



**DYNAMIC POLYMORPHIC
RECONFIGURATION TO EFFECTIVELY
“CLOAK” A CIRCUIT’S FUNCTION**

THESIS

Jeffrey L. Falkinburg, First Lieutenant, USAF
AFIT/GCE/ENG/11-03

**DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY**

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the United States Government. This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

AFIT/GCE/ENG/11-03

DYNAMIC POLYMORPHIC
RECONFIGURATION TO EFFECTIVELY
“*CLOAK*” A CIRCUIT’S FUNCTION

THESIS

Presented to the Faculty
Department of Electrical and Computer Engineering
Graduate School of Engineering and Management
Air Force Institute of Technology
Air University
Air Education and Training Command
in Partial Fulfillment of the Requirements for the
Degree of Master of Science in Computer Engineering

Jeffrey L. Falkinburg, B.S.C.E.

First Lieutenant, USAF

March 2011

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

DYNAMIC POLYMORPHIC
RECONFIGURATION TO EFFECTIVELY
"CLOAK" A CIRCUIT'S FUNCTION

Jeffrey L. Falkinburg, B.S.C.E.
First Lieutenant, USAF

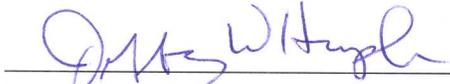
Approved:



Dr. Yong C. Kim
Chairman

28 Feb 2011

Date



Lt Col Jeffrey W. Humphries, PhD
Member

28 Feb 2011

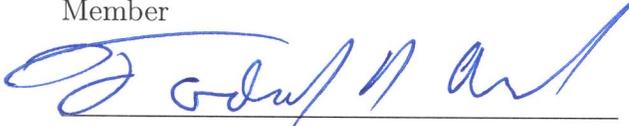
Date



Dr. Michael R. Grimaila
Member

28 FEB 2011

Date



Maj Todd R. Andel, PhD
Member

28 FEB 2011

Date

Abstract

Today's society has become more dependent on the integrity and protection of digital information used in daily transactions resulting in an ever increasing need for information security. Additionally, the need for faster and more secure cryptographic algorithms to provide this information security has become paramount. Hardware implementations of cryptographic algorithms provide the necessary increase in throughput, but at a cost of leaking critical information. Side Channel Analysis (SCA) attacks allow an attacker to exploit the regular and predictable power signatures leaked by cryptographic functions used in algorithms such as RSA. The RSA public key cryptographic algorithm is particularly vulnerable to SCA attack in the timing of the multiplication and squaring operations used in the modular exponentiation process. Hardware obfuscation is used to modify the circuit's hardware to intentionally conceal its functionality from an attacker. In this research the focus is on a means to counteract this vulnerability by creating a Critically Low Observable Anti-Tamper Keeping Circuit (CLOAK) capable of continuously changing the way it functions in both power consumption and timing. A Field-Programmable Gate Array (FPGA) based Encryption System testbed was developed to rapidly prototype and conduct SCA of protected and unprotected cryptographic algorithms. This research has determined that a polymorphic circuit design capable of varying circuit power consumption and timing can protect a cryptographic device from an Electromagnetic Analysis (EMA) attacks. In essence, we are effectively CLOAKing the circuit functions from an attacker.

Acknowledgements

First and foremost, I owe a large debt of gratitude to my wife and children for their continued support in my career and educational advancement. Without their support, none of this would have been possible. I would also like to thank my advisor, Dr. Yong Kim, for all of his guidance and support throughout this entire process. Without him I never would have been able to complete this thesis. I would also like to extend a sincerest thanks and appreciation to my committee members: Lt Col Jeffrey Humphries, Dr. Michael Grimaila, and Maj Todd Andel for their support and dedication to my research.

