
**AIN SHAMS UNIVERSITY**
**FACULTY OF ENGINEERING**
**Computer and Systems Engineering**

# Homomorphic Encryption for Secure Data Computations

A Thesis submitted in partial fulfillment of the requirements of
Master of Science in Electrical Engineering
(Computer and Systems Engineering)

by

**Mohamed Tarek Ibn Ziad Mohamed Hassan**

Bachelor of Science in Electrical Engineering
(Computer and Systems Engineering)
Faculty of Engineering, Ain Shams University, 2014

Supervised By

**Dr. Hassan Mohamed Shehata Bedour**

**Dr. Yousra Mohsen Ali Alkabani**

Cairo, 2017


**AIN SHAMS UNIVERSITY**
**FACULTY OF ENGINEERING**
**Computer and Systems Engineering**

**Examiners' Committee**

Name: **Mohamed Tarek Ibn Ziad Mohamed Hassan**
Thesis: **Homomorphic Encryption for Secure Data Computations**
Degree: **Master of Science in Electrical Engineering (Computer and Systems Engineering)**

| **Name and affiliation** | **Signature** |
|---|---|
| **Prof. Dr. Hossam Aly Hassan Fahmy**<br>Prof. at Electronics and Communications Engineering Dept.<br>Faculty of Engineering, Cairo University. | . . . . . |
| **Prof. Dr. Mohamed Watheq Ali Kamel El-Kharashi**<br>Prof. at Computer and Systems Engineering Dept.<br>Faculty of Engineering, Ain shams University. | . . . . . |
| **Dr. Hassan Mohamed Shehata Bedour**<br>Associative Prof. at Computer and Systems Engineering Dept.<br>Faculty of Engineering, Ain shams University. | . . . . . |

Date: June 2017

# Statement

This Thesis is submitted as a partial fulfillment of Master of Science in Electrical Engineering, Faculty of Engineering, Ain shams University. The author carried out the work included in this Thesis, and no part of it has been submitted for a degree or a qualification at any other scientific entity.

**Mohamed Tarek Ibn Ziad Mohamed Hassan**

Signature

.....

**Date:** 01 June 2017

# Researcher Data

**Name:** Mohamed Tarek Ibn Ziad Mohamed Hassan
**Date of Birth:** 25/06/1992
**Place of Birth:** Cairo, Egypt
**Last academic degree:** Bachelor of Science
**Field of specialization:** Electrical Engineering
**University issued the degree :** Ain Shams University
**Date of issued degree :** 07/2014
**Current job :** Teaching Assistant at the Faculty of Engineering, Ain Shams University

# Abstract

**T**he tremendously increasing amount of the available data nowadays opens the door to using third parties to handle data storage and processing. This raises many concerns regarding end-users' privacy and whether the targeted third parties are trusted or not. On the one hand, end-users, either clients or organizations, cannot afford the cost and complexity of processing their own data by their local trusted components. On the other hand, depending only on third parties, such as cloud computing services, with no security guarantee in mind, will be more like building castles out of mud. One possible solution for the former issue is using homomorphic encryption (HE) techniques. These techniques allow third party services to compute over data while the data itself remains encrypted. Thus, one can make use of the great computational power offered by third parties without sacrificing his/her own privacy.

HE could be categorized into two main categories; partially homomorphic encryption (PHE), and fully homomorphic encryption (FHE). While FHE can help solve privacy issues completely, it introduces high performance overhead. To avoid such overhead, PHE can be used. Thus, the main goal of this Thesis is to *"explore the efficiency of using PHE techniques in solving real-world problems, in which computing over encrypted data is a must"*.

The contributions of this Thesis are multi-fold. We selected three different domains of applications; securing electronic voting (e-voting) systems, defeating Hardware Trojans (HTs) in FPGA-based designs, and operating blindly over encrypted images. The common part of all the above different domains is the availability of secure data that needs to be processed by third parties without being revealed.

In the context of *securing e-voting systems*, we implement an FPGA-based e-voting system, which uses a VGA screen and a Xilinx Spartan 3E FPGA board as a voting site and a remote server to collect results. We launch a couple of attacks on the system by injecting an HT in our e-voting machine to tamper with the voting results. We show the role of HE in securing our design via the usage of ElGamal cryptosystem. Protection techniques are proposed and implemented. Then, they are evaluated by showing their delay, power, and area overheads. The reported power overhead is negligible, the delay overhead does not exceed 10%, and the device resources overhead does not exceed 4%.

In the context of *defeating HTs in FPGA-based designs*, we implement two designs that support PHE (multiplicative only and additive only) based on ElGamal encryption/decryption scheme. Furthermore, we integrate the two designs together and introduce a dual-circuit design that achieves a higher improvement in area and power than a regular design that combines the two original separated designs. Our architectures are implemented on a Spartan-6 FPGA board from Xilinx. The area reduction reached 30% and savings in power consumption were 20% for encryption and 12% for decryption.

In the context of *operating blindly over encrypted images*, we introduce ***CryptoImg***, a library of modular privacy preserving image processing operations over encrypted images using the homomorphic properties of Paillier cryptosystem. Secure operations, such as image adjustment, spatial filtering, edge sharpening, edge detection, morphological operations, and histogram equalization, are safely outsourced to third-party servers with no privacy issues. We present how these operations can be implemented with much less time overhead, and a single communication round. ***CryptoImg*** can be used from either mobile or desktop clients with low client-side overheads. Experiments show the efficiency of our proposed library. For instance, the image negation operation in the encrypted domain requires less than one minute with zero error using 1024-bit key size.

To conclude, the Thesis successfully managed to show the efficiency of using PHE techniques, such as ElGamal and Paillier, as a replacement of FHE ones in three different real-world problems, which require computing over encrypted data. The overheads accompanied by using such techniques are reasonable compared to the huge overheads of the FHE techniques reported in the literature.

# Summary

Encryption is the art of converting text messages into a secret code using a certain encryption key. Hundreds of years ago, people used to propose new encryption algorithms to secure their own data and protect their privacy. Unfortunately, once the data is encrypted, it would remain in its secret-useless form till a certain key is used to decrypt it. Homomorphic encryption is the kind of encryption that permits one to perform useful computations on encrypted data without decrypting them. The results of these computations are equivalent to the results of the similar computations done over the plain data. By this way, one could safely allow third parties to make useful computations over his own data without scarifying his privacy.

Building on the above description, the Thesis aims at exploring the efficiency of using homomorphic encryption techniques in solving real-world problems, in which computing over encrypted data is a must. Although homomorphic encryption includes two different categories; the fully homomorphic encryption techniques, and the partially homomorphic encryption ones, the Thesis focuses on the second category only. The reason behind this research direction is to avoid the high overheads associated with the usage of fully homomorphic encryption methods.

The Thesis is divided into six chapters, along with the table of contents, list of figures, list of tables, abbreviations, symbols, and the references.

The Thesis contents are presented hereafter.

Chapter 1 presents the introduction to the Thesis. It mainly highlights the motivation behind the proposed work. It also states the Thesis contributions, which span through three different applications. Each contribution/application is illustrated in a separate chapter with its experimental setup and numerical results.

Chapter 2 describes the needed background about homomorphism and surveys existing fully and partially homomorphic encryption schemes. It mainly focuses on the two partially homomorphic cryptosystems, used in this work; ElGamal and Paillier.

Chapter 3 illustrates the first Thesis contribution, which is using homomorphism in E-voting systems. It introduces a couple of possible attacks and countermeasures to an FPGA-based voting machine. The full hardware implementation is described and the countermeasures overheads are highlighted.

Chapter 4 introduces the second contribution, which is securing FPGA-based designs from Hardware Trojans using homomorphism. The Thesis implements two partially homomorphic encryption designs based on ElGamal encryption/decryption scheme. The first design is a multiplicative homomorphic, whereas the second one is an additive homomorphic. The design realization on a low-cost FPGA is described and the area/timing results are reported.

Chapter 5 describes the third contribution, which is a cloud-based library, *CryptoImg*, which allows performing image processing operations over encrypted images. New algorithms are introduced and the communication/computation overhead are stated using various test cases.

Chapter 6 concludes the work, states the list of contributions, and discusses some possible future work directions.

**Keywords**

Electronic voting, ElGamal Encryption, Fully homomorphic encryption, Hardware Trojans, Homomorphism, Image Processing, Paillier Encryption, Partially homomorphic encryption, Secure computations.

# Acknowledgment

All praise is due to Allah, Most Merciful, the Lord of the Worlds, Who taught man what he knew out. I would like to thank God almighty for bestowing upon me the chance, strength and ability to complete this work.

I always struggle writing acknowledgments because words cannot do justice to my gratitude. However, I will do my best. I wish to express my deep gratitude to my advisors, Dr. Hassan Mohamed Shehata Bedour and Dr. Yousra Mohsen Ali Alkabani for their guidance and important remarks on the developed results and the written manuscript. I was greatly inspired by their excitement, vision, and creativity. I would also like to thank Prof. Mohamed Watheq Ali Kamel El-Kharashi for his constant and enthusiastic support.

I would also like to thank the following researchers from University of California Los Angeles (UCLA ), Amr Alanwar, Moustafa Alzantot, and Mani Srivastava, for their help on the development and review of ***CryptoImg***, discussed in Chapter 5.

I am always thankful to my parents for their unfailing help along my life and for their efforts during the Thesis development. I am also thankful to my wonderful friends who make me enjoy a lot of aspects of life outside of work.

Mohamed Tarek Ibn Ziad Mohamed Hassan
Computer and Systems Engineering Department
Faculty of Engineering
Ain Shams University
Cairo, Egypt
June 2017

# Contents

**Abstract** ix

**Summary** xi

**Acknowledgment** xiii

**Table of Contents** xiv

**List of Figures** xix

**List of Tables** xxi

**List of Abbreviations** xxiii

**List of Symbols** xxv

**1 Introduction** 1
- 1.1 Motivation . . . 1
- 1.2 Main Contributions . . . 2
- 1.3 Thesis Organization . . . 3

**2 Background** 5
- 2.1 Fully Homomorphic Encryption (FHE) . . . 5
- 2.2 Partially Homomorphic Encryption (PHE) . . . 7
  - 2.2.1 ElGamal Scheme . . . 8
  - 2.2.2 CRT-based ElGamal (CEG) Scheme . . . 9
  - 2.2.3 Paillier Cryptosystem . . . 9
- 2.3 Hardware Trojan . . . 11
- 2.4 Hardware Trojan Taxonomy . . . 11
  - 2.4.1 Phase of Insertion . . . 11
  - 2.4.2 Level of Abstraction . . . 14
  - 2.4.3 Methods of Activation . . . 15
  - 2.4.4 Effects . . . 16
  - 2.4.5 Location . . . 16

**3 E-voting Attacks and Countermeasures** 17

3.1 Motivation . . . 18
3.2 Related Work . . . 19
3.3 E-voting System Overview . . . 21
3.4 Scenario for a Possible Attack . . . 23
3.5 Protection Against Proposed Attack . . . 24
3.6 Other Attacks and Countermeasures . . . 25
3.6.1 Sequence Cheat Code Attack . . . 26
3.6.2 Used Bits Attack . . . 26
3.7 Evaluation . . . 28
3.7.1 Experimental Setup . . . 28
3.7.2 E-voting Protection Results . . . 29
3.7.2.1 Resetting Unused Bits . . . 29
3.7.2.2 Using Enhanced SB Method . . . 31
3.8 Conclusion . . . 32

**4 Homomorphic Data Isolation for Protection against Hardware Trojans 33**
4.1 Motivation . . . 34
4.2 Related Work . . . 34
4.3 HT Protection using PHE Overview . . . 36
4.4 HT Protection using PHE Methods . . . 37
4.4.1 Sufficient PHE Support . . . 37
4.4.1.1 ElGamal Scheme Implementation . . . 38
4.4.1.2 CEG Scheme Implementation . . . 39
4.4.2 Dual-Circuit Design . . . 40
4.5 Evaluation . . . 41
4.5.1 Experimental Setup . . . 41
4.5.2 PHE Methods Results . . . 42
4.5.3 Dual-Circuit Design Results . . . 42
4.6 Conclusion . . . 45

**5 *CryptoImg*: Privacy Preserving Processing over Encrypted Images 47**
5.1 Motivation . . . 48
5.2 Related Work . . . 49
5.3 ***CryptoImg*** System Overview . . . 50
5.4 Secure Operations in Encrypted Domain . . . 52
5.4.1 Secure Image Adjustment . . . 52
5.4.2 Secure Noise Reduction . . . 53
5.4.3 Secure Edge Detection and Sharpening . . . 54
5.4.4 Secure Morphological Operations . . . 55
5.4.5 Secure Histogram Equalization . . . 56
5.5 Evaluation . . . 56
5.5.1 Experimental Setup . . . 56
5.5.2 Visual Output Results . . . 57
5.5.3 Computation Time Results . . . 59
5.6 Conclusion . . . 61

**6 Conclusion and Future Work** **63**
6.1 Conclusion . . . . . 63
6.2 Contributions . . . . . 63
6.2.1 Secure E-voting . . . . . 64
6.2.2 Secure FPGA-based Designs . . . . . 64
6.2.3 Secure Image Processing . . . . . 64
6.3 Future Work . . . . . 65
6.3.1 Secure E-voting . . . . . 65
6.3.2 Secure FPGA-based Designs . . . . . 65
6.3.3 Secure Image Processing . . . . . 65

**Bibliography** **67**

# List of Figures


2.1 Possible stages for launching Hardware Trojan attacks by third parties. . 12
2.2 Hardware Trojan taxonomy. . . . . 13

3.1 E-voting versus regular voting. . . . . 19
3.2 The e-voting process. . . . . 22
3.3 Abstract view of our proposed e-voting machine. . . . . 22
3.4 MicroBlaze block diagram. . . . . 23
3.5 Block diagram of using a malicious secret core within an untrusted e-voting machine. . . . . 24
3.6 Simple obfuscating function depending on data xoring. . . . . 25
3.7 Inverse obfuscating function in order to retrieve data. . . . . 25
3.8 E-voting protection against proposed attack. . . . . 25
3.9 Experimental setup for the e-voting system. . . . . 28
3.10 The Xilinx Spartan 3E starter kit overview. . . . . 29

4.1 Homomorphic encryption to protect from Hardware Trojans. . . . . 37
4.2 Block diagram for ElGamal encryption/decryption scheme. . . . . 38
4.3 Block diagram for the CRT-based ElGamal (CEG) encryption/decryption scheme. . . . . 40

5.1 System Architecture of ***CryptoImg***. . . . . 51
5.2 The different edge detection operators supported by ***CryptoImg***. . . . . 54
5.3 Visual output evaluation for the first operations set applied in PD and ED. 57
5.4 Visual output evaluation for the second operations set applied in PD and ED. . . . . 58

# List of Tables


3.1 Device utilization comparison for resetting unused bits method. . . . . 30
3.2 Power comparison for resetting unused bits method. . . . . 30
3.3 Device utilization comparison for enhanced Simple Blockage method. . 31
3.4 Power comparison for enhanced Simple Blockage method. . . . . 31

4.1 Resource utilization of ElGamal and CEG schemes. . . . . 42
4.2 Timing performance of ElGamal and CEG schemes. . . . . 42
4.3 Power consumption (mW) of ElGamal and CEG encryption/decryption schemes. . . . . 43
4.4 Area reduction of our dual ElGamal design over the regular ElGamal design. . . . . 43
4.5 Timing comparisons between our dual ElGamal design and the regular ElGamal design. . . . . 44
4.6 Power consumption (mW) of our dual ElGamal design and the regular ElGamal design. . . . . 44

5.1 Precision effect on the introduced ***CryptoImg*** error. . . . . 59
5.2 Execution Time (sec) of the Paillier encryption/decryption of an image. 60
5.3 Execution Time (sec) of the proposed ***CryptoImg*** operations on different clients. . . . . 60

# List of Abbreviations

| | |
|---|---|
| **AES** | **A**dvanced **E**ncryption **S**tandard |
| **ASIC** | **A**pplication **S**pecific **I**ntegrated **C**ircuit |
| **CBIR** | **C**ontent-**B**ased **I**mage **R**etrieval |
| **CEG** | **C**RT-based **E**l**G**amal |
| **CRC** | **C**yclic **R**edundancy **C**heck |
| **CRT** | **C**hinese **R**emainder **T**heorem |
| **DCR** | **D**ecisional **C**omposite **R**esiduosity |
| **DLP** | **D**iscrete **L**ogarithm **P**roblem |
| **DoS** | **D**enial **o**f **S**ervice |
| **ED** | **E**ncrypted **D**omain |
| **EDA** | **E**lectronic **D**esign **A**utomation |
| **EM** | **E**stimation **M**aximization |
| **EN** | **E**ncoding **F**unction |
| **E-voting** | **E**lectronic **v**oting |
| **EVM** | **E**lectronic **V**oting **M**achine |
| **FHE** | **F**ully **H**omomorphic **E**ncryption |
| **FP** | **F**loating **P**oint |
| **FSM** | **F**inite **S**tate **M**achine |
| **FSL** | **F**ast **S**implex **L**ink |
| **HT** | **H**ardware **T**rojan |
| **IP** | **I**ntellectual **P**roperty |
| **ISE** | **I**ntegrated **S**ynthesis **E**nvironment |
| **JTAG** | **J**oint **T**est **A**ction **G**roup |
| **LPF** | **L**ow **P**ass **F**ilter |
| **LWE** | **L**earning **W**ith **E**rrors |

| | |
|---|---|
| **Mob** | **Mob**ile device |
| **NLM** | **N**on-**L**ocal **M**eans |
| **OS** | **O**perating **S**ystem |
| **PC** | **P**ersonal **C**omputer |
| **PD** | **P**laintext **D**omain |
| **PHE** | **P**artially **H**omomorphic **E**ncryption |
| **PCB** | **P**rinted **C**ircuit **B**oard |
| **RO** | **R**ing **O**scillators |
| **RISC** | **R**educed **I**nstruction **S**et **C**omputer |
| **RLWE** | **R**ing **L**earning **W**ith **E**rrors |
| **RNS** | **R**esidue-**N**umber **S**ystem |
| **RSA** | **R**ivest-**S**hamir-**A**dleman |
| **RTL** | **R**egister **T**ransfer **L**evel |
| **SaaS** | **S**oftware-**a**s-**a**-**S**ervice |
| **SB** | **S**imple **B**lockage |
| **SIFT** | **S**cale-**I**nvariant **F**eature **T**ransform |
| **SMC** | **S**ecure **M**ulti-party **C**omputation |
| **SRAM** | **S**tatic **R**ead **A**ccess **M**emory |
| **SSS** | **S**hamir's **S**ecret **S**haring |
| **SHE** | **S**omewhat **H**omomorphic **E**ncryption |
| **TRNG** | **T**rue **R**andom **N**umber **G**enerator |
| **XPA** | **X**ilinx **P**ower **A**nalyzer |
| **XPS** | **X**ilinx **P**latform **S**tudio |

# List of Symbols

| | |
|---|---|
| $c$ | ciphertext message |
| $F$ | frequency |
| $G_x$ | horizontal gradient component |
| $G_y$ | vertical gradient component |
| $g$ | generator |
| $h_1$ | horizontal kernel |
| $h_2$ | vertical kernel |
| $I$ | image |
| $i$ | image pixel |
| $k$ | secret random exponent |
| $l$ | secret random exponent |
| $m$ | plaintext message |
| $n$ | prime number |
| $o$ | image pixel |
| $r$ | secret random exponent |
| $R$ | result image |
| $T$ | image transformation |
| $x$ | plaintext number |
| $\Theta$ | gradient direction |
| $\phi_{add}$ | addition operation error |
| $\phi_{mul}$ | multiplication operation error |
| $\otimes$ | self binding over floats |
| $\oplus$ | additive homomorphic over floats |

*To my princess*

# Chapter 1

# Introduction

The principal target of our introductory chapter is to highlight the motivation behind this work, state the main contributions of the Thesis, and define its organization.

## 1.1 Motivation

The keyword “Homomorphism” originally comes from the concatenation of two ancient Greek words; *homos*, which means “same”, and *morphe*, which means “shape” or “form” [1]. So, homomorphism could be interpreted as “the same shape”. The word was first used in algebra, where it means a certain transformation of one algebraic set to another one preserving all the relations between the first set elements in the second one too [2]. In the domain of cryptography, Homomorphism is used along with encryption. Homomorphic encryption is the notation used to describe the kind of encryption, which can be used to perform different arithmetic operations on encrypted data to directly obtain an encrypted result. Thus, using such encryption techniques enables the execution of specific computations, while maintaining the privacy of both the input data and the results. Depending on the number of arithmetic computations that are supported by an algorithm, a homomorphic encryption can be considered as either fully homomorphic encryption (FHE) or partially homomorphic encryption (PHE).

Nowadays, the homomorphic property of various cryptosystems, such as ElGamal and Paillier encryption schemes, is used to build many applications, such as secure voting systems [3], introduce privacy-preserving face recognition [4], fingerprint recognition [5], zero-knowledge watermarking [6], and location-based services. Furthermore, the use of cloud computing services, in which computations performed on user data are

outsourced to a public cloud, raises the need for a final solution to maintain the privacy of user data. On one hand, clients have many privacy and security concerns over their data. On the other hand, third-parties are vulnerable to malicious interventions or monitoring like hacking or eavesdropping. Combining those two opposite requirements together establishes one of the main motivations behind this work.

While FHE can help solve privacy issues, it is also desirable to reduce the performance overhead introduced by such methods. Thus, it is a good practice to utilize PHE techniques in the desired applications, instead of the FHE ones, to avoid such overheads. As a result, we started this research with the aim of designing and implementing efficient algorithms to use homomorphic encryption techniques in performing secure computations over encrypted data. We selected three different areas of applications; securing electronic voting (e-voting) systems, defeating Hardware Trojans (HTs) in FPGA-based designs, and operating blindly over encrypted images. We only used PHE techniques to reach our goals, as will be shown in coming chapters.

## 1.2 Main Contributions

The contributions of this Thesis could be categorized into three main parts as follows:

1. **Secure e-voting**
   (a) Implementing an e-voting machine using Xilinx FPGA board.
   (b) Injecting an HT within the FPGA design to tamper voting results.
   (c) Providing a protection technique against the proposed attack.
   (d) Showing the different overheads resulting from the protection technique, such as area, timing, and power.
   (e) Introducing two other attacks and their protection scenarios.
2. **Secure FPGA-based designs**
   (a) Discussing new ideas to have a blind data processing by the third party IP with a minimal cost.
   (b) Implementing ElGamal encryption scheme, which is a multiplicative PHE scheme and the CRT-based ElGamal (CEG) encryption scheme, which is additive PHE scheme, on a low-cost FPGA and showing the resource utilization, timing performance, and power analysis of both schemes.

(c) Introducing a dual-circuit design that supports both, multiplicative and additive homomorphic properties and providing the obtained savings on area and power over a regular design that has no resource sharing.

3. **Secure image processing**

   (a) Proposing a secure framework to perform image processing computations over images stored on a third-party server based on Paillier cryptosystem.

   (b) Supporting basic image processing operations such as image adjustment operations, spatial filtering, edge detection, morphological operations, and histogram equalization.

   (c) Discussing the benefit of integrating Paillier cryptosystem with a floating-point support in decreasing the pre/post processing in the similar work mentioned in the literature.

   (d) Introducing a user friendly Android application to submit encrypted images for secure processing.

## 1.3 Thesis Organization

The Thesis is organized as follows. Chapter 2 describes the needed background about the PHE cryptosystems, used in this work. Chapter 3 illustrates the first contribution, which is using homomorphism in E-voting systems. It introduces a couple of possible attacks and countermeasures to an FPGA-based voting machine. The full hardware implementation is described and the countermeasures' overheads are highlighted. After that, Chapter 4 introduces the second contribution, which is securing FPGA-based designs from Hardware Trojans (HTs) using homomorphism. The design realization on a low-cost FPGA is described and the area/timing results are presented. Then, Chapter 5 describes the third contribution, which is a cloud-based library, ***CryptoImg***, that allows performing image processing operations over encrypted images. New algorithms are introduced and the communication/computation overhead are stated using various test cases. Finally, Chapter 6 concludes the work and states some possible directions for future work.

