics, 2001.

[NS97] D. Naccache and J. Stern. A new public-key cryptosystem. In *Eurocrypt 97*, pages 27–36, 1997.

[OPB07] C. Orlandi, A. Piva, and M. Barni. Oblivious neural network computing from homomorphic encryption. *EURASIP Journal on Information Security*, 2007:11, 2007.

[OS07] R. Ostrovsky and W.E. Skeith. Private searching on streaming data. *Journal of Cryptology*, 20(4):397–430, 2007.

[OU98] T. Okamoto and S. Uchiyama. A new public-key cryptosystem as secure as factoring. In *Eurocrypt 98*, pages 308–318, 1998.

[Pai99] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in CryptologyEUROCRYPT99*, pages 223–238. Springer, 1999.

[PLB12] T. Pignata, R. Lazzeretti, and M. Barni. General Function Evaluation in a STPC Setting via Piecewise Linear Approximation. In *Information Forensics and Security (WIFS), 2012 IEEE International Workshop on*. IEEE, 2012. Submitted.

[PS00] D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, 2000.

[PSS09] A. Paus, A.-R. Sadeghi, and T. Schneider. Practical secure evaluation of semi-private functions. In *Applied Cryptography and Network Security (ACNS'09)*, volume 5536 of *LNCS*, pages 89–106. Springer, 2009. `http://www.trust.rub.de/FairplaySPF`.

[PSSW09] B. Pinkas, T. Schneider, N. P. Smart, and S. C. Williams. Secure two-party computation is practical. In *Advances in Cryptology – ASIACRYPT 2009*, volume 5912 of *LNCS*. Springer, December 6-10, 2009. Full version available at `http://eprint.iacr.org/2009/314`.

[PT85] J. Pan and W. J. Tompkins. A real-time QRS detection algorithm. *Biomedical Engineering, IEEE Transactions on*, pages 230–236, 1985.

[Rab81] M. Rabin. How to exchange secrets by oblivious transfer. Technical report, Technical Report TR-81, Harvard Aiken Computation Laboratory, 1981, 1981.

[Rab05] K. Rabah. Elliptic curve ElGamal encryption and signature schemes. *Information Technology Journal*, 4(3):299–306, 2005.

[RAD78a] R. L. Rivest, L. Adleman, and M. L. Dertouzos. On data banks and privacy homomorphisms. In R.A. DeMillo et al., editor, *Foundations of Secure Computation*, pages 169–179. Academic Press, 1978.

[RAD78b] R.L. Rivest, L. Adleman, and M.L. Dertouzos. On data banks and privacy homomorphisms. *Foundations of secure computation*, 32(4):169–178, 1978.

[Rap04] D. Rappe. *Homomorphic cryptosystems and their applications.* PhD thesis, University of Dortmund, Germany, 2004.

[RBO89] T. Rabin and M. Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 73–85. ACM, 1989.

[RGI05] J. Rodriguez, A. Goni, and A. Illarramendi. Real-time classification of ECGs on a PDA. *IEEE Transact. on Inform. Technology in Biomedicine*, 9(1):23–34, 2005.

[Riv92] R. Rivest. RFC1321: The MD5 message-digest algorithm. *RFC Editor United States*, 1992.

[Rob95] M.J.B. Robshaw. MD2, MD4, MD5, SHA and other hash functions. Technical report, Technical Report TR-101, RSA Laboratories, 1995. version 4.0, 1995.

[RS10] S. Rane and W. Sun. Privacy preserving string comparisons based on Levenshtein distance. In *Information Forensics and Security (WIFS), 2010 IEEE International Workshop on*, pages 1–6. IEEE, 2010.

[RSA78] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.

[Sch90] C.-P. Schnorr. Efficient identification and signatures for smart cards. In *Advances in Cryptology – CRYPTO 1989*, volume 435 of *Lecture Notes in Computer Science*, pages 239–252. Springer-Verlag, 1990.

[Sch08] T. Schneider. Practical secure function evaluation. Master's thesis, University of Erlangen-Nuremberg, February 27, 2008.

[Sch09] B. Schneier. Homomorphic Encryption Breakthrough. Blog: Schneier on Security, July 2009. `http://www.schneier.com/blog/archives/2009/07/homomorphic_enc.html`.

[Sha49] C. E. Shannon. Communication theory of secrecy systems. *Bell. Syst. Tech. J.*, 28:656–715, 1949.

[Sho97] V. Shoup. Lower bounds for discrete logarithms and related problems. In *Proceedings of the 16th annual international conference on Theory and application of cryptographic techniques*, pages 256–266. Springer-Verlag, 1997.

[SS08] A.-R. Sadeghi and T. Schneider. Generalized Universal Circuits for Secure Evaluation of Private Functions with Application to Data Classification. In *International Conference on Information Security and Cryptology (ICISC'08)*, volume 5461 of *LNCS*, pages 336–353. Springer, 2008.

[SSW09] A.R. Sadeghi, T. Schneider, and I. Wehrenberg. Efficient privacy-preserving face recognition. In *Information Security and Cryptology – ICISC 2009, International Conference on*. Springer, 2009.

[ST06a] B. Schoenmakers and P. Tuyls. Efficient binary conversion for Paillier encrypted values. *Advances in Cryptology-EUROCRYPT 2006*, pages 522–537, 2006.