Jeffrey L. Falkinburg

Table of Contents

	Page
Abstract	iv
Acknowledgements	v
List of Figures	ix
List of Tables	xiv
List of Abbreviations	xvi
I. Introduction	1
1.1 Motivation	1
1.2 Problem Statement	4
1.3 Research Objectives and Contributions	4
1.3.1 Polymorphic Circuit Design	5
1.3.2 Implement RSA with Reconfiguration	5
1.3.3 Analyze Side Channel Signatures	6
1.4 Thesis Organization	6
II. Literature Review	7
2.1 Reconfigurable Computing	7
2.1.1 FPGAs	8
2.1.2 Run-time and Partial Reconfiguration	9
2.1.3 Polymorphic Reconfiguration	10
2.2 Cryptographic Algorithms	12
2.2.1 Key Generation and Protection	13
2.2.2 AES	15
2.2.3 RSA	16
2.3 Hardware Obfuscation	19
2.3.1 Security Through Obscurity	19
2.3.2 Authentication Based Obfuscation	20
2.4 Circuit Vulnerabilities	21
2.4.1 Reverse Engineering	21
2.4.2 Invasive Techniques	22
2.4.3 Semi-invasive Techniques	22
2.4.4 Side Channel Analysis Attacks	23
2.5 Power Analysis	23
2.5.1 Power Analysis Attacks	25
2.5.2 Power Analysis Countermeasures	26
2.6 Timing Analysis	27
2.6.1 Timing Analysis Attacks	28

	Page
2.6.2 Timing Attack Countermeasures	28
2.7 Electromagnetic Analysis	29
2.7.1 EMA Attacks	30
2.7.2 EMA Countermeasures	31
2.8 Literature Review Summary	31
III. Methodology of the Side Channel Analysis “ <i>CLOAK</i> ”	
Countermeasure	32
3.1 Problem Definition	32
3.1.1 Goals and Hypothesis	32
3.1.2 Research Approach	33
3.2 System Boundaries	36
3.2.1 RSA Encryption Algorithm	37
3.2.2 Xilinx Virtex-5 and Virtex-6 FPGAs	37
3.2.3 Processor Core	37
3.2.4 <i>CLOAK</i> Countermeasure	37
3.3 System Services	38
3.4 Workload Parameters	38
3.4.1 Message Offered Load	38
3.4.2 Random versus Static Message	39
3.4.3 Encryption Key Length	39
3.4.4 Polymorphic Key Length	39
3.5 Performance Metrics	40
3.5.1 FPGA Area Overhead for Polymorphic Logic	40
3.5.2 Modular Exponentiation Timing Delay Variation	40
3.5.3 Encryption Circuit Timing Delay/Latency	41
3.5.4 Number of Traces	41
3.6 System Parameters	42
3.6.1 Background Noise	42
3.6.2 Implementation of RSA	42
3.6.3 Polymorphic Frequency	42
3.6.4 Xilinx FPGAs	43
3.6.5 Processor Core	43
3.7 Factors	44
3.7.1 Polymorphic Frequency	44
3.7.2 FPGA Version for RSA	44
3.7.3 Random versus Static Message	45
3.8 Evaluation Technique	45
3.9 Experimental Design	46
3.10 Methodology Summary	46

	Page
IV. Results	48
4.1 Experimental Setup	48
4.2 Attack Methodology (Top vs. Bottom and Probe Type)	50
4.3 RSA Encryption Development	54
4.4 Polymorphic Circuit Development	58
4.4.1 Flatten Power Signature	58
4.4.2 Randomize Power Signature	61
4.4.3 Randomize Circuit Timing	64
4.5 Protected RSA SCA Results	69
4.6 Virtex-6 FPGA Investigations	72
4.6.1 Unprotected RSA SCA Results	74
4.6.2 Protected RSA SCA Results	76
4.7 Design Comparison	78
4.8 Results Summary	80
V. Conclusion	82
5.1 Completed Objectives	82
5.2 Conclusions	83
5.3 Contributions	84
5.4 Future Work	84
5.5 Summary	85
A. Encryption System Flowcharts	86
B. RSA Traces	89
2.1 Virtex-5 FPGA Traces	89
2.2 Virtex-6 FPGA Traces	93
C. Version Control	95
D. Data Sheets	98
4.1 Willtek 1207 Inductive Probe	98
4.2 Inspector EM Probe	98
4.3 Lecroy WavePro 725Zi Oscilloscope	98
4.4 Agilent E3631A DC Power Supply	98
4.5 Virtex-5 FPGA	99
4.6 Virtex-6 FPGA	99
Bibliography	100
Vita	106