# Chapter 2

# Background

The aim of this chapter is to give a brief description of the idea of homomorphism, survey existing fully and partially homomorphic encryption schemes, with emphasis on the ElGamal and Paillier security schemes. Moreover, the chapter highlights the different attributes used to classify Hardware Trojans, as they are our main topic of interest in Chapters 3 and 4.

## 2.1 Fully Homomorphic Encryption (FHE)

As the most commonly used computations include the usage of both kind of basic operations; the addition and the multiplication, partially homomorphic encryption (PHE) techniques are not sufficient. As a result, FHE was introduced by Rivest, Adleman, and Dertouzos around forty years ago [7]. However, the research community had to wait till 2009. when it has been revisited again by Gentry [8]. The most attracting point regarding Gentry's proposal is that it was the first feasible FHE cryptosystem. In general, a FHE technique is a kind of encryptions, which provides the capabilities to perform any operation directly on encrypted data by converting it into a circuit of a certain depth.

In general, FHE includes four basic algorithms: `Keygen`, `Encrypt`, `Decrypt`, and `Eval`. The `Keygen` algorithm is responsible for generating the required keys. The `Encrypt` and `Decrypt` algorithms are used for encrypting and decrypting the plaintext messages, respectively. The `Eval` algorithm is built based on three different algorithms: `Add`, `Mult`, and `Recrypt`. The `Add` and `Mult` algorithms are used for addition and multiplication operations over ciphertexts, while the `Recrypt` operation cleans the ciphertext from the noise due to the homomorphic addition and multiplication operations. Without this

function the scheme would be Somewhat Homomorphic and therefore it would only evaluate circuits of a fixed depth.

Since its introduction in 2009, there were many trials to introduce more FHE techniques. For instance, Van Dijk *et al.* introduced a FHE technique based on ideals defined over integers [9]. Gentry *et al.* enhanced his FHE implementation by adding many tricks to overcome the performance degradation issues of FHE [10]. However, they did not fully managed to handle them. For instance, the key generation process in this implementation required an overall time of two seconds in the best case and two hours in the worst case. The main cause of this low performance was due to the noise, which resulted from performing new successive homomorphic operations. To get rid of such noise, FHE cryptosystems used to recrypt the messages after every homomorphic operation. The overheads corresponded to the *Recrypt* operation was very high and affected the whole system performance.

One shot towards reducing the FHE overheads was the introduction of the somewhat homomorphic encryption (SHE) cryptosystems. They have less overheads compared to FHE cryptosystems, as they eliminate the need for the high cost *Recrypt* operation. As a side effect, the SHE techniques can only perform a limited number of homomorphic operations. Many authors in the literature tried to tackle the noise problem by introducing new SHE techniques. For instance, Brakerski *et al.* suggested using the concept of learning with errors (LWE) to reduce the accumulative noise [11]. This new idea allowed computing more operations on encrypted data, i.e., circuits with deeper depth, using the lower overheads of the SHE cryptosystems.

In 2012, López-Alt, *et al.* introduced a SHE cryptosystems, which handle inputs from more than one public key [12]. Bos *et al.* proposed another FHE scheme, in which the security of the system only relies on the standard assumptions of lattice [13]. In the same context, three researchers led by Gentry described a LWE-based FHE scheme, which is highly customized to evaluate the advanced encryption standard (AES) circuit efficiently [14]. The results reported in their paper showed that their proposal was capable of successfully evaluating a single AES encryption operation within five minutes.

Although new FHE and SHE cryptosystems had made a noticeable enhancements compared to previous ones, the large ciphertext sizes and timing overheads remain a major obstacles in the way of using such cryptosystems. Those overheads are very high while being compared to the overheads of the PHE cryptosystems. This opens the door for using hardware to accelerate the computations of FHE techniques. For example, Göttert *et al.* implemented FPGA-based modules for lattice-based computations, which represented a major part of the FHE cryptosystems [15]. Moreover, Pöppelmann and

Güneysu introduced an efficient hardware implementation of a FHE technique based on the ring-learning-with-errors (RLWE) theorem [16]. However, these implementations did not achieve a stable state that allowed them to be used efficiently in real-world applications. Thus, we focus on PHE in this Thesis.

## 2.2 Partially Homomorphic Encryption (PHE)

As mentioned before, PHE has been introduced many years ago. It gives the chance to perform only one kind of operations, either addition or multiplication, on ciphertexts without revealing data. For example, let us consider the two messages, $m_1$ and $m_2$, where both messages are encrypted and their ciphertexts are given by $E(m_1)$ and $E(m_2)$, respectively. If the multiplication of the two ciphertexts is equivalent to the ciphertext of the multiplication of the two messages as shown in (2.1), we call this a multiplicative homomorphic scheme. On the other hand, if the multiplication of the two ciphertexts equals the ciphertext of the addition of the two messages as shown in (2.2), we call this an additive homomorphic scheme.

$$E(m_1) \times E(m_2) = E(m_1 \times m_2) \tag{2.1}$$

$$E(m_1) \times E(m_2) = E(m_1 + m_2) \tag{2.2}$$

One of the earliest discoveries in the area of PHE was the Goldwasser-Micali encryption scheme [17]. This technique offers the ability to perform homomorphic operations with respect to the bitwise xor logic operation. The security of the former technique relied on the "quadratic residuosity problem". There also exist PHE techniques that support another kind of operation, the addition operation. Benaloh [18] and Paillier [19] cryptosystems are two stunning examples for the additive PHE methods.

On the other hand, there exist two well-known schemes, which are multiplicative homomorphic schemes. The first one is the Rivest-Shamir-Adleman (RSA), which is considered as one of the most excessively used public-key cryptosystems [20]. The second is ElGamal encryption scheme [21]. In this Thesis, we selected ElGamal and Paillier cryptosystems to be used in our proposed solutions.

### 2.2.1 ElGamal Scheme

ElGamal cryptosystems is an efficient and widely used technique, which has many applications in different domains. To illustrate its functionality, let us assume that we have two users, called *Alice* and *Bob*, respectively. *Alice* has a certain message $m$ and she would like to send it to *Bob*. ElGamal process works as follows. *Bob* generates his keys by choosing a secret random exponent $k$ and a generator $g$. So, his public key is $(g,h)$, where $h = g^k (mod\, n)$ and $n$ is a large prime. Before sending $m$ to *Bob*, *Alice* must generate a random exponent $l$ and sends the ordered pair $(c_1, c_2)$ to *Bob*, where $c_1$ and $c_2$ are defined by (2.3) [21].

$$\begin{aligned} c_1 &= g^l (mod\, n) \\ c_2 &= h^l \times m (mod\, n) \end{aligned} \tag{2.3}$$

*Bob* can easily decrypt the ciphertext using (2.4).

$$m = {c_1}^{-k} \times c_2 (mod\, n) \tag{2.4}$$

This PHE scheme is considered homomorphic with respect to multiplication because if $(x_1, y_1)$ and $(x_2, y_2)$ are valid encryptions for messages $m_1$ and $m_2$, with the same key, then $(x_1 x_2, y_1 y_2)$ is a valid encryption of $m_1 m_2$ [21].

Hu *et al.* proposed a simple modification to make ElGamal additively homomorphic by placing the message $m$ in the exponent [22]. So, if we encrypt two messages $m_1$ and $m_2$ using (2.3) but multiply $h^l$ with $g^m$ instead of $m$, the multiplication of the two ciphertexts results in a valid encryption of $g^{m_1+m_2}$. The problem here is that recovering the message requires finding the solution of the Discrete Logarithm Problem (DLP). ElGamal security itself is built upon the hardness of the DLP. To solve this problem, they introduced a new scheme, called CRT-based ElGamal (CEG) scheme, which depends on the Chinese Remainder Theorem (CRT). They managed to convert a single large-space DLP problem into multiple small-space DLP problems. This allows obtaining $m_1 + m_2$ easily, while retaining the full security of the scheme, as shown later.

### 2.2.2 CRT-based ElGamal (CEG) Scheme

To illustrate how CEG works, let us reuse the previous example of *Alice* and *Bob*. In the first step, *Bob* selects a secret random exponent $k$ along with a generator $g$. He also chooses $d_i$ for $i = 1,\ldots,t$, such that $gcd(d_i,d_j) = 1$ for $i \neq j$. So, *Bob*'s public key is $(g,h,(d_1,\ldots,d_t))$, where $h = g^k (mod\, n)$ and $n$ is a large prime. For encryption, *Alice* sends the encryption of message $m$ as a t-tuple of pairs $(c_1,c_2)$ by using (2.5).

$$\begin{aligned} c_1 &= g^{l_i} (mod\, n) \\ c_2 &= h^{l_i} \times g^{m_i} (mod\, n) \end{aligned} \tag{2.5}$$

where $m_i = m\ (mod\ d_i)$ and $l_i$ is a generated random exponent for $i = 1,\ldots,t$. *Bob* can decrypt the ciphertext using (2.6) and (2.7).

$$m = CRT^{-1}[(log_g(c_{2_i} \times c_{1_i}^{-k} (mod\, n)), i = 1,\ldots,t)] \tag{2.6}$$

$$CRT^{-1}[c_i] = \sum_{i=1}^{t} c_i \frac{d}{d_i} (\frac{d}{d_i}^{-1} mod\ d_i)\, mod\ d \tag{2.7}$$

Correctness and efficiency of the illustrated scheme is discussed in details in [22]. As part of this work, we implement the CEG scheme in hardware and show its resource utilization and power consumption in Chapter 4.

### 2.2.3 Paillier Cryptosystem

Our ***CryptoImg*** system, described in Chapter 5, relies upon the homomorphic properties of the Paillier cryptosystem [19]. Paillier cryptosystem is one of the most widely used PHE schemes.

To briefly describe the Paillier cryptosystem, *Alice* can select any two large prime numbers, for example $p$ and $q$. Then, let $N = pq$. Now, we can define a new domain $Z_{N^2}$, where $Z_{N^2} = \{0,1,\ldots,N^2-1\}$. The set of non-negative integers, which have multiplicative inverse modulo $N^2$ is denoted by $Z_{N^2}^*$, where $Z_{N^2}^* \subset Z_{N^2}$. Then, she needs to select a number $g$ from $Z_{N^2}^*$, where $g$ satisfies the conditions in (2.8).

$$gcd(L(g\lambda \, mod \, N^2), N) = 1 \tag{2.8}$$

where $L(u) = \frac{u-1}{N}$ and $\lambda = lcm(p-1, q-1)$. By the end of the key generation process, *Alice* would have the pair $(g, N)$ as a public key and $\lambda$ as a private key. It is worth mentioning that the length of $N$ should be greater than 1024 bits to ensure a powerful security level.

Before sending the message $m$ to *Bob*, *Alice* should encrypt it using (2.9)

$$\begin{aligned} c &= E(m, r) \\ &= g^m r^N mod N^2 \end{aligned} \tag{2.9}$$

where $c \in Z_{N^2}$ denotes the ciphertext and $r$ is another random exponent. It is worth noting that the Paillier cryptosystem is a provable semantically secure encryption system whose security guarantees are proven based on the computational hardness assumption of the Decisional Composite Residuosity (DCR) problem.

Equation (2.10) gives an optimized notation for the encryption process.

$$c = E(m, r) \equiv [\![m]\!] \tag{2.10}$$

For *Bob* to obtain the actual message $m$ from the ciphertext, $c$, he should compute the operations mentioned in (2.11).

$$\begin{aligned} m &= D(c, \lambda) \\ &= \frac{L(c^\lambda mod N^2)}{L(g^\lambda mod N^2)} mod N \end{aligned} \tag{2.11}$$

Paillier cryptosystem is an additive HE scheme as it provides a public-key operation $\oplus_z$ over two encrypted integers, which is equivalent to their plain-text addition, as shown in (2.12). It also supports a self-blinding operation $\otimes_z$, which allows multiplication of encrypted integer by a plaintext scalar $d$, as shown in (2.13) $\forall m_1, m_2 \in \mathbb{Z}_N$ [19].

$$\begin{aligned} \texttt{DEC}(\llbracket m_1 \rrbracket \oplus_z \llbracket m_2 \rrbracket) &= \texttt{DEC}((\llbracket m_1 \rrbracket \times \llbracket m_2 \rrbracket) mod\, n^2) \\ &= (m_1 + m_2)\, mod\, n \qquad (2.12) \\ \texttt{DEC}(\llbracket m_1 \rrbracket \otimes_z d) &= \texttt{DEC}(\llbracket m_1 \rrbracket^d\, mod\, n^2) \\ &= (m \times d)\, mod\, n \qquad (2.13) \end{aligned}$$

## 2.3 Hardware Trojan

Nowadays, HT is considered one of the hot research directions due to the popularity of the usage of hardware ICs in various domains [23]. Those applications include, but not limited to, cars, cell phones, satellites, medical devices, and strategically military components.

HT is simply defined as a malicious alteration of one's own hardware. This alternation may, under specific rare circumstances, result in information leakage out of the system or functional changes of the system itself [24]. As shown in Figure 2.1, those kinds of malicious circuitry can be injected by either third party IP owners or fabrication facilities. This obviously threatens the entire design community.

## 2.4 Hardware Trojan Taxonomy

The aim of this section is to make the reader familiar with the different types of HTs in general and to briefly highlight the various categories that could be used to classify them.

Generally speaking, HTs can be divided into five main categories, as shown in Figure 2.2. As introduced by Karri *et al.*, the classification is based on phase of insertion, level of abstraction, methods of activation, effects, and location [26].

### 2.4.1 Phase of Insertion

In order to have a fabricated IC in hand, the chip passes through many phases. Those phases are specification, design, fabrication, testing, and packaging. Each phase of them introduces a possible chance for an attacker to insert an HT.

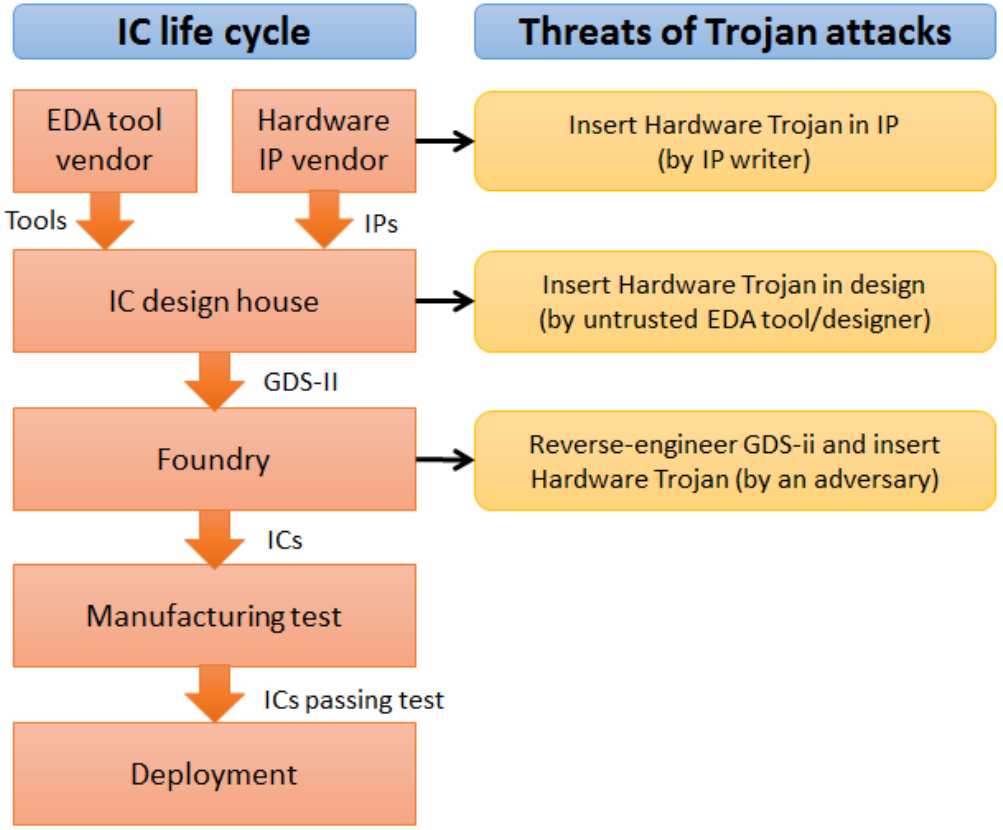



FIGURE 2.1: Possible stages for launching Hardware Trojan (HT) attacks by third parties during IC life cycle [25].

Firstly, the **specification phase**, could be described as the phase in which the designer states the main the properties of the IC, such as the desired functionality, the maximum amount of power consumption, the allowable size, and the accepted delay. One possible way to insert an HT here is to manipulate one of the required constraints of the IC. In our proposed solution of Chapter 4, we care about the data itself, which is being encrypted using a PHE scheme before traveling to the suspected third party IPs. Thus, manipulating the requirements of those third party IPs in the specification phase would not leak any further information.

After that, the **design phase** comes to business. All the previously prepared characteristics are taken into consideration by the designers while writing the hardware codes. This phase may also require the usage of ready-made implementations from different vendors. That opens the door for inserting unwanted HTs. Again, this will not affect our solution as even if the used IPs are suspicious, they will know nothing about our encrypted data.

The **fabrication** and **assembly phases** are more related to the ASIC flow, where complete designs are moved to the fabrication facilities in order to produce them. If such

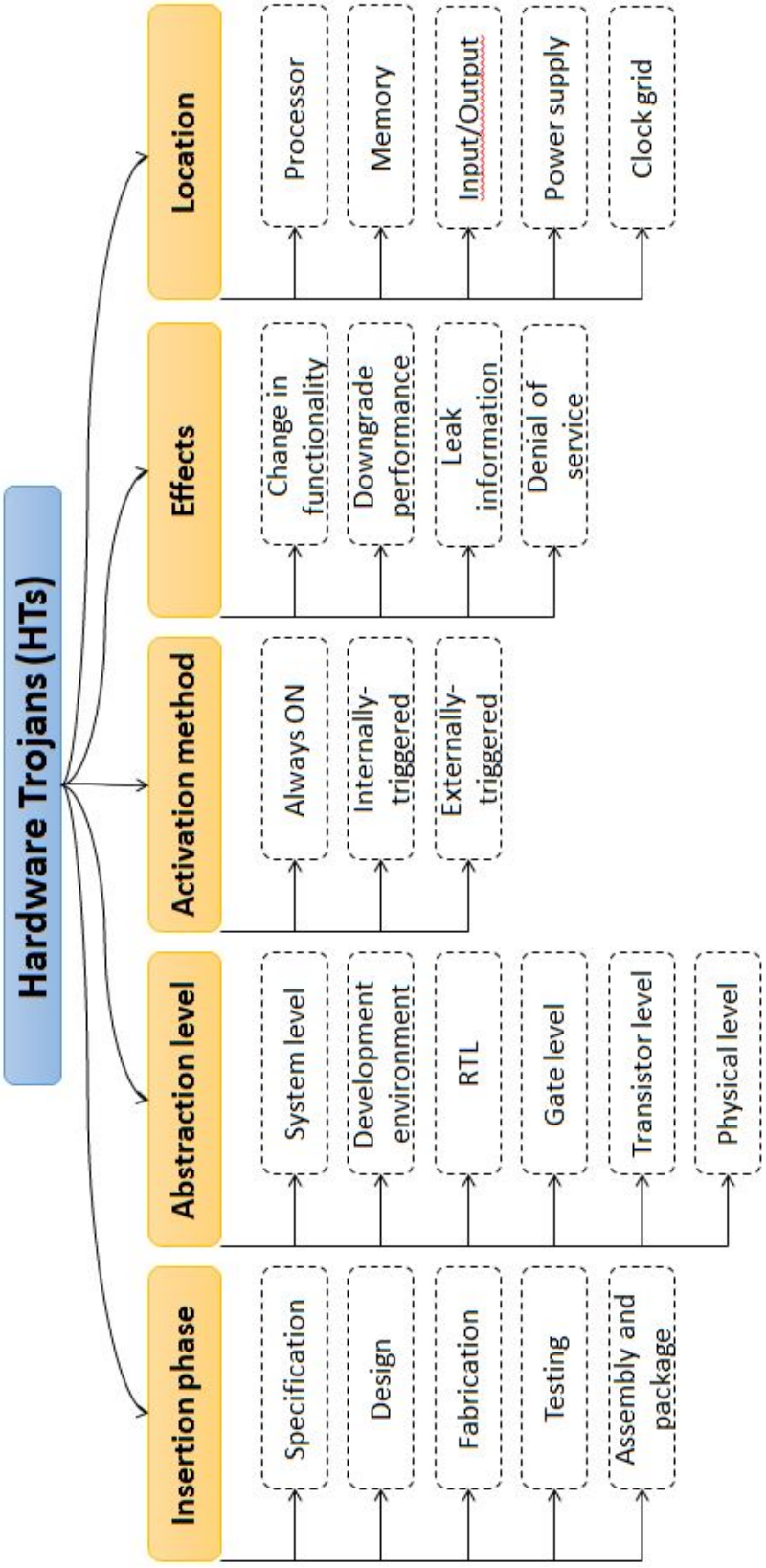



FIGURE 2.2: Hardware Trojan taxonomy [27].

facilities are untrusted, they may add more logic to the users' design before fabricating them. Additionally, they may connect the outputs of the users' IP to a hidden node during the assembly process. This node could be used later to leak important information about the IP behavior. So, we need to make sure that at least the encryption/decryption hardware architectures that we use within our solution is produced at home or using trusted manufacturers.

Finally, the **testing phase** also may be used by an intruder to bypass certain testing vectors that might lead to the discovery of his/her own HT inserted in a previous phase. So, it is very critical to insure the trustworthy of the testing engineers and make sure that the testing coverage is good enough. For our proposed solution, we can safely work with malicious third party IPs as our data is always in an encrypted form.

### 2.4.2 Level of Abstraction

Here, we move to the second class of HT taxonomy, where the target level of abstraction plays the main role. For the attacker, he/she can insert the HT within different levels, such as the system level, development environment level, register-transfer level (RTL), gate level, transistor level, and physical level.

On the **system level**, main IPs are well defined and the required protocols of communication are selected. Any change in those pre-stated points might lead to a misbehavior of the final circuit. One rigid example is introduced in Chapter 3 where the unused bits of the input data could be used to launch an attack on the e-voting machine. This should not happen as those bits are designed to be unused by any system module in the first place.

The **development environment** is a generic name enclosing the processes of simulation, synthesis, placement and routing, etc. The mentioned processes are done using electronic design automation (EDA) tools, which might be used by an outsider to insert HTs. Furthermore, malicious simulation tools could be designed in order to bypasss certain rare conditions that might be a root for HT.

Regarding the **register transfer level (RTL)**, it gives the attacker high flexibility to affect the functionality of the original IP. An intruder can insert any additional logic to the actual code to act as an HT. The added logic can be able to work independently or wait for a certain trigger as will be highlighted in a coming categorization [28]. The same rules are applicable for the **gate level**, in which the attacker can monitor the inputs, outputs, and interconnections between design components.

As the gates are mainly built out of transistors, the **transistor level** is a possible level at which an HT could be inserted. In this case, the HT could appear as the usage of a transistor with different electrical characteristics than the golden one. That might have a noticeable effect on the implementation timing delay and power consumption.

At the **physical level**, the final layout of the circuit is determined. This includes the real size of each component, their locations on the board, and the spacing between each component and its neighbors. An attacker with access to this level of abstraction can modify the dimension of a single wire/component in order to disable the complete IP.

### 2.4.3 Methods of Activation

One efficient way to classify HTs is to investigate the mechanisms, which are used by HT owner to activate them. In this context, we have three main types of HTs; the "always-on" HT, the "internally-triggered" HT, and the "externally-triggered" HT.