[ST06b] B. Schoenmakers and P. Tuyls. Efficient binary conversion for Paillier encrypted values. In *Advances in Cryptology – EUROCRYPT'06*, volume 4004 of *LNCS*, pages 522–537. Springer, 2006.

[SYY99] T. Sander, A. Young, and M. Yung. Non-interactive cryptocomputing for NC^1. In *IEEE Symp. on Found. of Comp. Science (FOCS'99)*, pages 554–566. IEEE, 1999.

[TPKCL07] J.R. Troncoso-Pastoriza, S. Katzenbeisser, M. Celik, and A. Lemma. A secure multidimensional point inclusion protocol. In *Proceedings of the 9th workshop on Multimedia & security*, pages 109–120. ACM, 2007.

[TWBT95] S. Todd, M. Woodward, C. Bolton-Smith, and H. Tunstall-Pedoe. An investigation of the relationship between antioxidant vitamin intake and coronary heart disease in men and women using discriminant analysis. *Journal of Clinical Epidemiology*, 48(2):297–305, 1995.

[VAS85] J. A. Van Alste and T. S. Schilder. Removal of base-line wander and power-line interference from the ECG by an efficient FIR filter with a reduced number of taps. *Biomedical Engineering, IEEE Transactions on*, pages 1052–1060, 1985.

[Veu10] T. Veugen. Encrypted integer division. In *Information Forensics and Security (WIFS), 2010 IEEE International Workshop on*, pages 1–6. IEEE, 2010.

[VL06] L. Van Ly. Polly Two: A new algebraic polynomial-based public-key scheme. *Applicable Algebra in Engineering, Communication and Computing*, 17(3):267–283, 2006.

[Wag03] D. Wagner. Cryptanalysis of an algebraic privacy homomorphism. In *Proceedings of the 6th International Conference on Information Security (ISC 2003)*, number 1109 in LNCS, pages 234–239. Springer-Verlag, 2003.

[Yao82] A. C. Yao. Protocols for secure computations. In *IEEE Symposium on Foundations of Computer Science (FOCS'82)*, pages 160–164. IEEE, 1982.

[Yao86] A. C. Yao. How to generate and exchange secrets. In *IEEE Symposium on Foundations of Computer Science (FOCS'86)*, pages 162–167. IEEE, 1986.

[ZJGX04] Z. G. Zhang, H. Z. Jiang, D. F. Ge, and X. J. Xiang. Pattern recognition of cardiac arrhythmias using scalar autoregressive modeling. In *World Concress on Intelligent Control and Automation (WCICA'04)*, volume 6, 2004.

Curriculum Vitae

Riccardo Lazzeretti was born in Siena, Italy on October 26, 1979. He qualified from Industrial Technical Institute I.T.I.S. Tito Sarrocchi, Siena, in 1998 with vote 60/60 in Informatics.

He graduated cum laude in Informatics Engineering in April 2007, at the University of Siena (Information Engineering Department) with a thesis on "Lossless Compression of Encrypted Grey-Level and Color Images". In Since October 2007, he has been a Ph.D. student at the Department of Information Engineering of the University of Siena under the guidance of Prof. Mauro Barni.

His research interests focused on Signal Processing in the Encrypted Domain (s.p.e.d.). During his Ph.D. course he participated actively to the European FP6 SPEED project (`www.speedproject.eu`) and to the italian Priv-Ware project (`clem.dii.unisi.it/~vipp/projects/prin/`), funded by Miur (contract n. 2007JXH7ET). During his study he visited for six months the Philips Lab. Europe, in Eindhoven, Netherland, where he worked on quality evaluation of private signals, under the guidance of the Senior Scientist Jorge Guajardo.

Currently he has a research grant at the Department of Information Engineering of the University of Siena still working in the s.p.e.d. field, particularly in biometric and biomedical applications. Moreover he is participating to the project "Development of encrypted-domain data fusion protocols for Electronic Warfare applications", sponsored by Elt Elettronica, Roma, Italy.

To preserve the privacy of patients and service providers in biomedical signal processing applications, particular attention has been given to the use of secure multiparty computation techniques. This thesis focuses on the development of a privacy preserving automatic diagnosis system whereby a remote server classifies a biomedical signal provided by the patient without getting any information about the signal itself and the final result of the classification. Specifically, we present and compare two methods for the secure classification of electrocardiogram (ECG) signals: the former based on linear branching programs and the latter relying on neural networks. Moreover a protocol that performs a preliminary evaluation of the signal quality is proposed. The thesis deals with all the requirements and difficulties related to working with data that must stay encrypted during all the computation steps. The proposed systems prove that carrying out efficiently complex tasks, like ECG classification and quality evaluation, in the encrypted domain is indeed possible in the semihonest model, paving the way to interesting future applications wherein privacy of signal owners is protected by applying high security standards.

UNIVERSITÀ
DI SIENA
1240

The Ph.D. School of Information Engineering of the University of Siena is a school aiming at educating scholars in a number of fields of research in the Information Engineering area. The Ph.D. School of Information Engineering is part of the Santa Chiara High School of the University of Siena. A Scientific Committee of external experts recognized Ph.D. Schools belonging to Santa Chiara as excellent, according to their degree of internationalization, their research, and educational activities.