List of Figures

Figure		Page
1	Polymorphic Gate Conversion	11
2	Polymorphic Switch	12
3	4-bit LFSR	14
4	AES - Symmetric-Key Cryptographic Algorithm	15
5	RSA - Public-Key Cryptographic Algorithm	16
6	Montgomery Modular Multiplier	19
7	Cross-Correlation of Modular Exponentiation on a Smart Card [46]	29
8	Development of the Polymorphic Circuit Design using Block Diagrams	34
9	Polymorphic <i>CLOAK</i> Encryption System	36
10	Cross Section for Virtex-5 Flip-Chip Package [11]	43
11	Experimental System Configuration	45
12	Experimental System Setup	48
13	RSA Encryption System Connections	49
14	Inspector console output for a round of RSA Encryption using sample key and modulus from [37]	50
15	Capacitors on the bottom of the Virtex-5 FPGA	51
16	Depackaging the Virtex-5 Chip	52
17	Comparison of EM signal strength using Willtek probe	52
18	Comparison of EM signal strength between probes	53
19	EM Inductive Probe on Virtex-5 FPGA	53
20	RSA Encryption System	54

Figure	Page
21	SEMA of Combined Square and Multiply Trace with key (5 8E B5) - RSA Version A 56
22	SEMA of Separate Square and Multiply Trace with key (E B5) - RSA Version B 56
23	Separate Square and Multiply Trace after signal processing - RSA Version B 57
24	Separate Square and Multiply Trace after signal processing using constant timing - RSA Version 2B 58
25	SEMA overlaid on Level Power Consumption Trace with key (E B5) - RSA Version C 59
26	SEMA overlaid on Level Power Consumption Trace with key (E B5) after signal processing - RSA Version C 60
27	Level Power Consumption Trace after signal processing using constant timing - RSA Version 2C 60
28	Level Power Consumption Trace after elastic alignment and average - RSA Version C 61
29	Polymorphic Modular Multiplier with Polymorphic Adder 62
30	SEMA overlaid on Randomized Power Consumption Trace with key (E B5) - RSA Version E 62
31	SEMA overlaid on Randomized Power Consumption Trace with key (E B5) after signal processing - RSA Version E 63
32	Randomized Power Consumption Trace after elastic alignment and average - RSA Version E 64
33	Pseudo-Random Timing Flow Chart 65
34	<i>CLOAK</i> Modular Multiplier 66
35	Randomized Power and Timing Trace - RSA Version F 67

Figure	Page
36	SEMA overlaid on Randomized Power and Timing Trace with key (E B5) after signal processing - RSA Version F 68
37	Randomized Power and Timing Trace after elastic alignment and average - RSA Version F 68
38	Comparison of EM signals for square and multiply operations after signal processing 69
39	Comparison of EM signals for square and multiply operations using identical inputs - RSA Version F 70
40	Comparison of square and multiply operations using identical inputs 71
41	Comparison of standard deviation of square and multiply operations using identical inputs 72
42	Comparison of standard deviation of square and multiply operations after frequency filtering 72
43	Comparison of standard deviation of 100 traces using identical inputs 73
44	Fan Assembly on Virtex-6 FPGA 74
45	EM Inductive Probe on Virtex-6 FPGA 74
46	Comparison of EM signals for Separate Square and Multiply Trace after signal processing using fixed plaintext on a Virtex-6 FPGA 75
47	Comparison of EM signals for Separate Square and Multiply Trace after signal processing using random plaintext on a Virtex-6 FPGA 76
48	Comparison of EM signals for Randomized Power and Timing Trace after signal processing using fixed plaintext on a Virtex-6 FPGA 77
49	Comparison of EM signals for Randomized Power and Timing Trace after signal processing using random plaintext on a Virtex-6 FPGA 77