The **always-on** HTs are up and running throughout the life of the infected IP. For example, any HT that is inserted in the physical level could fall under this category.

On the other hand, there exist HTs that remain silent until a certain condition occurs. They are called "triggered" HTs. Those kind of Trojans are very dangerous as the majority of them depends on rare conditions, which could not be discovered in the regular testing phase, to be activated.

If the event, needed to trigger the HT, comes from an outside source, we called those HTs, **externally-triggered**. The e-voting example of Chapter 3 best describes this kind of Trojans, where the HT within the Microblaze core is only activated when it receives a pre-defined trigger from the input keypad.

On the other hand, the **internally-triggered** HTs become active only if a certain condition is reached from within the infected architecture itself. For instance, an HT could be activated when the internal finite state machine (FSM) of the implementation reaches a certain state or when an internal counter reaches a certain large value.

Finally, it is worth mentioning that some "triggered" HTs continue to work after being activated, while others return back to their silent state after a certain amount of time.

### 2.4.4 Effects

The major issue caused by HTs is their malicious side effects. Hence, it would be helpful to use the different evil effects caused by HTs to classify them. Although there are many undesirable effects for HTs, we will only consider the widely-known ones.

On the top of the list comes the **change in functionality** effect. An HT injected in any fabrication phase could easily change one or more of the IP functions to completely different ones. The functionality change might be as small as altering the value of a single IP output or as large as disabling the whole system.

**Downgrading performance** is a well-known side effect of HTs. A Trojan of this type might increase the architecture critical path or add more useless sequential logic to consume larger amounts of power. Furthermore, HTs could be designed to **leak information** regarding the working IP. The types of leaked information are numerous, such as the applied voltage/current, operational frequency, etc. Those information could be used to estimate the actual functionality of the infected IP. Moreover, HTs might leak sensitive users' information while the infected IPs are running, so users privacy are badly affected.

Furthermore, HTs could be used to cause **denial-of-service** (DoS). That means an HT forcing a certain module to consume a limited circuit resource, like (current, power, network bandwidth). This will cause the rest of the architecture resources to be out of service and hence decreasing system reliability.

### 2.4.5 Location

Trojans can also be categorized based on their actual locations within the targeted architecture. A global HT could affect the whole design, while a local HT is limited to a certain part of the architecture. This part could be the main **processor** core [29], different **memory** blocks, **input/output** ports, **power supply**, or the **clock grid**. Based on the HT location, one could estimate the dangerous of it. An HT inserted within the power supply circuit might disable the whole architecture, while an HT hidden within the memory could leak sensitive information about the users' data.

It is obvious that a certain HT could be successfully classified to more than one of the previously mentioned categories. For instance, an HT could be inserted in the main processor within the design phase, be written as an RTL code, be triggered by an external event, and cause a performance downgrade.

# Chapter 3

# E-voting Attacks and Countermeasures

Since more than 50 years, electronic voting (e-voting) systems have been first introduced. Many changes have been made upon them over the decades. They started to be widely used as they do offer various advantages over the traditional voting methods. However, e-voting also introduces many security challenges that need to be handled wisely, otherwise, it might bomb the whole voting process. E-voting machines may contain harmful back-doors, which can affect the dependability of the system.

Through this chapter, one of the e-voting challenges is introduced; the existence of a hardware Trojan (HT) that totally tampers the voting results. We used a Xilinx FPGA board to implement a simple e-voting machine. Here, the idea of homomorphism appears via the usage of CRT-based ElGamal (CEG) cryptosystem to encrypt the votes before sending them to the main server. We inject an HT within the FPGA design to tamper voting results. We provide a couple of protection mechanisms and evaluate them by showing their overheads. Our solution adds about 4% as per logic resources and less than 10% as per timing delay. The additional power consumption is almost negligible. Furthermore, we highlight the differences between our proposed protection mechanisms and other techniques from the literature.

Starting with the motivation in Section 3.1, we make the literature review in Section 3.2. We then introduce a full e-voting system implementation in Section 3.3. After that, we show a simple scenario for an untrusted machine and how it would be used to affect the election results in Section 3.4. We introduce protection against the proposed Trojan in Section 3.5. Furthermore, we suggest other attacks and countermeasures in Section 3.6. The evaluation is mentioned in Section 3.7. Finally, Section 3.8 concludes the chapter.

## 3.1 Motivation

Democracy as an expression means "ruling by people". Citizens need to have access to concrete information and to be capable of freely selecting their representatives to claim living in a democratic environment. Democracy itself mainly depends on the election process to satisfy population needs. Elections give the advantages for the citizens to freely select their representatives. No one can deny that the election process integrity is very important to ensure the integrity of democracy itself. Additionally, for the populace to accept the election results, the election system itself must fulfill a set of requirements. These requirements include, but are not limited to, transparency, robustness, and venerability. Without those features, the election system would be very questionable. Through out the mankind history, there existed a lot of election examples, which had been manipulated in order to redirect their output. Thus, designing an "acceptable" voting system, whatever its basis is electronic or paper-based system, is a critical operation that must go through many dedicated filters and competing criteria to verify it.

As shown in Figure 3.1, when a voting system uses a computerized element to either record, check, or collect votes, we can call this system an e-voting one. These types of systems started to become widely used as they introduce many advantages to all election parties; citizens, candidates, and election administrators [30]. Citizens seem to prefer electronic voting systems due to their privacy and accessibility. Candidates and election administrators usually enjoy the efficiency and speed of the entire e-voting process. Administrators, specifically, prefer e-voting lower cost compared to normal voting on the long run. If it is implemented properly, an e-voting system can eliminate a lot of common avenues of fraud, increase accessibility, speedup the process of collecting results, give more accurate and trusted results, increase convenience for voters, and reduce the cost of the entire elections process specially on the long run.

The process of designing an efficient and satisfyable election system is a critical process that requires special care. Common people usually have less trust in computerized operations because of the major stories about system crashes and hacking threats. As a result, e-voting elections must be more secure and trustworthy. Kohno *et al.* discussed some of e-voting system problems, such as certain vulnerabilities to network threats, the incorrect usage of cryptographic techniques, the escalation of unauthorized privileges, and the poor software development processes [31].

Here, we select an FPGA-based e-voting system to be the basis of our work [32]. The security of the selected e-voting system relies on utilizing the PHE additive property of

FIGURE 3.1: E-voting versus regular voting.

the CRT-based ELGamal scheme [22]. We propose three different scenarios to launch malicious attacks on the system and suggest the suitable countermeasures.

## 3.2 Related Work

E-voting security is one of the most important topics nowadays. The Caltech MIT Voting Technology Project highlighted that the public confidence in any voting system not only depends on the system reliability but also on the security of the system itself [33]. Kumar and Begum introduced an Electronic Voting Machine (EVM) and its variation [34]. They also discussed issues of EVM, Taxonomy, and Bio-metric-based EVM. Additionally, Yumeng *et al.* reviewed the research on the e-voting schemes that aims at achieving a trusted voting system with all its properties and possible challenges.

Fauzia *et al.* described an implementation of a secure yet efficient e-voting system based on the "Fujioka-Okamoto-Ohta" protocol [35]. The proposed implementation includes the automation of an online voting system providing some new features that were not previously offered in the literature. Those features include allowing voters to verify their own votes, introducing simple and easy to use interface, keeping the privacy of voters' choices, and preventing either ineligible voters from voting or eligible voters from voting twice.

Another design of an e-voting machine was introduced by Alam *et al.* [36]. The authors' principal goal of the project was not to design a perfectly efficient device. Instead, the authors' goal was to design a mother component, which could be easily adopted to any recent technology. Their machine also used the voter's ID to recognize valid voters and to prevent multiple votes from the same voter as well.

Talking about FPGA-based e-voting systems, Wollinger *et al.* provided a summary of security issues that might arise in case of performing cryptographic operations on FPGAs [37]. They mainly focused on how to preserve secrets within the FPGA device itself against certain attacks, such as the "readback attack", which aim to read out the FPGA's SRAM contents or its bitstream. The SRAM contains valuable information about the data used. The bitstream, which define the FPGA configuration, may allow an attacker to use reverse-engineering to estimate the actual logic design within it.

FPGA manufacturers, such as Xilinx and Intel FPGA, provided features to prevent those reverse-engineering attacks [38]. For instance, they protected their FPGA chips against "IP core theft" attacks by encrypting the bitstream itself. Then, during the booting phase, the FPGA can launch an internal module that stores the key, which could be used to decrypt the bitstream in order to use it. In this scenario, an invader that has already read the encrypted bitstream would not be able to learn anything. He/She will not also be capable of performing any queries in order to read the decryption key from the FPGA.

On the other hand, Alkabani and Koushanfar presented an alternative method protecting the secrecy of the FPGA's bitstream by leveraging chip-to-chip behavioral variations in order to achieve what is called *active hardware metering* [39]. The method, which they used mainly depends on making the FPGA configuration unique for any given chip. So, the movement of a certain configuration from one chip to another would not cause a correctly functioning implementation.

Oksuzoglu *et al.* presented a minimal design of a secure e-voting system [32]. They realized their implementation using a simple FPGA board from Xilinx. They named their proposed system "VoteBox Nano" as they have already followed the same guidelines firstly illustrated by the original "VoteBox". They only restricted some network features so as to fit on a cheap FPGA. It is a very simple design running without any operating system (OS). It only consists of an FPGA connected to an interface screen (VGA) and a keypad to allow the voter to select his desired candidate and confirm her choice. For the VoteBox Nano, secrecy of the design itself was not the main problem. The main issue was how to detect tampering.

Drimer *et al.* described an algorithm, which permits the FPGA to safely reject any undesirable configuration updates [40]. Dutt *et al.* proposed the idea of "parity groups" [41]. They simply added parity bits to the logic components implemented on the FPGA. By this method, modifying any block of the internal logic blocks will directly cause a parity failure with no corresponding changes elsewhere.

In case of obtaining machine components from third parties, design is exposed to further challenges that need to be faced. That includes hardware spywares and hidden backdoors. Old-fashioned testing and verification methods are not the suitable candidates for detecting the issues, resulting from HTs. Generally, HTs are not designed to be activated at test time as they mainly depend on certain rare conditions to trigger them. In the last decade, various techniques have been proposed to detect HT within FPGA-based designs. Those techniques could be divided into two main parties; architectural techniques [42], and side-channel dependent ones [43, 44]. Those techniques would be discussed in detail in Subsection 4.2.

## 3.3 E-voting System Overview

The implemented e-voting system is similar to VoteBox Nano design [32]. Figure 3.2 shows an overview of the full e-voting system. A voter logins to any of the e-voting boxes, which are distributed over the country. Then, e-voting box encrypts the vote using CEG encryption algorithm, which is an additive PHE technique. Encrypted votes are sent to the main secured server via a network connecting the whole country holding the elections. Each single vote would be represented by a v-tuple of encrypted zeros/ones, where $v$ is the total number of voting states (candidate 1, candidate 2, candidate 3, etc.). The main server will receive the encrypted votes, use the homomorphic property to add them together, and send the final encrypted results to the election supervisor. The decryption methodology, performed by the supervisor, follows the same concept of CEG Algorithm used in encrypting votes. Finally, voting results can be ready on even the same day without human interference.

Figure 3.3 shows an abstraction of our proposed e-voting box. The true random number generator (TRNG) core block is responsible for generating keys, which are required for vote encrypting. TRNG depends mainly on post-processing of digitized noise. Every encrypted value in the system needs a unique random number. We should highlight that choosing a TRNG algorithm is critical as numbers prediction may allow the attacker to decrypt the ciphertexts. It is worth mentioning that the voter's privacy mainly depends on the hardness of predicting the selected random numbers. Input keypad and VGA screen cores represent the input and output modules, respectively. The machine screen will display the names of the candidates with their numbers arranged from 1 to $n$. The voter will use the keypad buttons to select his candidate and confirm his choice. He/She can also control some other features, such as determining the screen brightness.

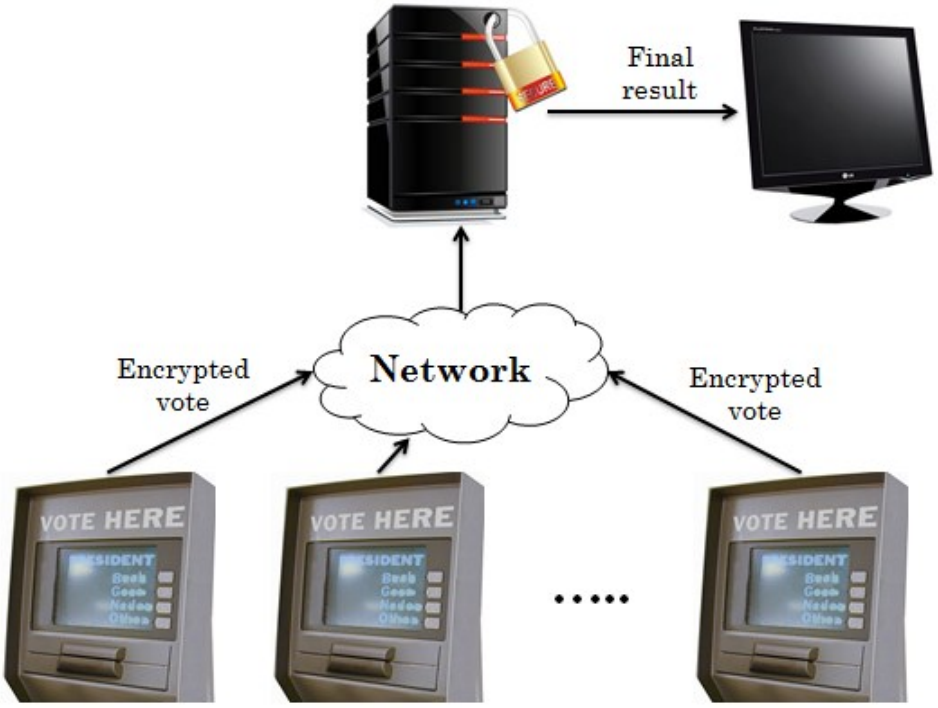



FIGURE 3.2: The e-voting process.

Xilinx MicroBlaze is the main block in the design [45]. Its main block diagram is shown in Figure 3.4. It is a virtual microprocessor, which is built by adding blocks of code, called cores, within a Xilinx FPGA. It presents a 32-bit Harvard RISC architecture with separate 32 bit instruction and data buses. Hence, it supports executing programs and accessing data from both on-chip and external memory at the same time. It is specially

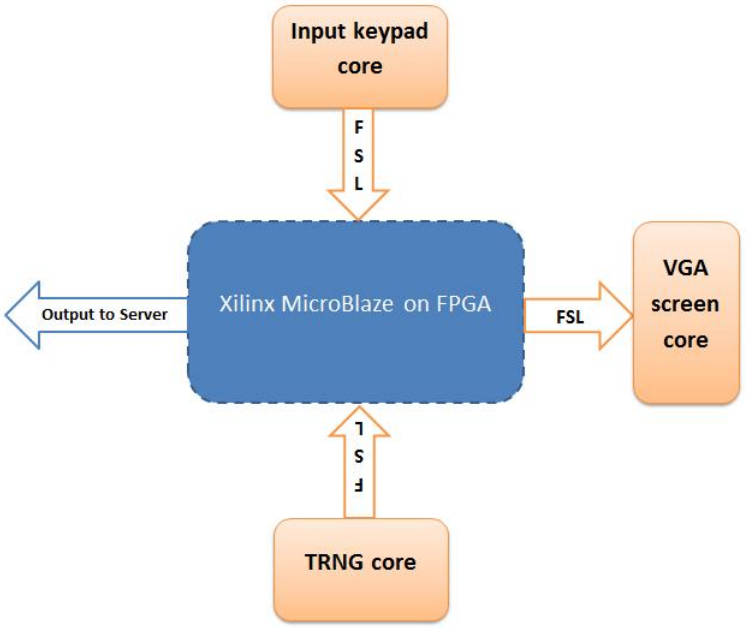



FIGURE 3.3: Abstract view of our proposed e-voting machine.

optimized for Xilinx FPGA boards. The white items, such as the program counter, the special purpose registers, the bus interfaces, and the ALU, represent the backbone of the MicroBlaze architecture. On the other hand, the gray-shaded items represent the MicroBlaze optional features that could be added or removed based on the application needs.

MicroBlaze is connected to other cores using the Fast Simplex Link (FSL). FSL is an interconnect, which supports a point-to-point communication in only one direction. It provides a direct communication between any two IPs on the FPGA when implementing an interface to the FSL interconnect. The MicroBlaze is capable of supporting up to 8 FSLs [45].

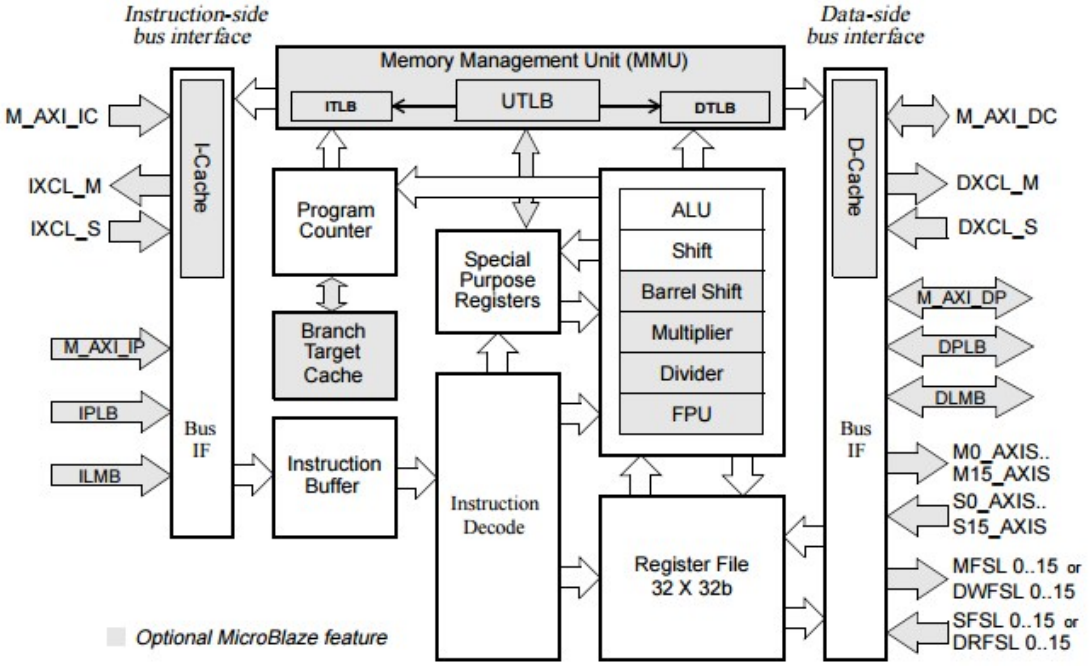



FIGURE 3.4: MicroBlaze block diagram [45].

## 3.4 Scenario for a Possible Attack

An untrusted FPGA-based voting machine may be used to tamper with the legal votes of users. Attacking vendor may inject cores connected to the MicroBlaze via FSL. These cores are responsible for dealing with inputs from the keypad and interfacing with the output screen. The attacker may add a hidden core that replaces the user's vote with another one, if it receives a special external trigger. In our case study, we assume that the voting system contains a secret core connected to the MicroBlaze core and takes its input from the FSL coming from the input keypad core, as shown in Figure 3.5.

We propose a scenario in which an attacker uses the input keypad as follows. He/She will press the push button that control screen brightness with a secret sequence depending on the position of the wanted candidate. That secret sequence would be translated into data sent to the MicroBlaze via the FSL in the unused bits beside the regular data. As a result of triggering the MicroBlaze back-door, all the coming sent votes will support the wanted candidate whatever the voter chooses. Repeating that several times on several machines will affect the whole election results significantly.

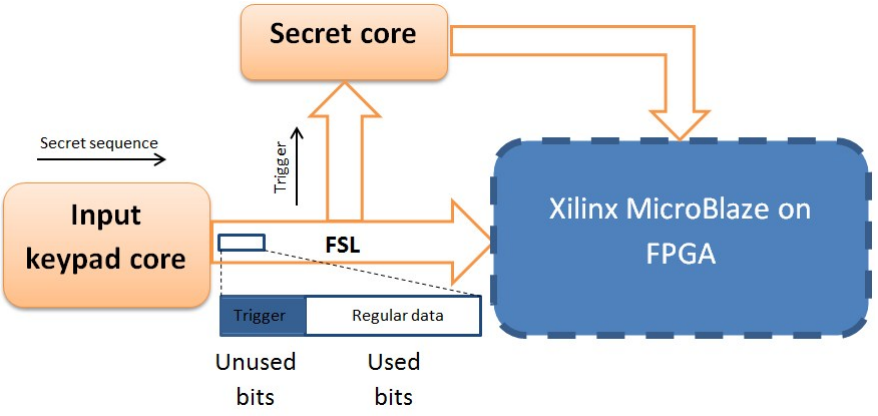



FIGURE 3.5: Block diagram of using a malicious secret core within an untrusted e-voting machine.

## 3.5 Protection Against Proposed Attack

There are several ways to protect our system from untrusted third party IPs. We suggest to use the *Simple Blockage (SB)* method introduced in [46] with a simple improvement. The authors of the original work proposed to obfuscate the output of the IP under suspection before sending any data out of it. Later on, they will undo that obfuscation step at the receiver's input. This idea would help protecting data from leaking and avoiding injected triggering. They introduced using either RC4 or other simple obfuscating function. Here, we choose to protect the design using a simple xoring function, as shown in Figure 3.6 and Figure 3.7. Obfuscation will take place between keypad and MicroBlaze. In our case, the data transmitted via the FSL is 32-bit.

We enhance the *SB* method by resetting any unused bits to zero before receiving them at the MicroBlaze, as shown in Figure 3.8. We only allow the trusted-known used bits to go. Obfuscating the unused bits cost is wasted and will be omitted. Furthermore, an attacker may depend on the unused bits to discover our simple obfuscating function. We did not use the partial reconfiguration feature to change obfuscating function periodically as proposed in [46], as the partial reconfiguration feature doubles FPGA

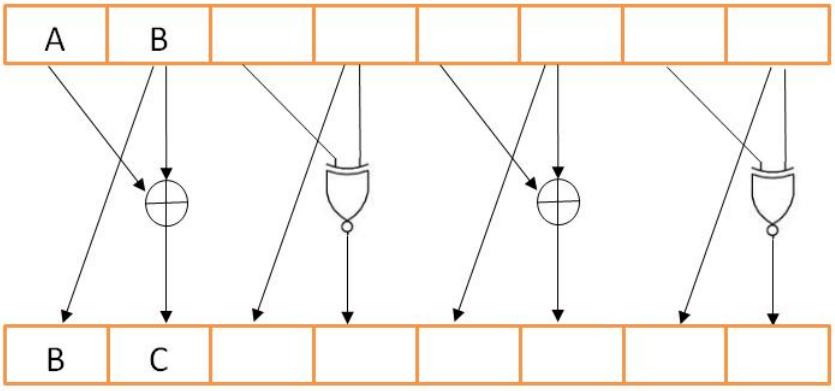



FIGURE 3.6: Simple obfuscating function depending on data xoring.

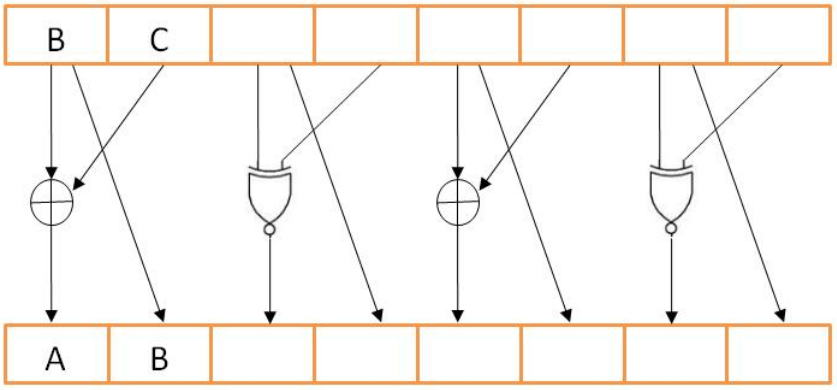



FIGURE 3.7: Inverse obfuscating function in order to retrieve data.

and e-voting box area cost. Subsection 3.7.2 shows function overheads in details. Our technique would prevent the trigger that would turn the secret core on.

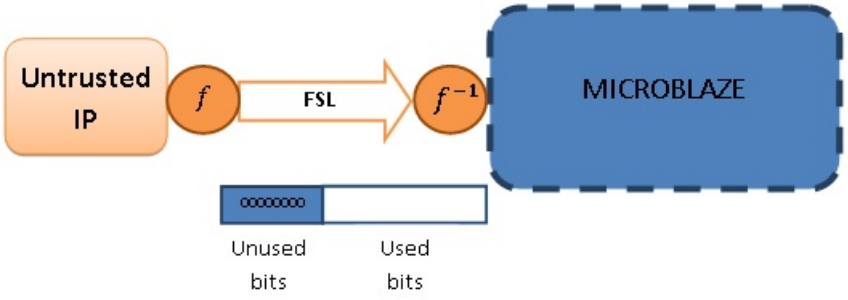



FIGURE 3.8: E-voting protection against proposed attack.

## 3.6 Other Attacks and Countermeasures

In this section, we discuss a couple of other feasible attacks and propose the suitable countermeasures for them.

### 3.6.1 Sequence Cheat Code Attack

In Section 3.4, we assumed that the secret core trigger will be sent to the MicroBlaze core in the unused bits in one data packet. That is called *single-shot cheat code* as described in [47].

Another possible attack may be based on sending a bigger cheat code within a number of successive pieces. Those pieces could be represented by using different cycles/inputs. This attack is called *sequence cheat code* attack. These successive cheat codes can be introduced to the system through the FSL bus connecting the input keypad core and the MicroBlaze.

Waksman and Sethumadhavan proposed two different ways to solve cheat code issues [47]. One for the single-shot cheat code using data obfuscation (i.e., encrypting the suspected IPs input values in order to eliminate any malicious codes). Another way is to use sequence-breaking against sequence cheat code. The sequence breaking method suggests to pseudo-randomly scramble the order of inputs, which enters the suspected IPs. By this way, we can prevent those untrusted units from recognizing the malicious sequences of inputs, which can activate an HT.

However, our proposed method in Section 3.5 is capable of protecting the design from that external trigger either if it is single-shot cheat code or even sequence cheat code. So, we do not have a need for extra hardware for handling sequence cheat code, and that is an advantage of our proposal over Waksman methodology.

### 3.6.2 Used Bits Attack

Now, let us introduce another attack where we suggest that keypad core is infected with no extra hidden cores. So, the trigger will be sent in the used bits. In all previous attacks, we assumed that the special trigger is sent in the unused bits of the data packet. The main risk is that the MicroBlaze sees that confidential data (user's vote) in unencrypted form and thus can manipulate it. Additionally, this core, along with many other hardware cores, are usually obtained as a third party IP. In this case, using our technique to secure data transfer would not prevent triggering the hidden back-door because the trigger will be obfuscated at the output of the untrusted IP (input keypad core), transferred via the FSL, and then return back to its original form at the input of the MicroBlaze core.

We should mention that Waksman and Sethumadhavan presented a solution for this attack by using data obfuscation for computational units [47]. The selection of the

correct method for obfuscation mainly depends on the IP type. The authors suggested classifying the IPs into two main classes; computational IPs and non-computational ones. For the later class, the IPs do not actually perform any computations on the data. As a result, the authors simply used an encryption method to obfuscate the data values before entering the non-computational IP.

On the other hand, for computational IPs (as in our proposed attack here), data encryption is much more complicated than the non-computational case. In certain scenarios, duplicating the whole design might be more efficient than using obfuscation. This high complexity will appear in our case because our computational unit uses ElGamal Algorithm for public key encryption [21]. They suggested using the third party computational IP without giving it the advantage of recognizing the data. They depend mainly on HE schemes, as shown in (3.1). But, the main problem of this solution will be the overhead cost of all e-voting boxes.

$$Gamal(xy) = Gamal(x)Gamal(y) \tag{3.1}$$

In case one wants to encrypt a certain data value $x$, where $x$ represents the vote in our case, using ElGamal Algorithm on a special purpose encryption core, the following steps should be followed.

1. Use a TRNG module in order to generate a random value $y$ and calculate its encryption result $Gamal(y)$
2. Use a trusted, regular ALU to compute $z = xy$, where $x$ is the user's vote to be encrypted.
3. Send $z$ to the encryption core that should returns $Gamal(z) = Gamal(xy)$, which will be sent to the main server.
4. Add the received encrypted votes $Gamal(z)$ at the server side.
5. Decrypt the summation result and use a trusted ALU on the server side to divide the result by $y$.

We use the untrusted cryptographic unit within the e-voting box to encrypt the vote, $x$, without allowing this untrusted component of knowing the actual vote value. This will protect from the triggering code injected in the real data.

## 3.7 Evaluation

Here, we will describe the experimental setup, which we used for creating the e-voting system. Furthermore, the numerical results for the area and power overheads are mentioned in detail.

### 3.7.1 Experimental Setup

The total experiment was done by using **Spartan 3E starter kit** [48]. The needed logic is implemented using Verilog. Figure 3.9 illustrates the experiment setup. Our FPGA board is connected to a VGA screen via a VGA cable. The input keypad is connected to the FPGA using a PS2 interface.

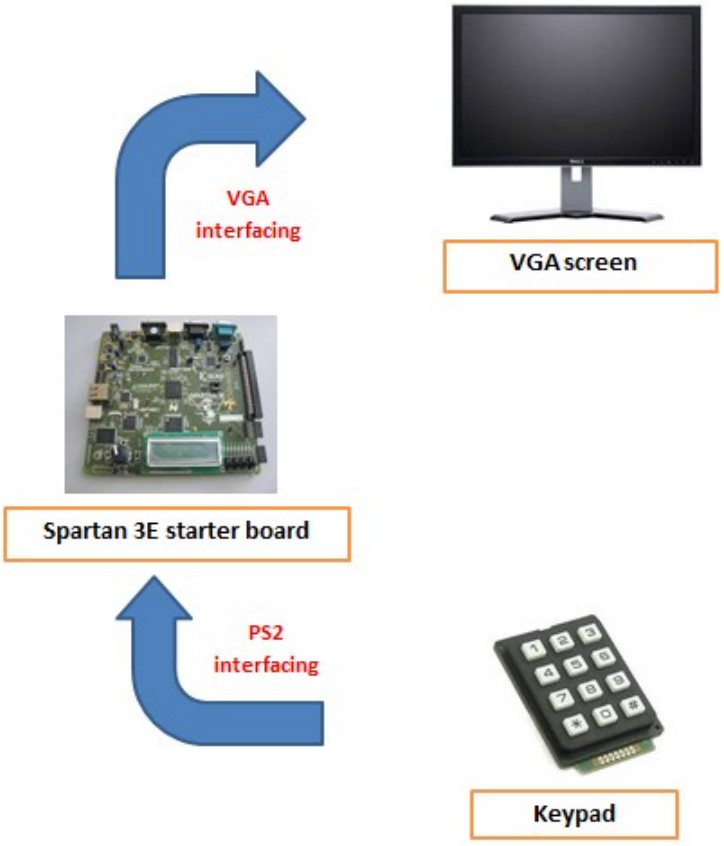



FIGURE 3.9: Experimental setup for the e-voting system.

Figure 3.10 represents the used FPGA kit. It mainly includes a Xilinx XC3S500E Spartan-3E FPGA (FG 320 package, with a $-4$ as a speed grade), 232 user Input/Output pins, Over 10,000 logic cells, 16 Mbits of SPI serial Flash, SPI serial Flash configuration, DDR memory interfaces, 64 MByte of DDR SDRAM, PS/2 mouse or keyboard

port, VGA display port, 50 MHz clock oscillator, and a MicroBlaze 32-bit embedded RISC processor.

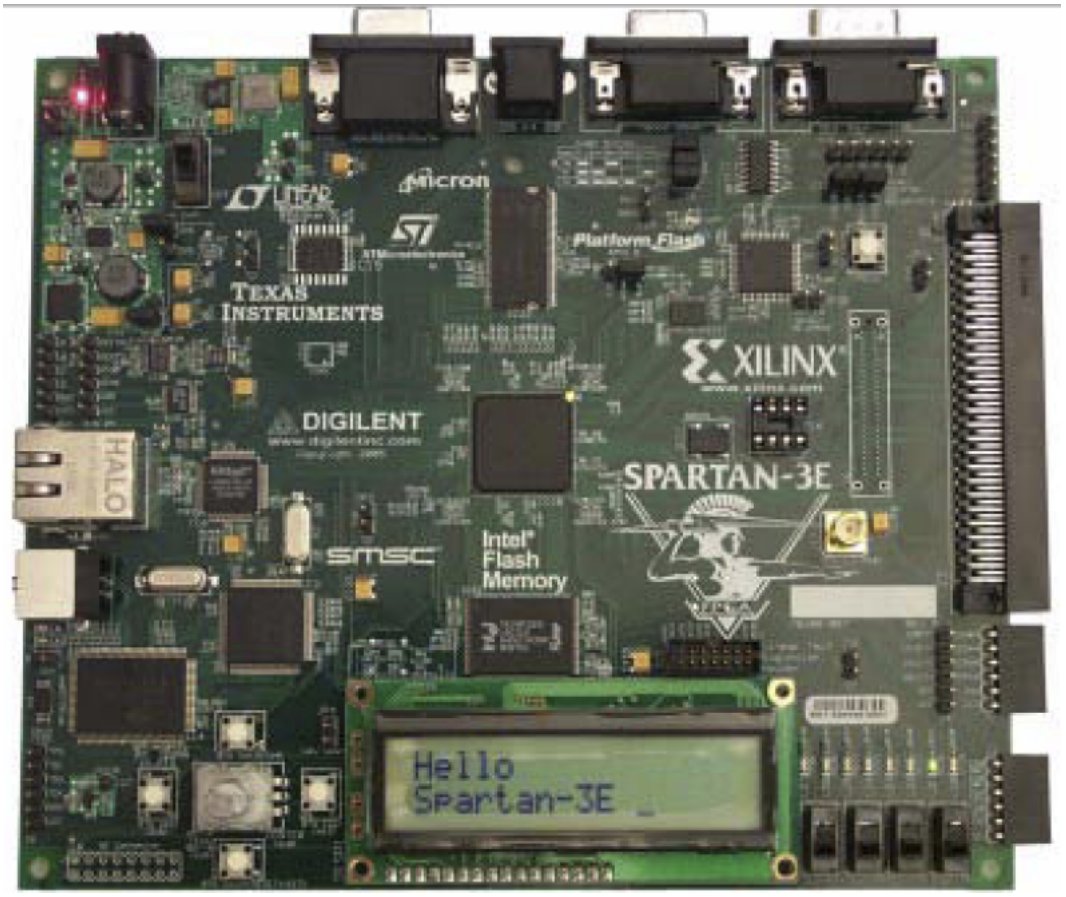



FIGURE 3.10: The Xilinx Spartan 3E starter kit overview [48].

### 3.7.2 E-voting Protection Results

Here, we evaluate the effect of using the protection technique, introduced in Section 3.5, on device resources, power, and time delays. Each time we make a comparison between results coming from the protected design and the original untrusted design. Furthermore, we analyze the overhead of resetting only the used bits in the first experiment. Then, we examine using the simple obfuscation function.

#### 3.7.2.1 Resetting Unused Bits

Table 3.1 presents device utilization summary for the untrusted system and the protected one. The first column represents logic resources. The second and third columns represent the original system data and the protected one, respectively. The last column shows the protection overhead.

TABLE 3.1: Device utilization for untrusted and protected systems (with and without resetting unused bits) showing overhead percentage.

| Logic resources | Untrusted system Without resetting unused bits | Protected system With resetting unused bits | Overhead (%) |
|---|---|---|---|
| No. of used slice flip flops | 3,401 | 3,428 | 0.79 |
| No. of used 4-input LUTs | 4,266 | 4,396 | 3.05 |
| Total no. of used 4-input LUTs | 4,391 | 4,521 | 2.96 |

Power is also calculated using Xilinx Power Analyzer (XPA) [49] at 50 MHz clock. Table 3.2 shows the power comparison between the untrusted system and the protected one. It was found that dynamic power, which represents the power dissipated in clocks, logic, signals, DSPs, and IOs, slightly increased after inserting the protection method while the leakage power remains constant.

TABLE 3.2: Power comparison between original and protected systems (with and without resetting unused bits).

| Logic resources | Power consumption (W) | |
|---|---|---|
| | Untrusted system Without resetting unused bits | Protected system With resetting unused bits |
| Logic | 0.009 | 0.009 |
| Signals | 0.007 | 0.008 |
| BRAMs | 0.006 | 0.006 |
| MULTs | 0.001 | 0.001 |
| DCMs | 0.041 | 0.043 |
| IOs | 0.340 | 0.340 |
| Leakage | 0.094 | 0.094 |
| Total | 0.498 | 0.501 |

From the timing prospective, we used the Post-PAR Static timing report generated from Xilinx Platform Studio (XPS) v14.6 to get the design statistics. For the untrusted design, the minimum period is **16.757 ns** (max frequency: **59.677 MHz**). The maximum net delay is **2.059 ns**. For the protected design, the minimum period is **15.846 ns** (max frequency: **63.107 MHz**). The maximum net delay is **2.265 ns**. So, delay overhead is **0.206 ns**, which is below 10%.

#### 3.7.2.2 Using Enhanced SB Method

Table 3.3 presents device utilization summary for the untrusted system and the protected one, when using the enhanced simple obfuscation method which is described in Section 3.5. The first column represents logic utilization. The second and third columns represent the original system data and the protected one, respectively. The last column shows the protection overhead.

TABLE 3.3: Device utilization for untrusted and protected systems (with and without enhanced Simple Blockage) showing overhead percentage.

| Logic resources | Untrusted system Without enhanced SB | Protected system With enhanced SB | Overhead (%) |
|---|---|---|---|
| No. of used slice flip flops | 3,401 | 3,436 | 1.03 |
| No. of used 4-input LUTs | 4,266 | 4,437 | 4.00 |
| Total no. of used 4-input LUTs | 4,391 | 4,562 | 3.89 |

The power is calculated using XPA at 50 MHz clock. Table 3.4 shows the power comparison between the untrusted system and the protected one. It was found that dynamic power increased after inserting the protection method while the leakage power remains constant.

TABLE 3.4: Power comparison between original and protected systems (with and without enhanced Simple Blockage).

| Logic resources | Power consumption (W) | |
|---|---|---|
| | Untrusted system Without enhanced SB | Protected system With enhanced SB |
| Logic | 0.009 | 0.009 |
| Signals | 0.007 | 0.007 |
| BRAMs | 0.006 | 0.006 |
| MULTs | 0.001 | 0.001 |
| DCMs | 0.041 | 0.043 |
| IOs | 0.340 | 0.340 |
| Leakage | 0.094 | 0.094 |
| Total | 0.498 | 0.500 |

From the timing prospective, we use the Post-PAR Static Timing Report generated from XPS to get the design statistics. For the untrusted design, the minimum period is **16.757 ns** (max frequency: **59.677 MHz**). The maximum net delay is **2.059 ns**. For the protected design, the minimum period is **19.737 ns** (max frequency: **50.666 MHz**). The maximum net delay is **2.264 ns**. So, delay overhead is **0.205 ns**.

## 3.8 Conclusion

In this chapter, we highlighted e-voting challenges. In addition to that, we implemented an e-voting system using a VGA screen and a Xilinx FPGA board. After that, an HT was injected in our e-voting FPGA in order to manipulate the final results. We showed the role of HE in securing our design via the usage of ElGamal cryptosystem. Our proposed attack mainly relies on targeting the unused portion of the message bits moved between the input IP and the MicroBlaze main core. We suggested two different mechanisms to handle this problem. We showed that our proposed solution has minimum side effects on the circuit power and timing delays. The reported power consumption overhead was very low and the overhead on the timing delay overhead was below 10%. Device resources overheads did not exceed 4%.

The work discussed in this chapter was published in the tenth International Symposium on Frontiers of Information Systems and Network Applications (FINA 2014), held in conjunction with the 28th IEEE International Conference on Advanced Information Networking and Applications (AINA-2014) in Victoria, BC, Canada, 2014 [50].

# Chapter 4

# Homomorphic Data Isolation for Protection against Hardware Trojans

In this chapter, we are going to utilize the concepts of homomorphism in the domain of securing FPGA-based designs, specially for protection against HTs. During the last decade, the number of FPGA-based designs has been dramatically increased. That greatly affects the FPGA revenues, which exceeded the 4 Billion Dollars level in the last few years. That is mainly because of the high flexibility offered by FPGAs compared to other hardware-based platforms, like ASICs, and its high efficiency compared to software-based solutions, like general purpose processors. For foregoing reasons, FPGA-based designs' security has become a critical issue that needs to be highly taken into consideration.

In order to protect FPGA-based design against HTs, we select ElGamal homomorphic scheme as a basis for our work and implement two PHE schemes based on it. The first scheme is a multiplicative homomorphic one, whereas the second one is an additive homomorphic one. In order to evaluate our proposed designs, we implement all our architectures on a low-cost Spartan-6 FPGA board from Xilinx. Then, we report the logic area utilization, timing delay, and the total consumed power. Furthermore, we present a dual-circuit design, which combines our two introduced designs; the multiplicative and additive one. We utilize the idea of resource sharing to reduce the area overhead as much as possible.

The remaining parts of this chapter are organized as follows. The motivation is described in Section 4.1. Related work is provided in Section 4.2. HT protection using PHE is introduced in Section 4.3. The proposed methods and implementations are illustrated in Section 4.4. Then, Section 4.5 provides the experimental results and presents

the actual overheads of the suggested architectures. Finally, Section 4.6 concludes the chapter.

## 4.1 Motivation

Increasing the complexity of systems proclaims the outsourced manufacturing concept nowadays. As a result, a considerable number of trust issues, within the design industry, has been raised. Anyone with access to any step of the manufacturing process could alter the final product to inject an HT. As defined in Chapter 2, HT is simply a malicious alteration of one's own hardware. This alternation may, under specific rare circumstances, result in information leakage out of the system or functional changes of the system itself. Those kinds of malicious circuitry can be injected by either third party IP owners or fabrication facilities.

There is a critical need to generate novel methods, which are capable of enhancing the trust measures between the designers, fabrication facilities, and end-users. From their own point of view, end-users require more guarantees from fabrication facilities. They need their own products to be stable in the working environment, not leaking any private information in future uses, and of course not being controlled by unknown entities. Additionally, the widespread of the design/fabrication process among several parties has a direct effect on increasing the difficulty of HT detection and protection. We believe that HE is one possible way to resolve the foregoing issue and provide a feasible solution for defeating HTs.

In this chapter, we are discussing new ideas to have a blind data processing by the third party IP with a minimum cost. We achieve our goals by implementing ElGamal encryption cryptosystem, which is considered a multiplicative PHE technique and the CRT-based ElGamal (CEG) encryption cryptosystem, which is considered an additive PHE technique, on a low-cost FPGA and showing the resource utilization, performance, and power analysis of both schemes. In addition to that, we introduce a dual-circuit design that supports both, multiplicative and additive homomorphic properties and provide the obtained savings on area and power over a regular design that has no resource sharing.

## 4.2 Related Work

As our work is mainly directed towards increasing the protection of FPGA-based designs against HT threats, we first survey the HT detection techniques and the proposed

solutions to recover from them. In general, HT detection techniques could be grouped within two major classes; side-channel dependent techniques, and architectural techniques [42]. Side-channel dependent techniques mainly work on detecting the existence of any HT logic in a certain architecture by finding the HT circuit overload on various architecture parameters. For instance, the overall delay with respect to a golden circuit or the total power consumption compared to a non-infected one are two possible examples of these parameters. So, the final target of such techniques is to localize the impact of the HT on the architecture without waiting for them to be active. For instance, Jin *et al.* recorded path delay information from various trusted chips and used the collected data as a reference for detecting the existence of HTs in untrusted chips [43]. As such measurements differ from chip to another, even if all chips are trusted, Yoshimizu suggested using the idea of symmetric path delays to detect HTs [51]. His proposal relied on the fact that the timing delay differences between a pool of symmetric paths would remain unchanged only if no HT exists. In addition to that, Rad *et al.* investigated the effect of an HT on the transient current of an IC power supply via the usage of statistical techniques [52].

On the other hand, architectural techniques aim to enhance the probability of activating HTs during testing phases. Salmani *et. al* inserted dummy flip-flops in the design in order to increase a Trojan activity [53]. The authors used a transition probability threshold to help them choosing the suitable locations for the flip-flops insertion in order not to badly affect the total logic area. In the context of gate level HT detection, Rajendran *et al.* suggested a new technique for protecting the architecture, of both FPGAs and ASICs, via the usage of ring oscillators (ROs) [54]. The authors added more logic gates to the architecture in order to convert architecture paths to ROs. As a result, they could detect the presence of HTs by monitoring any changes that may happen in the frequency of the ROs.

In addition to that, Al-Anwar *et al.* developed a new methodology to protect the designs versus a hardware Spyware [55]. In their work, the authors introduced multiplexing between multiple variants implementations, so they decreased the probability of leaking sensitive information. Then, they proposed detecting the infected third party FPGA implementations by using cyclic redundancy check (CRC). In another work, Al-Anwar *et al.* enhanced the previous methodology by using partial reconfiguration technology to remove an infected IP [56, 57]. Furthermore, they introduced voting between the output of an odd number of similar third party IPs from different manufacturers in order to get a safe output.

It is obvious that all the foregoing HT detection methods need a golden, i.e., non-infected, chip to operate. That requirement represents a real problem as it is feasible only if the design is completely done by only one system designer and does not utilize any component from a third party vendor [58]. Those methods will become completely useless if the designer decides to make use of any IP from a third party. In order to overcome this fatal issue, Baumgarten *et al.* suggested utilizing configurable logic barriers inside the architecture [59], preventing the activation of any HT added during the IC manufacturing stages. Additionally, Zhang and Tehranipoor suggested using ATPG methods, code coverage analysis, and formal verification to ensure if the circuit is Trojan-altered or not [60]. They believed their proposed methods could be a suitable way to eliminate the need for a non-infected architecture as a reference. Additionally, Moein *et al.* have recently proposed a detailed comparison between different HTs' attributes in order to enhance the detection methods [61, 62]. In another work, Moein *et al.* also suggested an approach to implement different HT attacks and countermeasures [63].

In a different context, Konstantinou *et al.* offered an FHE-based solution for trust issues, which usually occur between IP users and owners, respectively [64]. The authors mainly targeted the functional verification part of the IP securing phases. Their suggestion includes creating a functionally similar version of the original IP design in the encrypted domain. Moreover, they used FHE-encrypted test vectors to verify the functionality of the targeted IP. Thus, the design itself and the input vectors are privacy-preserving even if they were outsourced to a third party. Unfortunately, their experiments showed a slowdown that reached a three orders of magnitude. That was expected due to the high overheads that come with the usage of the FHE techniques, which we tried to overcome by depending on PHE only in this work.

## 4.3 HT Protection using PHE Overview

Consider the case where a third party IP is needed to carry out some operation on data $A$ and will produce output data $B$. Figure 4.1 shows the ideal world, where the third party IP does not have any access to the real data as it is homomorphically encrypted. This will give us the capability to carry out the required operation by the third party IP without revealing the original data. Thus, we can retrieve the result $B$ after the decryption process.