Figure	Page
50	RSA Encryption System Flowchart on the Virtex-5 FPGA 86
51	Encryption System Flowchart on the Virtex-6 FPGA 87
52	AES Encryption System Flowchart on the Virtex-5 FPGA 88
53	Full 512-bit Combined Square and Multiply Trace - RSA Version A 89
54	Full 512-bit Separate Square and Multiply Trace - RSA Version B 89
55	Full 512-bit Separate Square and Multiply Trace using constant timing - RSA Version 2B 89
56	Full 512-bit Level Power Consumption Trace - RSA Version C 89
57	Full 512-bit Level Power Consumption Trace after signal processing - RSA Version C 90
58	Full 512-bit Level Power Consumption Trace using constant timing - RSA Version 2C 90
59	Full 512-bit Level Power Consumption Trace after elastic alignment and average - RSA Version C 90
60	Full 512-bit Randomized Power Consumption Trace - RSA Version E 90
61	Full 512-bit Randomized Power Consumption Trace after signal processing - RSA Version E 91
62	Full 512-bit Randomized Power Consumption Trace after elastic alignment and average - RSA Version E 91
63	Full 512-bit Randomized Power and Timing Trace - RSA Version F 91
64	Full 512-bit Randomized Power and Timing Trace after signal processing - RSA Version F 91

Figure	Page
65	Full 512-bit Randomized Power and Timing Trace after elastic alignment and average - RSA Version F 92
66	Comparison of EM signals for square and multiply operations 92
67	Full 512-bit Separate Square and Multiply Trace after signal processing using fixed plaintext - RSA Version B on a Virtex-6 FPGA 93
68	Full 512-bit Separate Square and Multiply Trace after signal processing using a different clock and fixed plaintext - RSA Version 4B on a Virtex-6 FPGA 93
69	Full 512-bit Separate Square and Multiply Trace after signal processing using random plaintext - RSA Version B on a Virtex-6 FPGA 93
70	Full 512-bit Separate Square and Multiply Trace after signal processing using a different clock and random plaintext - RSA Version 4B on a Virtex-6 FPGA 93
71	Full 512-bit Separate Square and Multiply Trace after signal processing using fixed plaintext - RSA Version F on a Virtex-6 FPGA 94
72	Full 512-bit Separate Square and Multiply Trace after signal processing using a different clock and fixed plaintext - RSA Version 4F on a Virtex-6 FPGA 94
73	Full 512-bit Separate Square and Multiply Trace after signal processing using random plaintext - RSA Version F on a Virtex-6 FPGA 94
74	Full 512-bit Separate Square and Multiply Trace after signal processing using a different clock and random plaintext - RSA Version 4F on a Virtex-6 FPGA 94

List of Tables

Table		Page
1	Factors and Levels	44
2	RSA Design Execution Time and Size for Virtex-5 FPGA	78
3	Comparison of RSA Design Execution Time	79
4	RSA Design Execution Time and Size for Virtex-6 FPGA	79

List of Algorithms

1	Key Generation for RSA Public-Key Cryptosystem [45], [57]	17
2	The RSA Algorithm [45]	17
3	Square-and-Multiply Algorithm for Modular Exponentiation	18