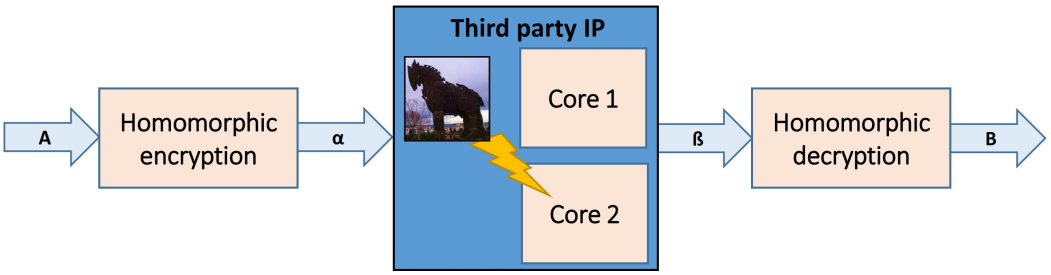



FIGURE 4.1: Homomorphic encryption to protect from Hardware Trojans.

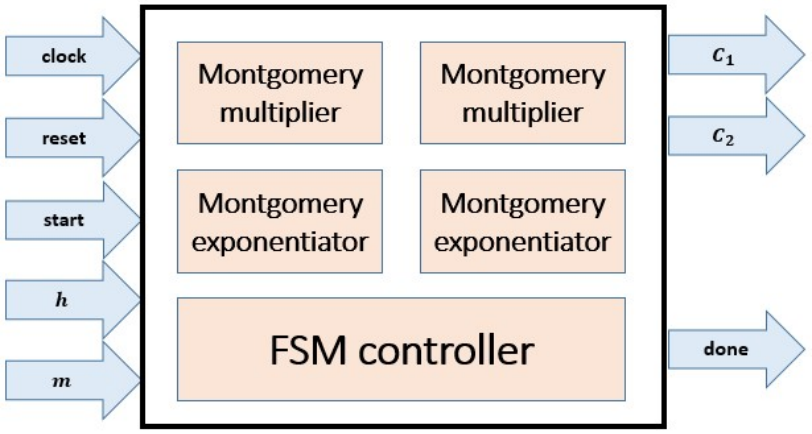
## 4.4 HT Protection using PHE Methods

Here, we introduce our suggested methods for defeating HT in third party IPs. First, we propose two schemes that support PHE for the third party IP, which performs one type of operation (multiplication only or addition only). Then, we combine the two methods in a dual-circuit design that supports both multiplication and addition to satisfy applications that utilize the two operations.

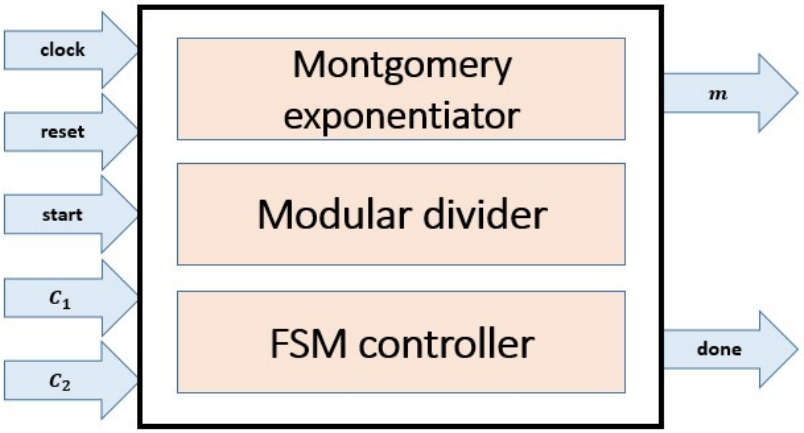
### 4.4.1 Sufficient PHE Support

Upon classifying the IPs based on processing type, one concludes that there is no need to afford the high cost of FHE if the third party IP does only one type of operation. It is totally sufficient to have PHE encryption/decryption before/after the suspected IP. In other words, if the suspected IP is used in an e-voting system and only does addition operation to count votes on the server side, it is enough to support one of the additively homomorphic schemes mentioned before in Subsection 2.2 [50]. For non computational suspected IPs, it is adequate to do simple obfuscation functions before the suspected IP and do the inverse of that function afterwords.

Here, we discuss two hardware implementations that support PHE using ElGamal homomorphic encryption technique, described in Subsection 2.2.1. The first implementation is the main ElGamal encryption/decryption scheme [21], which is a multiplicative homomorphic scheme, whereas the second one is the CEG scheme [22], which is an additive homomorphic scheme.




(A) Encryption.




(B) Decryption.

FIGURE 4.2: Block diagram for ElGamal encryption/decryption scheme.

#### 4.4.1.1 ElGamal Scheme Implementation

Figure 4.2 shows the block diagram of our implementation of ElGamal encryption/decryption scheme. The encryption module consists of two Montgomery modular multipliers, two Montgomery modular exponentiators, and a finite state machine (FSM) controller that is responsible for synchronizing other components' inputs and outputs to perform the encryption operations defined in (2.3). The decryption module consists of one Montgomery modular exponentiator, one modular divider, and an FSM controller that is also responsible for synchronizing other components' inputs and outputs to perform the decryption operations defined in (2.4). Both modules use a clock and reset signals as inputs. Reset and done signals are utilized to define the start and the end of module operations, respectively. The message $m$, ciphertexts $C_1$ *and* $C_2$, and the public key $h$ are all $k$ bits vectors, where $k$ is a user-defined integer.

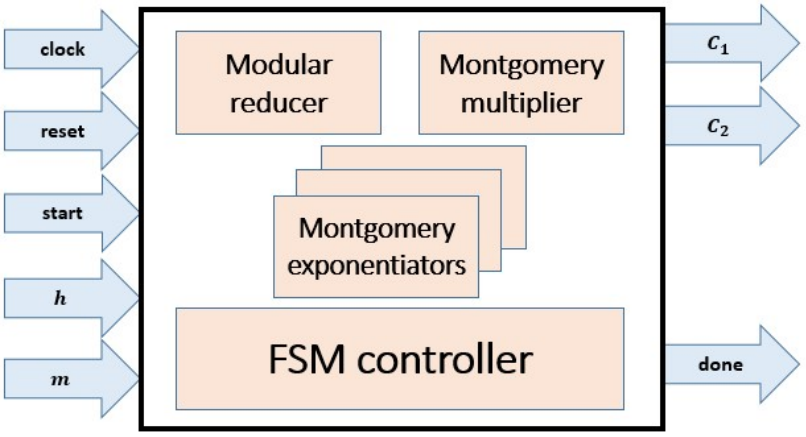
Montgomery multipliers are used in the design as the Montgomery's algorithm is considered one of the most efficient and widely used algorithms for modular multiplication [65]. Besides Montgomery, there exists a couple of modular multiplications methods, such as the "double, add, and reduce" method and the "multiply and reduce" one. However, those techniques have more complex computations than the Montgomery's algorithm [66]. The main operations included within the binary Montgomery multiplier are addition, subtraction, and shift operation. Shift is used instead of the division operation, which is a time-consuming operation in conventional modular multiplication. Actually, the Montgomery multiplier computes $Z = X \times Y \times R^{-1} mod\ M$ instead of $Z = X \times Y mod\ M$, where $R$ is a chosen integer that should be a power of two and relatively prime to $M$. So, in this case, the operands of the process need to be converted into Montgomery's domain before this multiplier is used and out of the Montgomery's domain after using the multiplier.

Generally speaking, the modular exponentiation operation is done by performing successive number of modular multiplication operations. For our modular exponentiators, the LSB-first algorithm using Montgomery multiplication is used. This algorithm computes $Z = Y^X mod\ M$ in $k$ executions of a loop that, in turn, includes at most two Montgomery multiplication operations, which are executed concurrently. That improves the performance of the module [66].

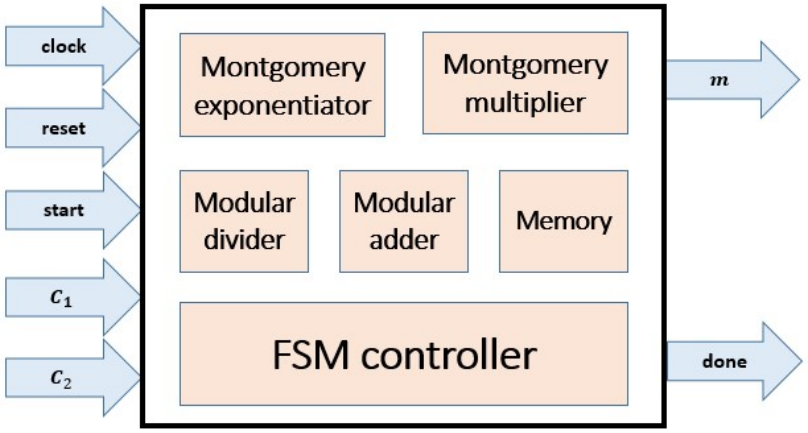
The decryption part of the scheme includes the usage of a modular divider module. We implemented the plus-minus algorithm as it gives the shortest computation time with a cost-effective area [67]. The key generation module consists mainly of a Montgomery exponentiation circuit and a TRNG module, which is not in the scope of this chapter. Finally, we should mention that the usage of only one multiplier and one exponentiator is enough to achieve the desired encryption results, but that results in a high critical path delay.

#### 4.4.1.2 CEG Scheme Implementation

Figure 4.3 shows the block diagram for our implementation of the CEG encryption/decryption scheme. This design is quietly different from ElGamal design discussed before as the encryption operations defined in (2.5) requires the usage of multiple Montgomery exponentiators. As the timing delay needed by one exponentiator is more than the delay of a single multiplier, the FSM controller is modified to utilize only one Montgomery multiplier. A modular reducer circuit is used to handle the operation of reducing $m$ into several $m_i$ based on the relation of $m_i = m(mod\, d_i)$ for $i = 1, \ldots, t$.




(A) Encryption.




(B) Decryption.

FIGURE 4.3: Block diagram for the CRT-based ElGamal (CEG) encryption/decryption scheme.

For the decryption module, it consists of one Montgomery modular exponentiator, one Montgomery modular multiplier, one modular divider, one modular adder, FSM controller, and a single block of memory used to facilitate the implementation of the inverse CRT needed in (2.6) [22]. Input and output vectors are now $k \times t$ *bits* instead of $k$ *bits*, where $t$ is the number of ciphertext pairs.

### 4.4.2 Dual-Circuit Design

The main motivation for this design is that some third party IPs require the usage of more than one single type of operation. For instance, an IP may need to perform both addition and multiplication but not at the same time. One can imagine the functionality of that IP as a simple ALU that uses a selection line to switch its mode between two different operations. In this case, using one type of PHE schemes would not be sufficient. We have to implement two different schemes, such as implementing the two

schemes described above, in order to prevent the attacker from revealing the ALU input and output data. We suggest a solution for this issue by combining the two previously-proposed schemes, ElGamal and the CEG, in a single dual-circuit design. Thus, the proposed design supports both additive and multiplicative homomorphism. Moreover, another possible example for an application that needs the availability of both homomorphic operations is when we have two unique IPs in a design and the first IP performs addition while the second IP performs multiplication. Assuming that both IPs will not work on the same time, one can instantiate only one instance of our dual-circuit module and control its functionality to perform the needed operation of any of the two IPs, when needed, with the minimal cost in area and power consumption.

Furthermore, we try to share resources as much as we can between the two schemes in order to have minimal design cost. For example, computing $C_1$ in (2.3) and (2.5) needs an exponentiation operation. The same situation occurs when computing $C_2$ as we need an exponentiation operation followed by a multiplication operation. The only difference is that the modified versions in (2.5) reuse their modules many times based on the value of $t$. Thus, we use the duality concept that enables us of sharing as much resources between the two circuits in order to reduce the design area. As the CEG scheme uses the same basic blocks of ElGamal scheme with some additional blocks, we depend on the same architecture shown in Figure 4.3 and add a *select* signal that chooses between the multiplicative homomorphic and the additive homomorphic circuits. The FSM controller is modified to be able of handling the two cases with the same building modules. The case is the same for the key generation and decryption modules. By using this simple idea, we manage to decrease the area cost a lot and allow for the two homomorphic properties to be available on a single module. That completely solves the issue of the third party IP, which needs to perform both addition and multiplication operation.

## 4.5 Evaluation

This section evaluates the performance of our proposed methods, described in Section 4.4, in terms of resource utilization, delay, and power consumption.

### 4.5.1 Experimental Setup

The proposed methods are implemented on Xilinx Spartan-6 XC6SLX75 with FGG484 package and -2 speed grade [48]. We use Xilinx Integrated Synthesis Environment (ISE)

14.6 tool to obtain the logic and performance results after placement and routing [68]. The power is calculated using Xilinx Power Analyzer (XPA) [49] at 100 MHz clock.

### 4.5.2 PHE Methods Results

Table 4.1 shows the resource utilization of our two PHE schemes, ElGamal and CEG, using vectors of size equals 8 bits.

TABLE 4.1: Resource utilization of ElGamal and CRT-based ElGamal (CEG) encryption/decryption schemes for k = 8 bits.

| Logic resources | Encryption | | Decryption | |
|---|---|---|---|---|
| | ElGamal | CEG | ElGamal | CEG |
| Number of Registers | 295 | 614 | 207 | 364 |
| Number of LUTs | 420 | 715 | 259 | 442 |
| Number of BRAMs | 0 | 0 | 0 | 1 |

Table 4.2 shows the maximum operating frequency of the two proposed PHE schemes along with the needed number of cycles to finish their work.

TABLE 4.2: Timing performance of ElGamal and CRT-based ElGamal (CEG) encryption/decryption schemes for k = 8 bits.

| | Encryption | | Decryption | |
|---|---|---|---|---|
| | ElGamal | CEG | ElGamal | CEG |
| Frequency (MHz) | 161.277 | 164.352 | 123.870 | 121.862 |
| No. of cycles | 171 | 480 | 153 | 512 |

From power prospective, Table 4.3 shows the power analysis for ElGamal encryption/decryption scheme and the CRT-based one. It was found that the dynamic power slightly decreased in case of encryption and increased in case of decryption due to the usage of the memory component and its logic controller in decryption. The leakage power remains constant in the both cases.

### 4.5.3 Dual-Circuit Design Results

Here, we compare the results of our proposed dual-circuit design to using regular two IPs, one for ElGamal and another for CEG design without any resource sharing between them. We want to assess the gain of our resource sharing. In order to differentiate

TABLE 4.3: Power consumption (mW) of ElGamal and CRT-based ElGamal (CEG) encryption/decryption schemes for k = 8 bits.

| Logic resources | Encryption | | Decryption | |
|---|---|---|---|---|
| | ElGamal | CEG | ElGamal | CEG |
| Logic | 3.84 | 5.47 | 2.70 | 3.69 |
| Signals | 2.82 | 4.69 | 2.01 | 3.23 |
| BRAMs | 0.00 | 0.00 | 0.00 | 0.74 |
| IOs | 16.51 | 8.99 | 5.23 | 2.74 |
| Clocks | 5.65 | 7.87 | 4.21 | 5.87 |
| Leakage | 65.00 | 65.00 | 64.00 | 64.00 |
| Total | 93.82 | 92.02 | 78.15 | 80.27 |

between the two designs, we call the first design, *Dual ElGamal*, while the second design is called *Regular ElGamal*.

Firstly, Table 4.4 shows the area reduction that results from using our *Dual ElGamal* design over *Regular ElGamal* design. The area reduction column is calculated using (4.1). It is clear that the idea of dual-circuit design has greatly improved the usage of hardware resources.

$$Reduction\ space(\%) = \frac{Regular\ area - Dual\ area}{Regular\ area} \times 100. \tag{4.1}$$

TABLE 4.4: Area reduction of our dual ElGamal design over the regular ElGamal design for k = 8 bits.

| Logic resources | Encryption | | | Decryption | | |
|---|---|---|---|---|---|---|
| | Regular ElGamal | Dual ElGamal | Area reduction (%) | Regular ElGamal | Dual ElGamal | Area reduction (%) |
| Registers | 909 | 635 | 30.14 | 536 | 364 | 32.09 |
| LUTs | 1137 | 735 | 35.36 | 626 | 457 | 26.99 |
| BRAMs | 0 | 0 | 00.00 | 1 | 1 | 00.00 |

Table 4.5 gives the maximum operating frequency of our *Dual ElGamal* design and the *Regular ElGamal* design using vectors of size $k = 8\ bits$. The number of cycles here represents the clock cycles needed to perform one multiplicative homomorphic

TABLE 4.5: Timing comparisons between our dual ElGamal design and the regular ElGamal design for k = 8 bits.

| | Encryption | | Decryption | |
|---|---|---|---|---|
| | Regular ElGamal | Dual ElGamal | Regular ElGamal | Dual ElGamal |
| Frequency (MHz) | 161.277 | 158.51 | 117.099 | 121.344 |
| No. of cycles | 651 | 662 | 665 | 665 |

TABLE 4.6: Power consumption (mW) of our dual ElGamal design and the regular ElGamal design for k = 8 bits.

| Logic resources | Encryption | | Decryption | |
|---|---|---|---|---|
| | Regular ElGamal | Dual ElGamal | Regular ElGamal | Dual ElGamal |
| Logic | 9.25 | 6.29 | 5.91 | 3.82 |
| Signals | 8.14 | 6.02 | 5.67 | 3.49 |
| BRAMs | 0.00 | 0.00 | 0.74 | 0.74 |
| IOs | 25.27 | 10.83 | 5.67 | 3.61 |
| Clocks | 11.78 | 6.89 | 8.78 | 4.86 |
| Leakage | 65.00 | 65.00 | 65.00 | 64.00 |
| Total | 119.44 | 95.03 | 91.77 | 80.52 |

operation followed by one additive homomorphic operation. The needed number of cycles to get the final output is the same in both designs, except that the encryption part of our dual designs utilizes more clock cycles. That is due to the usage of only one Montgomery multiplier instead of two, as illustrated in Section 4.4.

From power prospective, Table 4.6 shows the power analysis for our *Dual ElGamal* design and the *Regular ElGamal* design. The usage of the duality idea results in an obvious improvement in total power consumption as it eliminates the power consumed by the duplicated modules. The savings in power consumption are 20.44% for encryption and 12.26% for decryption.

## 4.6 Conclusion

In this chapter, we highlighted the importance of homomorphic encryption in defeating HTs in third party IPs. As PHE is sufficient enough with some third party IPs, we implemented two designs that supports PHE (multiplicative only and additive only) based on ElGamal encryption/decryption scheme.

Furthermore, we integrated the two designs together and introduced a dual-circuit design that achieved a great improvement in area and power over a regular design that combines two IPs, one for ElGamal and another for CEG, without any resource sharing between them. Our architectures were implemented on a low-cost Xilinx Spartan-6 FPGA and area, delay, and power results were reported. The area reduction reached 30% and savings in power consumption were 20% for encryption and 12% for decryption.

The work discussed in this chapter was published in the IEEE Computer Society Annual Symposium on VLSI (ISVLSI 2015) at Montpellier, France [30].

# Chapter 5

# *CryptoImg*: Privacy Preserving Processing over Encrypted Images

Cloud computing services provide a scalable solution for the storage and processing of images and multimedia files. However, concerns about privacy risks prevent users from sharing their personal images with third-party services. In this chapter, we present the design and implementation of ***CryptoImg***, a library of modular privacy preserving image processing operations over encrypted images. By using PHE, ***CryptoImg*** allows the users to delegate their image processing operations to remote servers without any privacy concerns. Currently, ***CryptoImg*** supports a subset of the most frequently used image processing operations, such as image adjustment, spatial filtering, edge sharpening, histogram equalization, and others. We implement our library as an extension to the popular computer vision library `OpenCV`. ***CryptoImg*** can be used from either mobile or desktop clients. Our experimental results demonstrate that ***CryptoImg*** is efficient while performing operations over encrypted images with negligible error and reasonable time overhead on the supported platforms.

The organization of the rest of this chapter is as follows. The motivation is discussed in Section 5.1. Section 5.2 surveys some of prior work related to applying secure operations on images over encrypted domain and using Paillier encryption scheme for securing image storage and retrieval. Section 5.3 describes the ***CryptoImg*** architecture. Section 5.4 describes our proposed secure operations in details. Section 5.5 provides our experimental evaluation results. Finally, Section 5.6 concludes the chapter.

## 5.1 Motivation

Cloud computing is one of the fastest growing technologies. In 2015, Gartner research selected cloud computing as one of the top ten technology trends. Software-as-a-Service (SaaS) is a class of cloud computing that allows thin clients, such as mobile devices or web browsers, to make use of centrally hosted software services on demand [69]. During the past few years, there has been a proliferation of commercial SaaS solutions for various application domains including image editing. For example, services like Adobe Creative Cloud [70], and Pixlr [71] allow the user to upload pictures from his/her personal computer (PC) or mobile device (Mob) in order to apply different image enhancements online.

However, image processing on the cloud presents a serious threat to user's privacy. A malicious service provider can look into the user private photos in order to discover sensitive information such as identity, friends, visited places, etc. As privacy is a crucial issue for end users, mitigating privacy concerns is necessary to increase the adoption of online image processing services.

In this chapter, we present ***CryptoImg***, a library of modular image processing operations over encrypted images. We implement our operations by extending the `OpenCV` library and employing the Paillier cryptosystem [19]. The major enhancement, which ***CryptoImg*** introduces as compared to previous work in the field, discussed in Section 5.2, is that ***CryptoImg*** can efficiently perform the needed computations with minimal overhead, while guaranteeing the secrecy of private images. ***CryptoImg*** omits the need for multiple non-collided servers [72]. Moreover, ***CryptoImg*** supports different operations including image adjustment, spatial filtering, edge sharpening, edge detection, morphological operations, and histogram equalization over encrypted images. In addition to that, we are the first to support secure morphological operations besides other image processing operations in one package.

To formulate our problem, we mainly study the problem of protecting the confidentiality of private images against third-party services performing image processing. Our threat model assumes that the clients trust their own hardware and locally-running software programs, but they do not trust third-party remote servers. Although the pressure of market competition forces service providers to perform the requested image enhancement operations correctly, these servers might threaten the user's privacy by abusing the given images to uncover private information for their own business interest. By giving the server access to nothing more than encrypted images, our system is secure under the "honest-but-curious" adversary model.

We rely on the Paillier cryptosystem which is provably secure using the hardness of the DCR assumption [19]. This means that it is infeasible for any attacker to break the encryption unless he/she has an efficient algorithm that can solve a family of problems that are computationally intractable. Compared to previous work in image processing over encrypted images, our solution provides better security guarantees than the model adopted by [72, 73], which requires more than one server and becomes insecure against colluding servers.

## 5.2 Related Work

We survey some of prior work related to applying secure operations on images over encrypted domain and using Paillier encryption scheme for securing image storage and retrieval. We highlight the differences between the previous work and ours.

Recently, various privacy preserving algorithms using HE have emerged in different domains including: information retrieval, data mining, and image processing. Shortell and Shokoufandeh addressed the problem of privacy-preserving image processing by using FHE to process the data while encrypted [74]. They used their solution to implement brightness/contrast filter. Moreover, they extended FHE to support FP numbers via multiplying each value by a factor of $10^d$, where $d$ depends upon the precision of the desired decimal digits up to which we want to process the FP numbers. However, the reported execution time was 15 minutes on a scaled down image and three hours on the original image.

Lathey and Atrey [72] introduced a privacy-preserving method for image processing based on Shamir's Secret Sharing (**SSS**) scheme [75]. This method distributes the image enhancement task among multiple servers to ensure privacy. Their solution supports a number of low-level image processing tasks carried out on encrypted images, such as spatial filtering, anti-aliasing, edge enhancement, and dehazing. Although this approach allows performing both addition and multiplication operations over encrypted data, the security of this model is guaranteed only if the computation is distributed over $n$ ($>1$) entities with no more than $k$ among them are colluding. This model is impractical, as it requires non colluding servers, i.e., the targeted cloud servers are not permitted to communicate with each others, and thus provides only weak security guarantees. Moreover, they employed different pre-processing for each secure operation. Therefore, a sequence of secure operations cannot be done without decryption and re-encoding.

Hu *et al.* proposed a double-cipher method to implement nonlocal means (NLM) denoising over encrypted images [76]. As the NLM operation includes exponentiation, which is a non linear operation, the authors encrypted the plain image with two different cryptosystems before sending to the cloud. The first one was the Paillier scheme, in order to enable the mean filter, and the other was obtained by a distance-preserving transform, in order to enable the nonlocal search. However, their proposed method has higher communication overhead, due to outsourcing two different ciphers for every image. They also enabled only a single type of image processing operations. Gomathisankaran *et al.* also introduced a HE method to ensure user privacy during the process of medical image analysis. They encrypted the images with the homomorphic Residue Number System (RNS) encryption scheme and sent the encrypted shares to the cloud to perform simple filtering and edge detection operations on them. In their work, they addressed the issue of operating on negative numbers in encrypted domain by encoding the negative values into two's complement 16-bit positive values. They conducted their experiments on magnetic resonance images with the usage of 64-bit keys [77].

Moreover, secure multi-party computation (SMC) has been utilized to protect privacy of outsourced images. Hu *et al.* implemented two secure linear filtering protocols [73]. The first one relied on a combination of rank reduction and random permutation. The second one is based on random perturbation with the help of a third party entity. In the context of secure image retrieval, Zhang *et al.* proposed a secure image retrieval method for cloud computing, which is implemented based on content-based image retrieval (CBIR) framework [78].

Hsu *et al.* proposed a privacy-preserving realization of the well known computer vision algorithm, scale-invariant feature transform (SIFT) [79], which is used for detecting and describing local features in image [80]. Their proposal was based on Paillier cryptosystem with integer values support only. This limitation resulted in significant errors, specially with the rounding operation in their gaussian filter coefficients. We handle this issue by using an appropriate encoding technique.

## 5.3 *CryptoImg* System Overview

As shown in Figure 5.1, ***CryptoImg*** consists of two parties: client and server. The client represents either an individual PC or Mob, while the server is a powerful system offering processing and storage services over the cloud. The client owns private image data and desires to make use of the server image processing services, while keeping the confidentiality of the submitted image against unauthorized access. To achieve this goal,

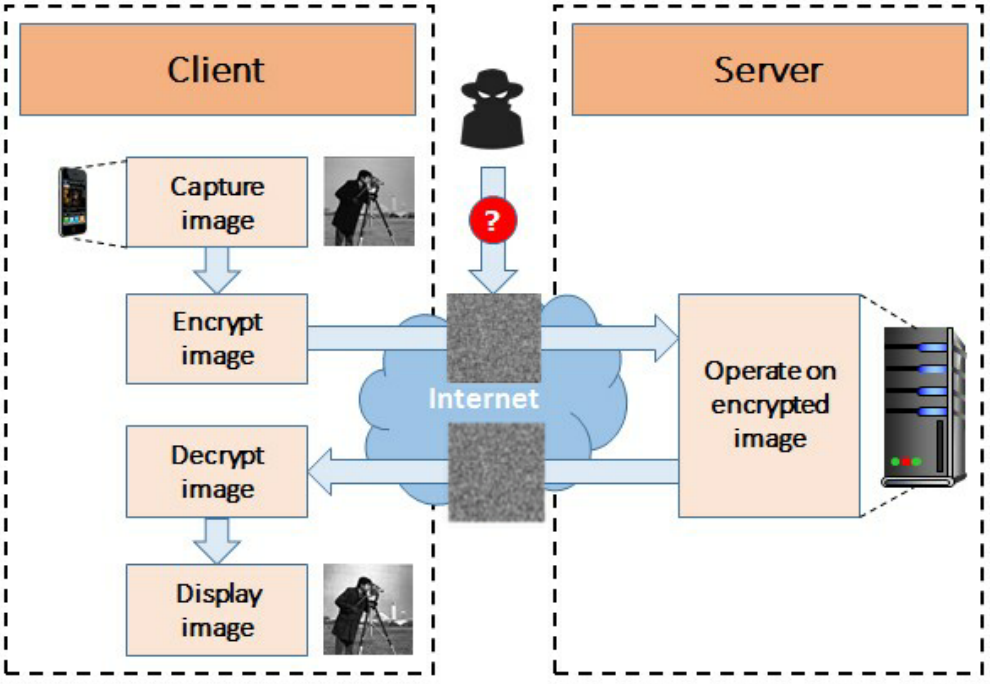



FIGURE 5.1: System Architecture of ***CryptoImg***.

the client encrypts the image before submitting it to the server. Using the PHE properties of Paillier cryptosystem, the server can perform operations over the encrypted image without revealing the source plain-image. The output encrypted image is sent back to the client to decrypt and display the processed image.

It is worth mentioning that, Paillier cryptosystem is defined over a group of positive integers $\mathbb{Z}_n$, while in practice many operations should happen over real numbers. Therefore, an encoding function $\mathbb{EN}$ with minimal quantization error is needed in order to perform secure computation over floating point (FP) numbers. We define $\phi_{add}$ and $\phi_{mul}$ as the error introduced due to addition and multiplication operations, as shown in (5.1) and (5.2), respectively. An optimal encoding mechanism should have $\phi_{mul} = \phi_{add} = 0$. Prior work over encrypted data represents FP numbers through multiplying by a large scaling factor as done in [74]. However, this representation has $\phi_{mul}$ equals the scale factor after each multiplication operation. Thus, it cannot be used with arbitrary number of multiplication operations over FP numbers.

$$\phi_{add} := abs(\,\mathbb{EN}(m_1 + m_2) - (\mathbb{EN}(m_1) + \mathbb{EN}(m_2)\,)) \tag{5.1}$$

$$\phi_{mul} := abs(\,\mathbb{EN}(m_1 \times m_2) - (\mathbb{EN}(m_1) \times \mathbb{EN}(m_2)\,)) \tag{5.2}$$

Therefore, we have chosen to use the same approach developed by Google's Encrypted BigQuery Client [81], which represents FP number by a mantissa $m$ and a non-positive

exponent $e$. A FP number in plaintext is represented by pair $(m, e)$. In encrypted domain, FP number is represented by a pair of an encrypted mantissa using Paillier cryptosystem and an unencrypted exponent $(\llbracket m \rrbracket, e)$. Self blinding and additive homomorphic over floats are denoted by $\otimes$ and $\oplus$, respectively. By using the addition and multiplication primitives $(\oplus_z, \otimes_z)$ of the Paillier cryptosystem, we can perform FP number addition and multiplication, as shown in Protocol 1. Additionally, signed numbers are handled by assigning the ranges $[0, n/3]$ and $[n/3, 2n/3]$ for positive and negative numbers, respectively, whereas the remaining range $(2n/3, n)$ is used for overflow detection. Subtraction accordingly over encrypted floats is denoted by $\ominus$.

**Protocol 1** Secure FP Numbers Processing.

**Multiplication:** $\llbracket c \rrbracket = a \otimes \llbracket b \rrbracket$
$\llbracket m_c \rrbracket = m_a \otimes_z \llbracket m_b \rrbracket$
$e_c = e_a + e_b$
**Addition:** $\llbracket c \rrbracket = \llbracket a \rrbracket \oplus \llbracket b \rrbracket$
**if** $e_a \leq e_b$
$\llbracket m_c \rrbracket = \llbracket m_a \rrbracket \oplus_z (Base^{e_b - e_a} \otimes_z \llbracket m_b \rrbracket)$, $e_c = e_a$
**if** $e_a > e_b$
$\llbracket m_c \rrbracket = \llbracket m_b \rrbracket \oplus_z (Base^{e_a - e_b} \otimes_z \llbracket m_a \rrbracket)$, $e_c = e_b$

## 5.4 Secure Operations in Encrypted Domain

The following subsections give details about the supported image processing operations by ***CryptoImg***.

### 5.4.1 Secure Image Adjustment

Image adjustment is done by applying transformation $T$ on an image $I$, which produces the resultant image $R$. We denote the individual pixels values in images $I$ and $R$ by $i$ and $r$, respectively. Therefore, the relationship between input pixels $i$ and output pixels $r$ can be represented by (5.3).

$$r = T(i) \tag{5.3}$$

***CryptoImg*** supports *brightness control* and *image negation*, as techniques for image adjustment.

For *brightness control*, the client requests to adjust the brightness of his/her image by adding value $v$, encrypting it along with the image pixels using his public key, $pk$, and sends both the encrypted value $[\![v]\!]$ and encrypted image $[\![I]\!]$ to the server. The server computes the encrypted values, $[\![r]\!]$, of output pixels for all pixels in the image using (5.4). Then, the server sends the encrypted image back to the client who will decrypt using its secret key $sk$.

$$[\![r]\!] = [\![i]\!] \oplus [\![v]\!] \tag{5.4}$$

***CryptoImg*** supports secure *Image negation* where the server computes the encrypted output pixel according to (5.5), for all pixels in input image with grey levels in the range $[0, L-1]$.

$$[\![r]\!] = [\![L-1]\!] \ominus [\![i]\!] \tag{5.5}$$

### 5.4.2 Secure Noise Reduction

Noise reduction and anti-aliasing operations are essential for many applications like medical and remote sensing images processing. Smoothing filter in spatial domain is a very common operation for anti-aliasing and noise removal, which is equivalent to a low pass filter (LPF) applied in the frequency domain. We denote the output image by $I_{spt}$ whose individual pixels $(u,v)$ are computed by performing average filter represented in (5.6). The filter $f(u,v)$ is applied first to $m \times n$ patch around $(u,v)$ pixel, then the intensity values of this patch are averaged.

$$[\![I_{spt}(u,v)]\!] = \frac{1}{m \times n} \otimes \sum_{u=1,v=1}^{m,n} f(u,v) \otimes [\![I(u,v)]\!] \tag{5.6}$$

The challenging part in mapping the average filtering operation to encrypted domain (ED) is how to map the division operation, which may result in a non integer result. As the original Paillier cryptosystem supports only operations over integers, we used our encoding technique, described in Section 5.3. It enables us to multiply by the FP term $1/(m \times n)$. Furthermore, arbitrary spatial filter masks can be applied in (5.6), as we do not restrict the filter value to be positive integers. On the other hand, authors in [74] did not support negative value in the filter mask.

$$h_1 = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \qquad h_2 = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

(A) Prewitt.

$$h_1 = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \qquad h_2 = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

(B) Sobel.

$$h_1 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -2 & 1 \\ -1 & -1 & -1 \end{bmatrix} \qquad h_2 = \begin{bmatrix} -1 & 1 & 1 \\ -1 & 2 & 1 \\ -1 & 1 & 1 \end{bmatrix}$$

(C) Robinson.

$$h_1 = \begin{bmatrix} 3 & 3 & 3 \\ 3 & 0 & 3 \\ -5 & -5 & -5 \end{bmatrix} \qquad h_2 = \begin{bmatrix} -5 & 3 & 3 \\ -5 & 0 & 3 \\ -5 & 3 & 3 \end{bmatrix}$$

(D) Kirsh.

FIGURE 5.2: The different edge detection operators supported by ***CryptoImg***. $h_1$ and $h_2$ represent the horizontal and vertical kernels, respectively [82].

### 5.4.3 Secure Edge Detection and Sharpening

Edge detection is one of the extremely important steps facilitating high-level image analysis [83]. Edges are pixels where image brightness changes abruptly, therefore gradient operators are commonly used to discover such pixels in the image. ***CryptoImg*** supports different kind of edge detection operators as Prewitt, Sobel, Robinson, and Kirsh, which are able to detect edges in different directions [82]. Those operators, which approximates the first derivative, are shown in Figure 5.2. Client sends the encrypted image to the server associated with the required operator identifier. Horizontal kernel $h_1$ and vertical kernel $h_2$ are convoluted with the encrypted image to find encrypted horizontal $[\![G_x]\!]$ and vertical $[\![G_y]\!]$ gradient components, as shown in (5.7) and (5.8). The client decrypts the resultant to find the gradient magnitude $G = \sqrt{{G_x}^2 + {G_y}^2}$ and gradient direction $\Theta = \text{atan2}\,(G_y, G_x)$ .

$$[\![G_x(u,v)]\!] = \sum_{u=1,v=1}^{m,n} h_1(u,v) \otimes [\![I(u,v)]\!] \tag{5.7}$$

$$[\![G_y(u,v)]\!] = \sum_{u=1,v=1}^{m,n} h_2(u,v) \otimes [\![I(u,v)]\!] \tag{5.8}$$

Additionally, edge sharpening operation in [74] can be reformulated, as shown in (5.9), in order to decrease the number of operations in the encrypted domain. Subtracting the

blurred image $I_{LPF}$ from the original one removes the low pass frequency component and yields the edge representation of the original image $I$. A positive constant, $k$, is used to control the amount of sharpening. For high-boost filtering, $k$ is greater than one, while it equals one in case of unsharp masking. $[\![I_{LPF}]\!]$ can be obtained using (5.6) using the appreciate mask.

$$[\![I_{shrp}(u,v)]\!] = ((k+1) \otimes [\![I(u,v)]\!]) \ominus (k \otimes [\![I_{LPF}(u,v)]\!]) \tag{5.9}$$

### 5.4.4 Secure Morphological Operations

Morphological operations represent a relatively separate part of image processing. They are widely used in many applications, such as document analysis, character recognition, industrial inspection, and the analysis of microscopic images in fields, like geology, biology, and material science. The basic idea in binary morphological operations, studied in this work, is to probe an image $I$ with a pre-defined shape, called the structuring element $B$ with size $m \times n$. The main two operations in binary morphology are erosion and dilation. Based on these two operations, more complex morphological operations can be computed, such as opening, closing, and shape decomposition. Protocol 2 describes the secure erosion and dilation operations. The erosion threshold value $T$ equals the number of ones in $B$. Conversely, the threshold value $T$ is equal to 1 to perform dilation.

**Protocol 2** Secure Morphological Operations.

1: Client sends $[\![I]\!]$ to the server associated with the requested structuring element $B$.
2: Server performs $[\![L(u,v)]\!] = \sum_{u=1,v=1}^{m,n} [\![I(u,v)]\!]$.
3: Server sends $[\![L]\!]$ to the client.
4: Client decrypts $[\![L]\!]$ using his private key. Then, image thresholding is applied on $L$ using threshold value $T$.

Additionally, it is worth mentioning that neither erosion nor dilation are invertible transformations. In other words, if a certain image is eroded and then dilated, the original image is not re-obtained. Moreover, eroded or dilated image could not be returned back to their original image using any other morphological operations. On the other hand, erosion followed by dilation creates an important morphological operations called "opening", whereas the dilation followed by erosion is called "closing". Opening and closing are two important morphological operations used to eliminate tiny image details and connecting small image objects, respectively.

### 5.4.5 Secure Histogram Equalization

Histogram equalization is a commonly used operation for contrast enhancement. It aims to create an image with equally distributed brightness levels over the whole brightness scale. As shown in Protocol 3, this goal is performed by calculating the cumulative image histogram $H_c$ for the input image. Then, a monotonic pixel brightness transformation $T(p)$ is applied such that the desired output histogram is almost uniform over the whole brightness scale. Original image histogram is denoted by $H$ and its size is $G$. Image size is $w \times \ell$. Intensity level is denoted by $p$.

**Protocol 3** Secure Histogram Equalization.

1: Client sends encrypted image histogram $H$.
2: Server computes the brightness transformation $T(p)$ as following:
$[\![H_c(0)]\!] = [\![H(0)]\!]$
$[\![H_c(p)]\!] = [\![H_c(p-1)]\!] \oplus [\![H(p)]\!], where\ p = 1, 2, \cdots G-1$
$[\![T(p)]\!] = (G-1)/(w \times \ell) \otimes [\![H_c(p)]\!]$.
3: Server sends $[\![T]\!]$.
4: Client decrypts and applies $T(p)$ on each image pixel.

## 5.5 Evaluation

The aim of this section is to describe the experimental setup needed to test our proposed library, ***CryptoImg***. In addition to that, the visual outputs and numerical timing results are shown and discussed in separate subsections.

### 5.5.1 Experimental Setup

***CryptoImg*** is implemented in C++ using GMP v6.0.0 [84] and NTL v9.5.0 [85]. as an extension to the popular computer vision library `OpenCV` [86]. We also developed an Android client application, which is implemented in Java. Our implementation of Paillier cryptosystem extends the work of [87] to introduce the FP support described earlier in Section 5.3.

For our experiments, we used a Intel Xeon(R) desktop machine with 8 cores at 2.20 GHz running Ubuntu 64-bit operating system. Our Android client application is installed on Nexus 5 (NX) mobile device, with Quad-core 2.30 GHz Krait 400 CPU.

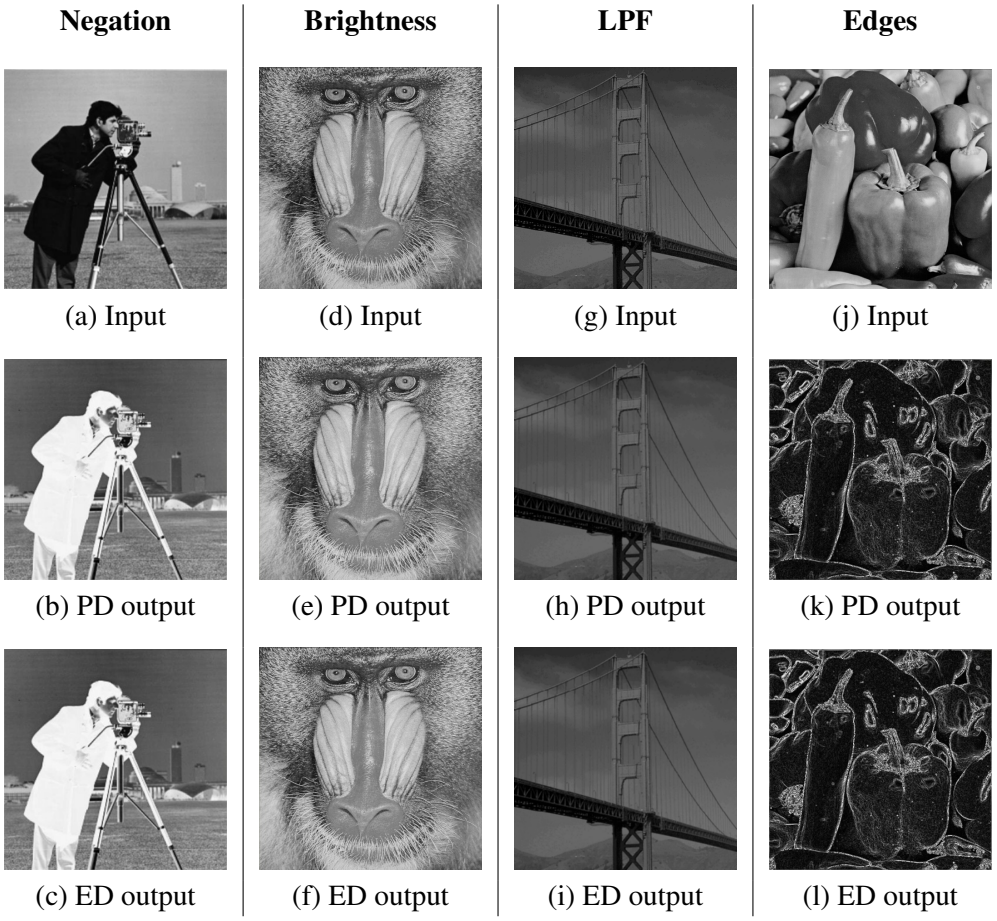



FIGURE 5.3: Visual output evaluation for the first operations set applied in plaintext domain (PD) and encrypted domain (ED) using $10^{-8}$ precision level.

### 5.5.2 Visual Output Results

We performed a number of experiments to evaluate the performance of different operations supported by ***CryptoImg***. We applied the operations to a number of gray level images from the public CVG-UGR gray level image database [88]. The dimensions of every image is $512 \times 512$ pixels and every pixel is represented by 8 bits. In case of morphological operations, selected binary images from another database [89]. Morphological operations are applied on binary images.

Figures 5.3 and 5.4 show the result of the proposed set of operations using a precision of $10^{-8}$. The precision determines the exponent of the encoded FP number using $\lfloor \log_{Base} precision \rfloor$. Here, we only show one output for each method. Figure 5.3-a represents the original images, which is encrypted using user Paillier public key and submitted to the server to obtain image negation. Figure 5.3-c shows the output after applying image negation in encrypted domain (ED). On the other hand, Figure 5.3-b

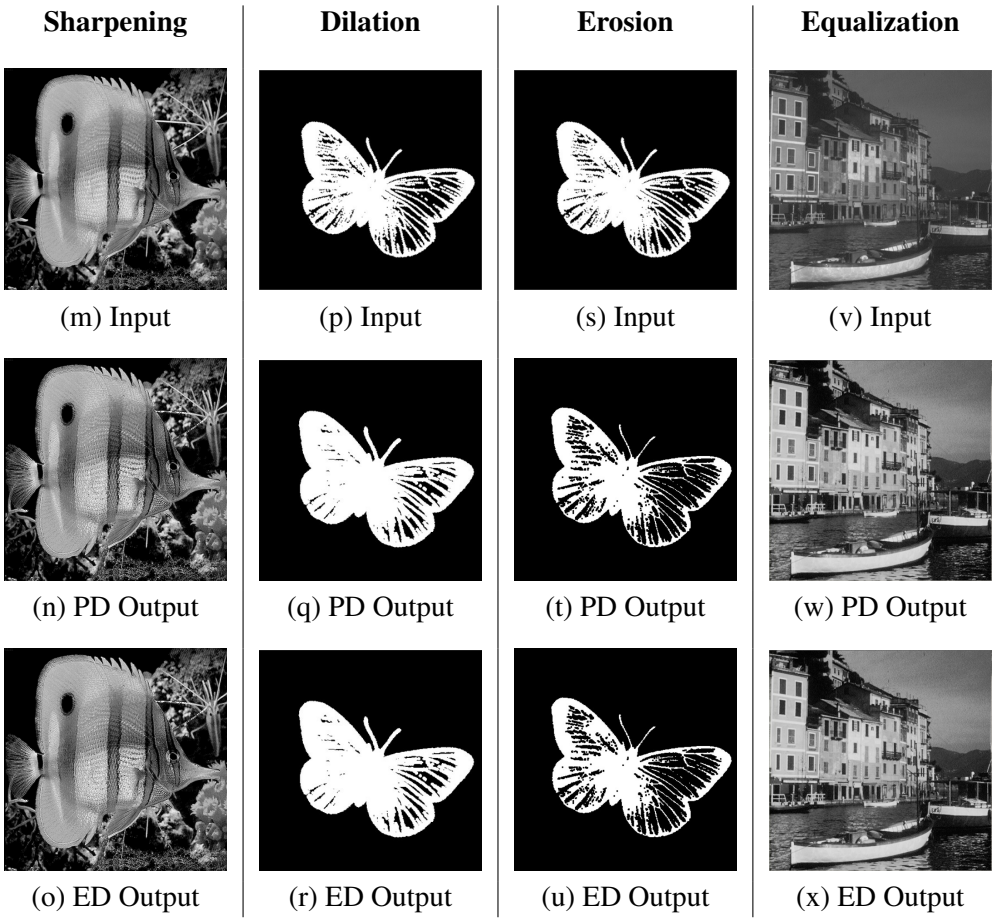



FIGURE 5.4: Visual output evaluation for the second operations set applied in plaintext domain (PD) and encrypted domain (ED) using $10^{-8}$ precision level.

shows the output of image negation in the plaintext domain (PD) using normal `OpenCV` APIs.

Figure 5.3-d through Figure 5.3-f show the same for brightness adjust. Additionally, Figure 5.3-i shows the decrypted output after applying secure averaging operation on Figure 5.3-g using a $3 \times 3$ filter. The visual effect of secure blurring and noise removal is compared with Figure 5.3-h, which is the normal average filter in the PD. For the sake of testing edge detection techniques, a simple Sobel filter is used to detect edges in Figure 5.3-j the outputs of the ED and PD are shown in Figure 5.3-l and Figure 5.3-k, respectively.