List of Abbreviations

Abbreviation		Page
USAF	United States Air Force	1
DoD	Department of Defense	1
MCT	Military Critical Technology	1
FISMA	Federal Information Security Management Act	1
JSF	Joint Strike Fighter	2
SCA	Side Channel Analysis	2
EMA	Electromagnetic Analysis	2
NIST	National Institute of Standards and Technology	3
FIPS	Federal Information Processing Standards	3
PIV	Personal Identification Verification	3
FPGA	Field-Programmable Gate Array	5
EM	Electromagnetic	5
ASIC	Applications Specific Integrated Circuit	7
JTAG	Joint Test Action Group	8
VHSIC	Very-High-Speed Integrated Circuit	8
VHDL	VHSIC Hardware Description Language	8
CLBs	Configurable Logic Blocks	8
LUT	Look-Up Table	8
PR	Partial Reconfiguration	9
AFIT	Air Force Institute of Technology	11
MUX	Multiplexer	11
PRNGs	Pseudorandom Number Generators	14

Abbreviation	Page
LSFR	Linear Feedback Shift Register 14
AES	Advanced Encryption Standard 15
RSA	Rivest, Shamir, and Adleman 16
LSB	Least Significant Bit 18
IP	Intellectual Property 19
FSM	Finite State Machine 20
PUF	Physically Uncloneable Functions 21
CMOS	Complementary Metal-Oxide Semiconductor 23
IC	Integrated Circuit 23
SPA	Simple Power Analysis 23
DPA	Differential Power Analysis 23
EC	Elliptic Curve 24
SASEBO	Side-channel Attack Standard Evaluation Board 25
DES	Data Encryption Standard 25
DDL	Dynamic Differential Logic 27
SEMA	Simple Electromagnetic Analysis 30
DEMA	Differential Electromagnetic Analysis 30
SUT	System Under Test 36
CES	<i>CLOAK</i> Encryption System 36
CUT	Component Under Test 36

DYNAMIC POLYMORPHIC
RECONFIGURATION TO EFFECTIVELY
“*CLOAK*” A CIRCUIT’S FUNCTION

I. Introduction

The proliferation of computing technology has brought with it an increased demand for information protection. Today much of our lives are digitized and stored on computer systems. These computer systems in one way or another handle the majority of transactions conducted on a daily basis. Society has become more and more dependent on the integrity and protection of the digital information used in these transactions causing an ever increasing need for digital security. The United States (U.S.) military, including the United States Air Force (USAF), relies on its military technological superiority to maintain our country’s military dominance. Operations conducted on Department of Defense (DoD) information networks are of particular importance to the United States Cyber Command (USCYBERCOM) and the USAF mission.

1.1 Motivation

The DoD considers any technology that makes a significant contribution to the military potential of our country to be Military Critical Technology (MCT) [1]. Compromise of MCT, such as cryptographic algorithms, can have disastrous consequences for the end user and security of the nation. The Federal Information Security Management Act (FISMA) of 2002 describes the importance of information security to national defense and the economic security of the nation [5]. Cryptographic algo-

rithms were adopted by Federal agencies and private industry to provide information security to protect information by providing integrity, confidentiality, and availability [5]. In other words, information security provides confidence in the authenticity of data, provides a way to keep information secret, and provides information in a timely and reliable manner. Cryptographic algorithms provide differing levels of security based on the time value and sensitivity requirements of the data being secured.

Threats to this information security include an adversary being able to hack into our live video intelligence feeds [20] or information systems and retrieve valuable information. Vital information about the 300 billion dollar Joint Strike Fighter (JSF) program was stolen through such means [34]. To execute a successful attack on a system, an adversary only has to compromise the weakest link in a system to gain access to critical technology. Side Channel Analysis (SCA) attacks on secure communications devices can be conducted without end users knowing their emissions are being exploited. These attacks only exemplify the need for more robust secure communication protocols.

Discovered by Paul Kocher et al. [39], power analysis attacks measure the power dissipation of a circuit when a transistor is switched from 0 to 1, and conversely 1 to 0. Power analysis is conducted by placing a meter between power and ground of the circuit to recover information regarding cryptographic operations being performed [39]. Timing analysis attacks exploit the slight differences in the amount of time required to perform encryption operations [38], [40]. These slight differences in encryption operations leak information about the secret key being used. Electromagnetic Analysis (EMA) attacks are similar to power analysis except they do not require tampering of the device under attack. EMA exploits the TEMPEST¹ (i.e., electromagnetic radiation) information leaked by electronic devices commonly called

¹TEMPEST is a codename referring to the intelligence-bearing electromagnetic radiation emanating from a circuit while in operation

compromising emanations [55].