On the other hand, edge sharpening with $k = 1.0$ is applied on Figure 5.4-m. Edge sharpening in ED and PD are shown in Figure 5.4-o and Figure 5.4-n, respectively. Additionally, an example for the morphological operations is represented by applying a dilation operation on Figure 5.4-p. The output in PD and ED are shown in Figure 5.4-q and Figure 5.4-r, respectively. We also applied an erosion operation on Figure 5.4-s. The output in PD and ED are shown in Figure 5.4-t and Figure 5.4-u, respectively.

Finally, Protocol 3 is applied on Figure 5.4-v to perform histogram equalization in ED. The result is shown in Figure 5.4-x which is compared with PD outputs in Figure 5.4-w.

By comparing the output of operations in both encrypted and plain domains, we find that all our secure methods introduce zero error, except LPF and edge sharpening, which introduce a low error at higher precision. Table 5.1 shows the effect of choosing the precision level in the secure LPF and edge sharpening operations. The error is calculated by comparing the output in PD and ED. Based on that, we choose $10^{-8}$ as a reasonable precision.

### 5.5.3 Computation Time Results

We used two different implementations for Paillier cryptosystem for PC and Mob. Table 5.2 shows the computation time that ***CryptoImg*** takes to encrypt/decrypt images using different key sizes. The encryption/decryption process is done pixel by pixel. Therefore, if the original image size is $n \times n \times 8$ bits and a $k$ bit key is used, the size of the encrypted image would equal approximately $2k \times n \times n$ bits. That represents approximately a $k/4$ expansion factor. Histogram equalization operation does not require the encryption of all pixels. Only the histogram is encrypted, as explained in Protocol 3.

Zheng and Huang [90] suggested a way to tackle this issue by using characteristics of image format to compress the image after encryption. That helps limiting ciphertext expansion, while preserving the homomorphic property. Although their method had a smaller ciphertext expansion factor compared with the element-wise encryption method used here, it did not preserve the pixel location in ED, so position-aware operations like filtering and edge detection failed. Only image manipulation techniques would work well.

TABLE 5.1: Precision effect on the introduced ***CryptoImg*** error.

| Precision | Average Error | | Standard Deviation | |
|---|---|---|---|---|
| | LPF | Sharpening | LPF | Sharpening |
| $10^{-2}$ | 0.768 | 0.644 | 0.471 | 0.485 |
| $10^{-4}$ | 0.145 | 0.013 | 0.352 | 0.112 |
| $10^{-6}$ | 0.145 | 0.013 | 0.352 | 0.112 |
| $10^{-8}$ | 0.145 | 0.012 | 0.352 | 0.112 |
| $10^{-10}$ | 0.145 | 0.012 | 0.352 | 0.116 |

TABLE 5.2: Execution Time (sec) of the Paillier encryption/decryption of image using different key sizes on both personal computer (PC) and mobile device (Mob) clients. We used $512 \times 512$ images for PC and $256 \times 256$ images for Mob.

| Key Size | 256 | 512 | 1024 | 2048 |
|---|---|---|---|---|
| Encrypt-PC | 23.9164 | 156.905 | 1154.29 | 7670.49 |
| Decrypt-PC | 1.39223 | 1.93554 | 4.06813 | 9.62313 |
| Encrypt-Mob | 13 | 73 | 575 | 3701 |
| Decrypt-Mob | 10 | 48 | 325 | 2268 |

TABLE 5.3: Execution Time (sec) of the proposed operations using 1024-bit and 2048-bit keys on both personal computer (PC) and mobile device (Mob) clients in plaintext domain (PD) and encrypted domain (ED). The server is modeled as the PC. We used $512 \times 512$ images.

| Operation | PD | ED | | | | | |
|---|---|---|---|---|---|---|---|
| | | Pre-processing | | Server | | Post-processing | |
| | | PC | Mob | 1024-bit | 2048-bit | PC | Mob |
| Negation | 0.00122 | 0 | 0 | 42.4737 | 137.925 | 0 | 0 |
| Brightness | 0.00108 | 0 | 0 | 0.81994 | 2.39777 | 0 | 0 |
| LPF | 0.00763 | 0 | 0 | 180.508 | 609.199 | 0 | 0 |
| Sobel filter | 0.00642 | 0 | 0 | 147.567 | 482.195 | 0.0012 | 0.0940 |
| Sharpening | 0.00977 | 0 | 0 | 238.257 | 807.528 | 0 | 0 |
| Dilation | 0.00008 | 0 | 0 | 4.04937 | 10.8085 | 0.0005 | 0.0198 |
| Erosion | 0.00009 | 0 | 0 | 4.04937 | 10.8085 | 0.0006 | 0.0198 |
| Equalization | 0.00174 | 0.00182 | 0.177 | 0.01446 | 0.04835 | 0.0007 | 0.0290 |

Table 5.3 shows timing results of running our protocols using a PC or Mob clients with the configuration given in Section 5.3. For obtaining a high level of security, we set the Paillier key length to 1024-bits and 2048-bits in all scenarios. Edge sharpening is the most expensive operation, as it needs successive computations. The relatively high cost of the encryption process could be amortized by storing an encrypted version of the image on a cloud storage. The image is encrypted once and could be used as an input for many secure image processing operations.

Based on our results, we conclude that performing operations over encrypted images adds more cost in terms of computation time, communication, and storage. The added cost increases as the length of encryption key increases. However, our protocols add minimal computation overhead, which is orders of magnitude less than prior work

(e.g., [74]) and also minimal communication overhead, only one round between the client and server.

It is worth mentioning that not all image processing operations can be directly implemented in ***CryptoImg***. For instance, operations that require sorting/comparison, such as median filtering [82], would require more communication rounds between the client and server. Some gray-scale transformations, such as contrast manipulation, would not be feasible in ED as they rely on the knowledge of the original intensity value of the pixel to be able to map this value to another intensity value. More complicated algorithms, such as SIFT, could be supported in ED at the cost of adding more communication rounds between the client and server, a similar approach was used by [80].

## 5.6 Conclusion

In this chapter, we introduced ***CryptoImg***, a library of modular privacy preserving image processing operations over encrypted images based on the homomorphic properties of Paillier cryptosystem. Secure operations, such as image adjustment, spatial filtering, edge sharpening, edge detection, morphological operations, and histogram equalization, are safely outsourced to third-party servers with no privacy issues.

We presented how this operations can be implemented with much less time overhead, and single communication round. Moreover, ***CryptoImg*** can be used from either desktop or mobile clients with low client-side overheads. Experiments showed the efficiency of our proposed library. For instance, the image negation operation in the encrypted domain required less than one minute with zero error using 1024-bit key size.

The work discussed in this chapter was a joint work done with University of California at Los Angeles and was published in the 2nd IEEE Workshop on Security and Privacy in the Cloud (SPC), held in conjunction with the IEEE Conference on Communications and Network Security (CNS 2016) at Philadelphia, PA, USA, 2016 [91].

# Chapter 6

# Conclusion and Future Work

This chapter summarizes the work done in this Thesis. It also gives ideas for possible future work directions.

## 6.1 Conclusion

In this work, we tackled the problem of computing securely over encrypted data. Instead of going through the non-practical techniques of FHE, our target was to implement PHE methods and extend their functionality. We used three different applications to illustrate our idea. Numerical results showed the efficiency of our proposed ideas.

To conclude, the Thesis successfully managed to show the efficiency of using PHE techniques, such as ElGamal and Paillier, as a replacement of FHE ones in three different real-world problems, which require computing over encrypted data. The overheads accompanied by using such techniques are reasonable compared to the huge overheads of the FHE techniques reported in the literature.

## 6.2 Contributions

The Thesis provides three main contributions in the fields of securing e-voting machines against intruders [50], securing FPGA-based designs against untrusted third party IPs [92], and securing image processing operations over untrusted clouds [91].

### 6.2.1 Secure E-voting

The first domain of application was the e-voting domain. In this part, we highlighted e-voting challenges. We also implemented an e-voting system using a VGA screen and Xilinx FPGA board. Then, an HT was injected in our e-voting FPGA in order to manipulate the final results. We showed the role of homomorphic encryption in securing our design via the usage of ElGamal cryptosystem. Our proposed attack mainly relies on targeting the unused portion of the message bits moved between the input IP and the MicroBlaze main core. We suggested two different mechanisms to handle this problem. The first mechanism relied on resetting the unused bits of the sent data, where as the second mechanism utilized the SB method for data obfuscation. Moreover, we proved that our proposed solution has minimum side effects on the circuit power and timing delays. The reported power consumption overhead was very low and the overhead on the timing delay overhead was below 10%. Device resources overheads did not exceed 4%.

### 6.2.2 Secure FPGA-based Designs

Our second selected application is directly related to defeating Hardware Trojans in third party IPs using PHE techniques. PHE is sufficient enough with some third party IPs, which require a single type of operation or require both type of operations but not simultaneously. Thus, we implemented two designs that supports PHE (multiplicative only and additive only) based on ElGamal encryption/decryption scheme. Furthermore, we integrated the two designs together and introduced a dual-circuit design that achieved a great improvement in area and power over a regular design that combines two IPs, one for ElGamal and another for CEG, without any resource sharing between them. Our architectures were implemented on a Spartan-6 FPGA board from Xilinx. Area, delay, and power results were reported. The area reduction, compared to a regular non-shared design, reached 30% and savings in power consumption were 20% for encryption and 12% for decryption.

### 6.2.3 Secure Image Processing

Finally, we investigated the domain of securing image processing calculations. We introduced ***CryptoImg***, a library of modular privacy preserving image processing operations over encrypted images using the homomorphic properties of Paillier cryptosystem.

Secure operations, such as image adjustment, spatial filtering, edge sharpening, edge detection, morphological operations, and histogram equalization, are safely outsourced to third-party servers with no privacy issues. We presented how these operations can be implemented with a reasonable overhead and a single communication round. Moreover, ***CryptoImg*** can be used from either mobile or desktop clients with low client-side overheads. Experiments showed the efficiency of our proposed library. For instance, the image negation operation in the encrypted domain required less than one minute with zero error using 1024-bit key size.

## 6.3 Future Work

The work presented in this Thesis could go on a couple of possible future work directions.

### 6.3.1 Secure E-voting

The work described in Chapter 3 could be extended by adding more attacks whether on the ElGamal encryption scheme itself [93] or on the process of sending the packets, containing votes, from the e-voting machine to the main server [94].

### 6.3.2 Secure FPGA-based Designs

The work proposed in Chapter 4 can be extended through the evaluation of the hardware designs using different real-time scenarios and applications would prove the feasibility and efficiency of the implemented techniques against the FHE ones.

### 6.3.3 Secure Image Processing

Finally, it will be interesting to explore the feasibility of using the current secure operations, provided in Chapter 5, as building blocks to support more complex image processing algorithms. In addition to that, we could investigate the approaches suggested by Li *et al.* in order to reduce ciphertext expansion, which resulted from the usage of Paillier cryptosystem with very large keys [95].

# Bibliography


[1] R. Scott, H. G. Liddell, H. S. Jones, and R. McKenzie. *A Greek-English Lexicon: A New Edition*. Oxford, Clarendon, UK, 1940.

[2] D. S. Malik, J. N. Mordeson, and M. K. Sen. *Introduction to Abstract Algebra*. McGraw-Hill, Inc., New York, NY, USA, 2007.

[3] Y. Zhao, Y. Pan, S. Wang, and J. Zhang. An anonymous voting system based on homomorphic encryption. In *Proceedings of the 10th International Conference on Communications (COMM)*, Bucharest, Romania, May 2014.

[4] Z. Erkin, M. Franz, J. Guajardo, S. Katzenbeisser, I. Lagendijk, and T. Toft. Privacy-preserving face recognition. In *Proceedings of the 9th International Symposium on Privacy Enhancing Technologies (PETS)*, pages 235–253, Seattle, WA, USA, August 2009.

[5] M. Barni, T. Bianchi, D. Catalano, M. Di Raimondo, R. D. 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 *Proceedings of the 2010 Fourth IEEE International Conference on Biometrics: Theory, Applications and Systems (BTAS)*, Washington, DC, USA, September 2010.

[6] J. R. Troncoso-Pastoriza and F. Pérez-González. Zero-knowledge watermark detector robust to sensitivity attacks. In *Proceedings of the 8th Workshop on Multimedia and Security (MM & Sec)*, pages 97–107, Geneva, Switzerland, September 2006.

[7] R. Rivest, L. Adleman, and M. Dertouzos. On data banks and privacy homomorphisms. *Foundations of Secure Computation*, 4(11):169–180, 1978.

[8] C. Gentry. *A Fully Homomorphic Encryption Scheme*. PhD thesis, Stanford University, 2009. crypto.stanford.edu/craig, [online; Last visited May 19th, 2017].

[9] M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan. Fully homomorphic encryption over the integers. In *Proceedings of the 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, pages 24–43, French Riviera, France, May 2010.

[10] C. Gentry and S. Halevi. Implementing gentry's fully-homomorphic encryption scheme. In *Proceedings of the 30th Annual International Conference on Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, pages 129–148, Tallinn, Estonia, May 2011.

[11] Z. Brakerski, C. Gentry, and V. Vaikuntanathan. Leveled fully homomorphic encryption without bootstrapping. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference (ITCS)*, pages 309–325, Cambridge, MA, USA, January 2012.

[12] A. López-Alt, E. Tromer, and V. Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In *Proceedings of the Forty-fourth Annual ACM Symposium on Theory of Computing (STOC)*, pages 1219–1234, New York, NY, USA, 2012.

[13] J. W. Bos, K. Lauter, J. Loftus, and M. Naehrig. Improved security for a ring-based fully homomorphic encryption scheme. In *Proceedings of the 14th IMA International Conference on Cryptography and Coding (IMACC 2013)*, pages 45–64, Oxford, UK, December 2013.

[14] N. P. Smart and F. Vercauteren. Fully homomorphic SIMD operations. *Designs, Codes and Cryptography*, 71(1):57–81, 2014.

[15] N. Göttert, T. Feller, M. Schneider, J. Buchmann, and S. Huss. On the design of hardware building blocks for modern lattice-based encryption schemes. In *Proceedings of the 14th International Conference on Cryptographic Hardware and Embedded Systems (CHES)*, pages 512–529, Leuven, Belgium, September 2012.

[16] T. Pöppelmann and T. Güneysu. Towards efficient arithmetic for lattice-Based cryptography on reconfigurable hardware. In *Proceedings of the 2nd International Conference on Cryptology and Information Security in Latin America (LATINCRYPT)*, volume 7533, pages 139–158, Santiago, Chile, October 2012.

[17] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[18] J. B. Clarkson. Dense probabilistic encryption. In *Proceedings of the Workshop on Selected Areas of Cryptography (SAC)*, pages 120–128, Kingston, Ontario, Canada, May 1994.

[19] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Proceedings of the 17th International Conference on Theory and Application of Cryptographic Techniques (EUROCRYPT)*, pages 223–238, Prague, Czech Republic, May 1999.

[20] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *ACM Communications*, 21(2):120–126, February 1978.

[21] T. Elgamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.

[22] Y. Hu, W. J. Martin, and B. Sunar. Enhanced flexibility for homomorphic encryption schemes via CRT. In *Proceedings of the International Conference on Applied Cryptography and Network Security (ACNS)*, pages 93–110, Singapore, June 2012.

[23] K. Xiao, D. Forte, Y. Jin, R. Karri, S. Bhunia, and M. Tehranipoor. Hardware Trojans: Lessons learned after one decade of research. *ACM Transactions on Design Automation Electronics Systems*, 22(1):6:1–6:23, May 2016.

[24] M Tehranipoor and F Koushanfar. A survey of hardware trojan taxonomy and detection. *IEEE Design Test of Computers*, 27(1):10–25, 2010.

[25] S. Bhunia, M. S. Hsiao, M. Banga, and S. Narasimhan. Hardware Trojan attacks: Threat analysis and countermeasures. *Proceedings of the IEEE*, 102(8):1229–1247, August 2014.

[26] R. Karri, J. Rajendran, and K. Rosenfeld. Trojan taxonomy. In *Introduction to Hardware Security and Trust*, pages 325–338. Springer New York, New York, NY, USA, 2012.

[27] W. Hu, B. Mao, J. Oberg, and R. Kastner. Detecting hardware Trojans with gate-level information-flow tracking. *Computer*, 49(8):44–52, August 2016.

[28] Y. Jin, N. Kupp, and Y. Makris. Experiences in hardware Trojan design and implementation. In *Proceedings of the IEEE International Workshop on Hardware-Oriented Security and Trust (HOST)*, pages 50–57, San Francisco, CA, USA, July 2009.

[29] S. T. King, J. Tucek, A. Cozzie, C. Grier, W. Jiang, and Y. Zhou. Designing and implementing malicious hardware. In *Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats (LEET)*, pages 5:1–5:8, Berkeley, CA, USA, 2008.

[30] S. Everett, K. Greene, M. Byrne, D. Wallach, K. Derr, D. Sandler, and T. Torous. Is newer always better? The usability of electronic voting machines versus traditional methods. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, Florence, Italy, April 2008.

[31] T. Kohno, A. Stubblefield, A.D. Rubin, and D. S. Wallach. Analysis of an electronic voting system. In *Proceedings of the 2004 IEEE Symposium on Security and Privacy*, pages 27–40, Oakland, CA, USA, May 2004.

[32] E. Oksuzoglu and D. S. Wallach. VoteBox nano: A smaller, stronger FPGA-based voting machine. In *Proceedings of the 2009 USENIX/Accurate Electronic Voting Technology Workshop / Workshop on Trustworthy Elections (EVT/WOTE)*, Montreal, Canada, August 2009.

[33] Voting Technology Project. Voting: What is, what could be, Report of the Caltech MIT voting technology project, 2001.

[34] D. A. Kumar and T. U. S. Begum. Electronic voting machine x2014; a review. In *Proceedings of the 2012 International Conference on Pattern Recognition, Informatics and Medical Engineering (PRIME)*, pages 41–48, Salem, Tamilnadu, India, March 2012.

[35] N. Fauzia, T. Dey, I. Bhuiyan, and M. S. Rahman. An efficient implementation of electronic election system. In *Proceedings of the 10th international conference on Computer and Information Technology (ICCIT)*, Dhaka, Bangladesh, December 2007.

[36] M. R. Alam, M. Masum, M. Rahman, and A. Rahman. Design and implementation of microprocessor based electronic voting system. In *Proceedings of the 11th International Conference on Computer and Information Technology (ICCIT)*, pages 264–269, Khulna, Bangladesh, December 2008.

[37] T. Wollinger, J. Guajardo, and C. Paar. Cryptography on FPGAs: State of the art implementations and attacks. In *ACM Transactions on Embedded Computing Systems 3*, volume 3, pages 534–574, August 2004.

[38] A. Lesea. IP security in FPGAs. *White Paper (v1.0), Xilinx Inc.*, February 2007.

[39] Y. M. Alkabani and F. Koushanfar. Active hardware metering for intellectual property protection and security. In *Proceedings of the 16th USENIX Security Symposium (USENIX Security)*, pages 20:1–20:16, Boston, MA, USA, August 2007.

[40] S. Drimer and M. G. Kuhn. A protocol for secure remote updates of FPGA configurations. In *Proceedings of the 5th International Workshop on Reconfigurable Computing: Architectures, Tools and Applications (ARC)*, pages 50–61, Karlsruhe, Germany, 2009.

[41] D. Shantanu and L. Li. Trust-based design and check of FPGA circuits using two-level randomized ECC structures. *ACM Transactions on Reconfigurable Technology Systems*, 2(1):1–36, March 2009.

[42] M. Tehranipoor and F. Koushanfar. A survey of hardware Trojan taxonomy and detection. *IEEE Design & Test of Computers*, 27(1):10–25, January 2010.

[43] Y. Jin and Y. Makris. Hardware Trojan detection using path delay fingerprint. In *Proceedings of the IEEE International Workshop on Hardware-Oriented Security and Trust*, pages 51–57, Anaheim, CA, USA, June 2008.

[44] M. Banga, M. Chandrasekar, L. Fang, and M. S. Hsiao. Guided test generation for isolation and detection of embedded Trojans in ICs. In *Proceedings of the 18th ACM Great Lakes symposium on VLSI (GLSVLSI)*, pages 363–366, Orlando, FL, USA, May 2008.

[45] Xilinx. MicroBlaze processor reference guide, embedded development kit, EDK 14.6. UG081 (v14.6). https://www.xilinx.com/support/documentation/sw_manuals/xilinx14_6/mb_ref_guide.pdf. [online; Last visited May 19th, 2017].

[46] A. Al-Anwar, Y. Alkabani, M. W. El-Kharashi, and H. Bedour. Hardware Trojan protection for third party IPs on FPGA. In *Proceedings of the 16th Euromicro Conference on Digital System Design (DSD)*, pages 662–665, Santander, Spain, September 4–6, 2013.

[47] A. Waksman and S. Sethumadhavan. Silencing hardware backdoors. In *Proceedings of the 2011 IEEE Symposium on Security and Privacy (SP)*, pages 49–63, Oakland, CA, USA, May 2011.

[48] Xilinx Inc. Spartan-6 FPGA data sheet: DC and switching characteristics (DS162), v5.3. http://www.xilinx.com/support/documentation/data_sheets/ds162.pdf, June 27, 2014. [online; Last visited May 19th, 2017].

[49] Xilinx Power Analyzer overview. https://www.xilinx.com/support/documentation/sw_manuals/xilinx14_1/isehelp_start.htm#xpa_c_overview.htm. [online; Last visited May 19th, 2017].

[50] M. Tarek Ibn Ziad, A. Al-Anwar, Y. Alkabani, M. W. El-Kharashi, and H. Bedour. E-voting attacks and countermeasures. In *Proceedings of the 10th International Symposium on Frontiers of Information Systems and Network Aplications (FINA 2014), held in conjunction with the 28th IEEE International Conference on Advanced Information Networking and Applications (AINA-2014)*, pages 269–274, Victoria, BC, Canada, May 13–16, 2014.

[51] N. Yoshimizu. Hardware Trojan detection by symmetry breaking in path delays. In *Proceedings of the 2014 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pages 107–111, Arlington, VA, USA, May 2014.

[52] R. Rad, J. Plusquellic, and M. Tehranipoor. A sensitivity analysis of power signal methods for detecting hardware Trojans under real process and environmental conditions. *IEEE Transactions on Very Large Scale Integrated Systems*, 18:1735–1744, December 2010.

[53] H. Salmani, M. Tehranipoor, and J. Plusquellic. New design strategy for improving hardware Trojan detection and reducing Trojan activation time. In *Proceedings of the IEEE International Workshop on Hardware-Oriented Security and Trust (HOST)*, pages 66–73, San Francisco, CA, USA, July 2009.

[54] J. Rajendran, V. Jyothi, O. Sinanoglu, and R. Karri. Design and analysis of ring oscillator based Design-for-Trust technique. In *Proceedings of the 2011 IEEE 29th VLSI Test Symposium (VTS)*, pages 105–110, Dana Point, CA, USA, May 2011.

[55] A. Al-Anwar, Y. Alkabani, M. W. El-Kharashi, and H. Bedour. Defeating hardware spyware in third party IPs. In *Proceedings of the 2nd Saudi International Electronics, Communications and Photonics Conference (SIECPC)*, Riyadh, Saudi Arabia, April 27–30, 2013.

[56] A. Al-Anwar, Y. Alkabani, M. W. El-Kharashi, and H. Bedour. Hardware Trojan detection methodology for FPGA. In *Proceedings of the 2013 IEEE Pacific Rim Conference on Communications, Computers, and Signal Processing (PacRim)*, pages 177–182, Victoria, BC, Canada, August 27–29, 2013.