The National Institute of Standards and Technology (NIST) published the Federal Information Processing Standards (FIPS) publication 201 pursuant to FISMA of 2002 which specifies the architecture and technical requirements as a common identification standard for Federal employees and contractors[6]. This FIPS publication specifies RSA[56] be used at a minimum for all Personal Identification Verification (PIV) cards. PIV cards are used throughout the DoD to identify users for facility access and secure communication.

Hardware protection systems seek to secure an electronic device that may be out of our physical control. The consumer is assumed to be a trusted user and is able to examine the technology at their leisure. In the case of the new iPhone 4G prototype, by Apple[®], being left in a bar in Redwood City, California, the trusted user lost positive control of the device. Eventually the prototype found its way into the hands of Gizmodo[®]. Apple was able to remotely disable and wipe the phone which removes the software vulnerability, but did not stop the hardware from being subsequently dissected and features documented for release to the public. The reverse engineering and release of proprietary phone features and capabilities happened months before Apple's big debut of the next generation iPhone. The consumer is assumed to be a trusted user especially in the case of a military only secure communications device, the potential still exists that the device be lost or stolen, and subsequently fall into the hands of a less than trustworthy user.

This research directly supports the USAF mission in the defense of the U.S. and its global interests through the safeguarding of MCTs so that weapon system capabilities stay out of adversarial hands. Attacks have been shown to exploit timing vulnerabilities in modern cryptographic systems enabling the recovery of the entire secret key [38]. Once the attacker compromises the secret key the security of the

entire system is compromised.

1.2 Problem Statement

Current public-key encryption systems do not provide adequate software or hardware protection against side channel analysis attacks. In the case of cryptographic circuitry, they are particularly susceptible to side channel and power analysis. The RSA [56] public-key cryptographic algorithm is particularly vulnerable to SCA attacks in the timing of multiplication and squaring operations used in modular exponentiation accomplished in the private-key operations. These timing attack vulnerabilities were first discussed by Paul Kocher [38] in 1996. There has since been ample documentation on the timing attack vulnerabilities affecting current public-key encryption algorithms. This check and balance of these public algorithms help to ensure their security by openly testing their limits and finding ways to increase their ability to defeat these vulnerabilities. While defensive countermeasures exist in modern designs, these do not address all methods used by malicious adversaries.

The key vulnerability to a side channel analysis attack in cryptographic algorithms like RSA is the predictable time required to calculate the modular exponentiation in the private-key operations. Timing attacks can be conducted by physically connecting a meter between power and ground on the chip or by non physical means to measure the TEMPEST information (compromising emanations) leaked by the circuit. What if we can build a dynamic circuit that can continuously reconfigure itself in order to successfully obfuscate its intent/purpose from an observer/adversary?

1.3 Research Objectives and Contributions

The desired outcome of this research is the creation and analysis of a Critically Low Observable Anti-Tamper Keeping Circuit (CLOAK) that is capable of obfuscating the

circuit function from side channel analysis by an observer/adversary. This research seeks the creation and analysis of a *CLOAK* countermeasure to defend against SCA attacks on a cryptographic algorithm in order to protect the secret key. This research is broken down into three different sections with an ultimate goal of protecting the secret key.

1.3.1 Polymorphic Circuit Design.

The key to providing a secure cryptographic system is the protection of the secret key from the adversary. This research leverages and extends previous work by Roy Porter [54] [53] and Camdon Cady [22] with polymorphic circuits capable of changing the way they function based on an input key. A polymorphic circuit design is developed and implemented on a Field-Programmable Gate Array (FPGA). This polymorphic circuit design is capable of continuously changing the way it functions to obfuscate its side channel signature in both power consumption and timing. This effectively creates the *CLOAK* countermeasure for hardware obfuscation. The Electromagnetic (EM) side channel of the circuit is then characterized to ensure proper functionality by implementing it into the modular exponentiation circuitry of a hardware implementation of RSA.