[57] A. Alanwar, M. A. Aboelnaga, Y. Alkabani, M. W. El-Kharashi, and H. Bedour. Dynamic FPGA detection and protection of hardware Trojan: A comparative analysis. *International Journal of Computing and Digital Systems*, 5(3):211–223, May 2016.

[58] M. Tehranipoor, H. Salmani, X. Zhang, X. Wang, R. Karri, J. Rajendran, and K. Rosenfeld. Trustworthy hardware: Trojan detection and Design-for-Trust challenges. *Computer*, 44(7):66–74, July 2011.

[59] A. Baumgarten, A. Tyagi, and J. Zambreno. Preventing IC piracy using reconfigurable logic barriers. *IEEE Design & Test of Computers*, 27(1):66–75, January 2010.

[60] X. Zhang and M. Tehranipoor. Case study: Detecting hardware Trojans in third-party digital IP cores. In *Proceedings of the 2011 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pages 67–70, San Diego, CA, USA, June 2011.

[61] S. Moein, J. Subramnian, T. Aaron Gulliver, Fayez Gebali, and M. W. El-Kharashi. Classification of hardware Trojan detection techniques. In W. G. A. Abdelaal, M. W. El-Kharashi, A. M. Bahaa El-Din, M. Taher, and A. M. Zaki, editors, *Proceedings of the 2015 Tenth International Conference on Computer Engineering and Systems (ICCES)*, pages 357–362, Cairo, Egypt, December 23–24, 2015.

[62] S. Moein, S. Khan, T. Aaron Gulliver, Fayez Gebali, and M. W. El-Kharashi. An attribute based classification of hardware Trojans. In W. G. A. Abdelaal, M. W. El-Kharashi, A. M. Bahaa El-Din, M. Taher, and A. M. Zaki, editors, *Proceedings of the 2015 Tenth International Conference on Computer Engineering and Systems (ICCES)*, pages 351–356, Cairo, Egypt, December 23–24, 2015.

[63] S. Moein, Fayez Gebali, T. Aaron Gulliver, and M. W. El-Kharashi. Hardware attack risk assessment. In W. G. A. Abdelaal, M. W. El-Kharashi, A. M. Bahaa El-Din, M. Taher, and A. M. Zaki, editors, *Proceedings of the 2015 Tenth International Conference on Computer Engineering and Systems (ICCES)*, pages 346–350, Cairo, Egypt, December 23–24, 2015.

[64] C. Konstantinou, A. Keliris, and M. Maniatakos. Privacy-preserving functional IP verification utilizing fully homomorphic encryption. In *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 333–338, Grenoble, France, 2015.

[65] P. L. Montgomery. Modular multiplication without trial division. *Mathematics of Computation*, 44(170):519–521, 1985.

[66] J. P. Deschamps. *Hardware Implementation of Finite-Field Arithmetic*. McGraw-Hill, Inc., New York, NY, USA, 2009.

[67] J. P. Deschamps and G. Sutter. Hardware implementation of finite-field division. *Acta Applicandae Mathematica*, 93(1-3):119–147, 2006.

[68] Xilinx Inc. Xilinx ISE design suite 14.6 software manuals. http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_6/, June 2013. [online; Last visited May 19th, 2017].

[69] V. Choudhary. Software as a Service: Implications for investment in software development. In *Proceedings of the 40th Annual Hawaii International Conference on System Sciences (HICSS)*, Waikoloa, Big Island, Hawaii, January 2007.

[70] Adobe creative cloud. http://www.adobe.com/creativecloud.html. [online; Last visited May 19th, 2017].

[71] Pixlr: Photo Editor Online. https://pixlr.com/web. [online; Last visited May 19th, 2017].

[72] A. Lathey and P. K. Atrey. Image enhancement in encrypted domain over cloud. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 11(3):38:1–38:24, February 2015.

[73] N. Hu, S. S. Cheung, and T. Nguyen. Secure image filtering. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, pages 1553–1556, Atlanta, GA, USA, October 2006.

[74] T. Shortell and A. Shokoufandeh. Secure signal processing using fully homomorphic encryption. In *Proceedings of the 16th International Conference on Advanced Concepts for Intelligent Vision Systems (ACIVS)*, pages 93–104, Catania, Italy, October 2015.

[75] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, November 1979.

[76] X. Hu, W. Zhang, K. Li, H. Hu, and N. Yu. Secure nonlocal denoising in outsourced images. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 12(3):40:1–40:23, March 2016.

[77] M. Gomathisankaran, X. Yuan, and P. Kamongi. Ensure privacy and security in the process of medical image analysis. In *Proceedings of the 2013 IEEE International Conference on Granular Computing (GrC)*, pages 120–125, Beijing, China, December 2013.

[78] Y. Zhang, L. Zhuo, Y. Peng, and J. Zhang. A secure image retrieval method based on homomorphic encryption for cloud computing. In *Proceedings of the 19th International Conference on Digital Signal Processing (DSP)*, pages 269–274, August 2014.

[79] D. G. Lowe. Distinctive image features from scale-Invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, November 2004.

[80] C. Y. Hsu, C. S. Lu, and S. C. Pei. Image feature extraction in encrypted domain with privacy-preserving SIFT. *IEEE Transactions on Image Processing*, 21(11):4593–4607, November 2012.

[81] Google encrypted bigquery client. https://github.com/google/encrypted-bigquery-client. [online; Last visited May 19th, 2017].

[82] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*. Pearson, 2007.

[83] M. Sonka, V. Hlavac, and R. Boyle. Image pre-processing. In *Image Processing, Analysis and Machine Vision*, pages 56–111. Springer US, 1993.

[84] GMP: GNU multiple precision arithmetic library, v6.0.0. https://gmplib.org/. [online; Last visited May 19th, 2017].

[85] NTL: A library for doing number theory, v9.5.0. http://www.shoup.net/ntl/. [online; Last visited May 19th, 2017].

[86] OpenCV: Open source computer vision, v3.2. http://opencv.org/. [online; Last visited May 19th, 2017].

[87] R. Bost, R. A. Popa, S. Tu, and S. Goldwasser. Machine learning classification over encrypted data. https://github.com/rbost/ciphermed/tree/master/src/crypto, 2015. [online; Last visited May 19th, 2017].

[88] CVG-UGR image database. http://decsai.ugr.es/cvg/dbimagenes/. [online; Last visited May 19th, 2017].

[89] MPEG7 CE Shape-1 Part B image database. http://www.imageprocessingplace.com/downloads_V3/root_downloads/image_databases/. [online; Last visited May 19th, 2017].

[90] P. Zheng and J. Huang. An efficient image homomorphic encryption scheme with small ciphertext expansion. In *Proceedings of the 21st ACM International Conference on Multimedia (MM)*, pages 803–812, Barcelona, Catalunya, Spain, October 2013.

[91] M. Tarek Ibn Ziad, A. Alanwar, M. Alzantot, and M. Srivastava. CryptoImg: Privacy preserving processing over encrypted images. In *Proceedings of the 2nd IEEE Workshop on Security and Privacy in the Cloud (SPC), held in conjunction with the IEEE Conference on Communications and Network Security (CNS)*, pages 570–575, Philadelphia, PA, USA, October 2016.

[92] M. Tarek Ibn Ziad, A. Alanwar, Y. Alkabani, M. W. El-Kharashi, and H. Bedour. Homomorphic data isolation for hardware Trojan protection. In *Proceedings of the IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pages 131–136, Montpellier, France, July 8–10, 2015.

[93] B. Allen. *Implementing several attacks on plain ElGamal encryption*. PhD thesis, Iowa University, 2008.

[94] N. G. Tsoutsos and M. Maniatakos. Cryptographic vote-stealing attacks against a partially homomorphic e-voting architecture. In *Proceedings of the 2016 IEEE 34th International Conference on Computer Design (ICCD)*, pages 157–160, Scottsdale, AZ, USA, October 2016.

[95] Y. Li, J. Zhou, Y. Li, and O. C. Au. Reducing the ciphertext expansion in image homomorphic encryption via linear interpolation technique. In *Proceedings of the 2015 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pages 800–804, Orlando, FL, USA, December 2015.

# شكر

الثناء كله لله الرحمن الرحيم رب العالمين الذي علم الإنسان ما لم يكن يعلم. أود أن أشكر الله عز وجل على منحي الفرصة والقوة والقدرة على إتمام هذا العمل.

دائما ما تواجهني صعوبات شديدة في كتابة كلمات الشكر فالكلمات لا تستطيع أبدا أن تعبر عن مدي ثنائي وامتناني ومع ذلك سوف أبذل قصاري جهدي. أولا، أود أن أعبر عن امتناني العميق لمشرفيَ الأفاضل; دكتور حسن محمد شحاتة بدور ودكتورة يسرا محسن علي القباني وذلك علي توجيههما المستمر وملاحظاتهما الهامة بشأن النتائج المدونة بهذا العمل وكتابته بوجه عام. كما أود أن أتوجه بالشكر للأستاذ الدكتور محمد واثق علي كامل الخراشي علي دعمه الدائم والمتواصل.

أشكر أيضا الباحثين التالية أسمائهم من جامعة كاليفورنيا لوس أنجلوس وهم عمرو الأنور ومصطفي الزنتوت وماني سيرفستافا وذلك علي معاونتهم لشخصي في تطوير ومراجعة محتويات الفصل الخامس.

أود أن أشكر والديَ علي دعمهم المتواصل طوال عمري وعلي جهدهم طوال فترة إعداد هذه الرسالة. أيضا يملأني الإمتنان لأصدقائي الرائعين الذين جعلوني أتمتع بالكثير من جوانب الحياة خارج نطاق العمل.

**الفصل الرابع** يقدم الإسهام الثاني لهذه الرسالة وهو تأمين التصميمات الخاصة بمصفوفة البوابات المنطقية القابلة للبرمجة ضد أحصنة طروادة وذلك باستخدام الخاصية التماثلية. حيث يقوم الباحث بتنفيذ نظامي تشفير تماثلي جزئيين معتمدا علي نظام التشفير المسمي بالجمل. يمتاز النظام الاول بكونه يدعم خاصية التماثلية لعملية الضرب بينما يدعم النظام الثاني خاصية التماثلية لعملية الجمع. يشرح الفصل أيضا التنفيذ العملي للتصميم السابق علي مصفوفة البوابات المنطقية القابلة للبرمجة منخفضة الثمن كما يوضح أيضا نتائج المساحة المنطقية والنتائج الزمنية.

**الفصل الخامس** يستعرض الإسهام الثالث والأخير لهذه الرسالة وهو عبارة عن مكتبة برمجة سحابية تسمح بإجراء عمليات معالجة الصورعلي الصور المشفرة. يقدم الفصل طرق جديدة ويوضح الاثار المترتبة عليها من حيث عدد مرات الاتصال والعمليات الحسابية المستخدمة وذلك بواسطة عدد من الاختبارات العملية.

**الفصل السادس** يشتمل علي ملخص هذا العمل وما اشتمله من إسهامات ويستعرض عددا من الحلول المستقبلية للإضافة إلي هذا البحث.

**الكلمات المفتاحية** – التصويت الالكتروني، نظام التشفير الجمل، التشفير التماثلي الكلي، أحصنة طروادة، الخاصية التماثلية، معالجة الصور، نظام التشفير بيلر، التشفير التماثلي الجزئي، تأمين الحسابات.

# الملخص

إن التشفير هو فن تحويل الرسائل النصية إلي رموز سرية وذلك باستخدام مفتاح للشفرة. منذ مئات السنين، اعتاد الناس علي تقديم طرق تشفير جديدة لتأمين بياناتهم وحماية خصوصياتهم. لسوء الحظ، بمجرد تشفير البيانات فإنها تبقي سرية وغير مفيدة حتي يتم فك شفرتها باستخدام المفتاح الملائم. يعتبر التشفير التماثلي نوع من أنواع التشفير ولكنه يسمح للشخص بإجراء عمليات حسابية مفيدة علي البيانات المشفرة مباشرة بدون الحاجة إلي إزالة تشفيرهم مسبقا. أما عن نتائج هذه العمليات الحسابية فهي تصبح مكافئة تماما لنتائج نظيراتها المقامة علي البيانات الغير مشفرة. وبهذه الطريقة، يمكن للشخص السماح لطرف ثالث بإجراء عمليات مفيدة علي بياناته الخاصة بدون التضحية بخصوصيته.

بناء علي الوصف السابق، تهدف هذه الرسالة إلي استكشاف مدي فاعلية استخدام أساليب التشفير التماثلي في حل المشكلات الواقعية التي يعتبر إجراء العمليات فيها علي البيانات المشفرة ضرورة هامة. وعلي الرغم من تضمن التشفير التماثلي إلي نوعين مختلفين وهما التشفير التماثلي الكامل والتشفير التماثلي الجزئي، فإن تركيز هذه الرسالة ينصب علي النوع الثاني فقط. يرجع السبب وراء هذا الإتجاه البحثي إلي رغبتنا في تفادي الاثار الجانبية الشديدة المتعلقة باستخدام طرق التشفير التماثلي الكلية.

تنقسم الرسالة إلي ستة فصول بالإضافة الي قوائم بالمحتويات والاشكال الواردة بالرسالة والجداول والاختصارات والرموز وأيضا قائمة بالمراجع.

أما عن محتويات الرسالة فهي كالتالي.

**الفصل الاول** يستعرض مقدمة الرسالة ويوضح الحافز الرئيسي وراء تقديم هذا العمل. كما يقوم أيضا ببيان إسهامات هذه الرسالة والتي تتفرع الي ثلاثة تطبيقات متنوعة. سيتم شرح كل تطبيق من هذه التطبيقات في فصل منفصل بالإضافة إلي التجارب العملية والنتائج الرقمية.

**الفصل الثاني** يقدم الخلفية العلمية اللازمة عن خاصية التماثلية وطرق التشفير التماثلي بنوعيها الكلية والجزئية. كما يركز هذا الفصل بشكل خاص علي نظامي التشفير التماثلي الجزئيين; الجمل وبيلر.

**الفصل الثالث** يشرح بالتفصيل الإسهام الأول في هذه الرسالة وهو استخدام الخاصية التماثلية في أنظمة التصويت الالكتروني. حيث يقوم هذا الفصل بتقديم عددا من الهجمات المحتملة علي ماكينة تصويت الكتروني مبنية باستخدام (مصفوفة البوابات المنطقية القابلة للبرمجة) بالإضافة الي طرق الحماية من تلك الهجمات. كما يستعرض الفصل الشرح الكامل للتطبيق المنفذ وكذلك الاثار الجانبية للحلول المقترحة.

وقمت بتنفيذ التصميمين علي مصفوفة بوابات منطقية قابلة للبرمجة حيث بلغ مقدار التوفير في المساحة 30% بالاضافة إلي توفير الطاقة اللازمة لعمليتي التشفير وفك التشفير بمقدار 20% و12% علي الترتيب.

أما من جهة اجراء العمليات مباشرة علي الصور المشفرة، فقد قمت باقتراح برنامج لإجراء عمليات معالجة الصورعلي الصور المشفرة مع المحافظة علي خصوصية تلك الصور وذلك بالاعتماد علي نظام التشفير التماثلي المعروف بيلر. وبذلك نجحت في تحويل العمليات الأمنة من أمثال تعديل إضاءة الصور وتنقيتها وزيادة حدتها واستخراج الأطراف الحادة منها وكذلك العمليات الشكلية إلي جهة ثالثة بدون أي مشاكل تتعلق بالخصوصية. كما أوضحت كيف يمكن تنفيذ العمليات السابق ذكرها بأقل اثار جانبية ممكنة من حيث الزمن ومرات الاتصال. بالإضافة إلي ما سبق، يمكن لهذا البرنامج التعامل مع أجهزة الحاسب الالي المكتبية أو الهواتف النقالة وقد أثبتت التجارب العملية فاعليتها. علي سبيل المثال، احتاجت عملية عكس إحدي الصور إلي أقل من دقيقة واحدة لإجرائها في المجال المشفر بمقدار خطأ يساوي الصفر وباستخدام مفتاح تشفير حجمه 1024 بت.

وفي الختام، فقد نجحت الرسالة في إظهار فاعلية استخدام طرق التشفير التماثلي الجزئية مثل الجمل وبيلر كبديل لطرق التشفير التماثلي الكلية وذلك في ثلاثة تطبيقات عملية مختلفة تستلزم إجراء عمليات حسابية علي البيانات المشفرة. وتعتبر الاثار الجانبية المصاحبة لاستخدام طرق التشفير التماثلي الجزئية تلك مقبولة مقارنة بالاثار الجانبية الضخمة المصاحبة لاستخدام طرق التشفير التماثلي الكلية كما موضح بالأبحاث السابقة.

# المستخلص


إن الكمية الضخمة من البيانات المتوفرة في عصرنا الحالي تدع الباب مفتوحا علي مصراعيه أمام استخدام شركات من طرف ثالث بغرض التعامل مع هذه البيانات وتخزينها. هذا من شأنه أن يثير العديد من الشكوك المتعلقة بخصوصية المستخدمين، وإذا ما كانت شركات الطرف الثالث تلك موثوق بها أم لا. فمن جهة، لا يستطيع المستخدمون - سواء كانوا أفرادا عاديين أو مؤسسات - تحمل تكلفة وتعقيد إجراء العمليات المطلوبة علي البيانات الخاصة بهم باستخدام مكونات محلية تحت سيطرتهم. ومن جهة أخري، سيعد مجرد الاعتماد فقط علي شركات الطرف الثالث - مثل خدمات الحوسبة السحابية - بدون أخذ السرية والامان بالاعتبار كمن يبني قلاعا حصينة من الطين. لهذا يعد استخدام نظم التشفير التماثلي أحد الحلول الممكنة للمشكلة السابق ذكرها حيث تسمح هذه النظم لشركات الطرف الثالث بإجراء عملياتها علي البيانات مع الحفاظ علي سرية هذه البيانات. ومن هنا يمكننا الاستفادة من القدرات الحسابية الضخمة لهذه الشركات بدون التضحية بخصوصيتنا.

تنقسم نظم التشفير التماثلية إلي نوعين رئيسيين أحدهما نظام تشفير تماثلي جزئي والأخر نظام تشفير تماثلي كامل. بينما يمتاز نظام التشفير التماثلي الكامل بقدرته علي حل مشكلة الخصوصية بصورة تامة فإنه يسبب اثارا جانبية شديدة علي الأداء. ومن أجل تفادي هذه الاثار، يمكننا استخدام نظام التشفير الجزئي عوضا عنه. لهذا كان الغرض الاساسي لهذه الرسالة هو *استكشاف مدي فاعلية استخدام طرق التشفير التماثلي الجزئية في حل المشكلات الواقعية والتي يلزم فيها إجراء العمليات علي البيانات المشفرة مباشرة*.

تقدم هذه الرسالة عدة إسهامات حيث تم اختيار ثلاثة تطبيقات مختلفة وهي تأمين نظم التصويت الالكترونية، والقضاء علي أحصنة طروادة داخل التصميمات الخاصة بمصفوفات البوابات المنطقية القابلة للبرمجة، وأخيرا إجراء العمليات مباشرة علي الصور المشفرة. إن العامل المشترك بين هذه التطبيقات المتنوعة هو توافر بيانات مشفرة في حاجة إلي التعامل معها بواسطة طرف ثالث بدون انتهاك خصوصية هذه البيانات.

من جهة تأمين نظم التصويت الالكترونية، تم بناء نظام تصويت الكتروني باستخدام مصفوفة البوابات المنطقية القابلة للبرمجة وشاشة تصويت بالإضافة إلي خادم الكتروني بعيد لحصر النتائج. كما قمت باقتراح عدد من الهجمات علي النظام السابق ذكره من خلال زراعة أحصنة طروادة بداخله بهدف التلاعب بنتيجة التصويت النهائية. أيضا أوضحت دور التشفير التماثلي في تأمين التصميم من خلال استخدامه لنظام التشفير التماثلي المسمي الجمل وشرح كذلك طرق الحماية المقترحة. ومن ثم، تم تقييم تلك الطرق بناء علي الاثار الجانبية الناجمة عنها من حيث المساحة والزمن واستهلاك الطاقة حيث بلغ المعدل الاضافي للطاقة المستهلكة حدا ضئيلا كما لم يتجاوز التأخير الزمني 10% كذلك استهلاك مصادر الجهاز المستعمل لم يتعدي ال 4%.

من جهة القضاء علي أحصنة طروادة داخل التصميمات الخاصة بمصفوفات البوابات المنطقية القابلة للبرمجة، قمت بتنفيذ نظامي تشفير تماثلي جزئيين يدعمان عملية الجمع بمفردها وعملية الضرب بمفردها وذلك بالاعتماد علي نظام التشفير المسمي بالجمل. بالإضافة إلي ذلك قمنا بدمج النظامين معا للحصول علي نظام ثنائي يمتاز بتوفير بالغ في المساحة والطاقة المستهلكة مقارنة بالنظامين منفردين.

# تعريف بمقدم الرسالة

| | | |
|---|---|---|
| الاسم | : | محمد طارق بن زياد محمد حسن |
| تاريخ الميلاد | : | 25 - 06 - 1992 |
| محل الميلاد | : | القاهرة، مصر |
| آخر درجه جامعية | : | بكالوريوس العلوم الهندسية في الهندسة الكهربية |
| الجهة المانحة | : | جامعة عين شمس |
| تاريخ المنح | : | 2014 |
| الوظيفه الحالية | : | معيد بكلية الهندسة، جامعة عين شمس |

-


كلية الهندسة
قسم هندسة الحاسبات والنظم

رسالة الماجستير:

اسم الطالب : محمد طارق بن زياد محمد حسن
عنوان الرسالة : التشفير التماثلي لتأمين العمليات الحسابية علي البيانات
اسم الدرجة : ماجستير العلوم في الهندسة

لجنة الإشراف:

د/ حسن محمد شحاتة بدور
د/ يسرا محسن على القبانى

تاريخ البحث : ...../...../.....

الدراسات العليا:

ختم الإجازة: أجيزت الرسالة بتاريخ : ...../...../.....

موافقة مجلس الكلية : ...../...../.....
موافقة مجلس الجامعة : ...../...../.....

جامعة عين شمس
كلية الهندسة
قسم هندسة الحاسبات والنظم

اسم الباحث: **محمد طارق بن زياد محمد حسن**
**قسم الحاسبات والنظم**
**كلية الهندسة – جامعة عين شمس**

عنوان الرسالة: **التشفير التماثلي لتأمين العمليات الحسابية علي البيانات**

الدرجة: **ماجستير العلوم في الهندسة الكهربية (هندسة الحاسبات والنظم)**

**لجنة الفحص والحكم:**

| | | **التوقيع** |
|---|---|---|
| **أ. د./ حسام على حسن فهمى**<br>استاذ بقسم هندسة الإلكترونيات والإتصالات<br>كلية الهندسة - جامعة القاهرة | (مناقشا) | ..... |
| **أ. د./ محمد واثق علي كامل الخراشي**<br>استاذ بقسم هندسة الحاسبات والنظم<br>كلية الهندسة - جامعة عين شمس | (مناقشا) | ..... |
| **د./ حسن محمد شحاتة بدور**<br>استاذ مساعد بقسم هندسة الحاسبات والنظم<br>كلية الهندسة - جامعة عين شمس | (عن لجنة الإشراف) | ..... |

تاريخ المناقشة: **يونيو – 2017**

كلية الهندسة

قسم هندسة الحاسبات والنظم

# التشفير التماثلي لتأمين العمليات الحسابية علي البيانات

رسالة مقدمة للحصول على درجة ماجستير العلوم فى الهندسة
فى الهندسة الكهربية
( هندسة الحاسبات والنظم)

إعداد

محمد طارق بن زياد محمد حسن

حاصل على بكالوريوس العلوم الهندسية
فى الهندسة الكهربية
(هندسة الحاسبات والنظم)
كلية الهندسة، جامعة عين شمس، سنة 2014

***المشرفون***

**د./ حسن محمد شحاتة بدور**
**د./ يسرا محسن على القبانى**