1.3.2 Implement RSA with Reconfiguration.

An implementation of RSA is designed and synthesized on an FPGA using a structural approach, which is a design based on the components and their interconnects. This approach gives the developer more control over the implementation of the modular exponentiation portion of the cryptographic algorithm. The first implementation uses an unprotected implementation of the modular exponentiation circuitry for baseline analysis. The second implementation uses the polymorphic circuit design

for the modular exponentiation circuitry.

1.3.3 Analyze Side Channel Signatures.

The EM side channel of the original implementation of RSA is characterized to set an initial baseline for subsequent analysis of the side channel. The baseline is compared to the modified RSA implementation to characterize the extent the circuit is able to resist timing attacks and protect the secret key. It is expected that the combination of elements in this approach culminate in a proof of concept polymorphic circuit design that enhances the system protective countermeasures by obfuscating operations.

1.4 Thesis Organization

The work presented in this thesis is organized into five main sections. Chapter 2 follows the introduction and provides background information on related research, to include reconfigurable computing, cryptographic algorithms, hardware obfuscation, circuit vulnerabilities, and three types of side channel analysis. Chapter 3 details the methodology of my approach to the implementation and evaluation of the stated objectives. Chapter 4 details the results of the experiments and any adjustments required to facilitate the stated objectives. Finally, Chapter 5 provides a summary of the relevance the work accomplished and the major contributions of this thesis. Additionally, it contains recommendations for future directions for extending this research.

II. Literature Review

The following chapter serves to orient the reader with a diverse range of topics used in the creation and protection of MCTs through hardware obfuscation. This chapter is structured as follows: Section 2.1 covers the strengths and weaknesses of reconfigurable computing giving particular emphasis on FPGAs and the motivation behind its use in this research. The next section, Section 2.2, covers some common cryptographic algorithms and key generation techniques for use in information security. Section 2.3 describes the concept of obfuscating hardware operations. Section 2.4 details four classes of circuit vulnerabilities. Finally, the power, timing, and EM analysis categories of SCA attacks are developed to include current attacks and countermeasures of each and how they fit in with this research. These categories are detailed in Sections 2.5, 2.6, and 2.7 respectively.

2.1 Reconfigurable Computing

Reconfigurable computing refers to the ability for a system to reprogram itself or redefine the way the circuit functions at run-time. This idea of reconfigurable computing has been in existence for almost half a century [32]. An Applications Specific Integrated Circuit (ASIC) is an integrated circuit that is designed for a specific design purpose and that design is set at the manufacturer. Any small change in an ASIC design would force a redesign and refabrication of the entire chip, which is an expensive and time consuming process [29]. FPGAs on the other hand are a reconfigurable systems capable of having their operations defined after the chip is manufactured. The primary platform for reconfigurable computing is FPGAs.

The primary advantages to using FPGAs over ASICs is due to the reconfigurable nature of the circuitry. The ability to define or redefine a circuit's purpose at the user

level or in the field can have great advantages especially if this chip is hardwired into a deployed system. There may be a need to modify a systems purpose or the way a system operates based on the needs of the user of the end system. The capability of obfuscating a circuit's operations from an adversary by changing how a system or component does its routine operations can have unique applications in information security. This advantage can come at the cost of performance since FPGAs still lag behind ASICs in clock speed and throughput [29].

2.1.1 FPGAs.

An FPGA is a two-dimensional array of reprogrammable gates and other assorted functional units combined together to carry out different logical functions. FPGAs can be reprogrammed by the customer multiple times by downloading a bitstream through the use of propriety software and a Joint Test Action Group (JTAG) interface. During this configuration process the user defines the Very-High-Speed Integrated Circuit (VHSIC) hardware circuit description using the VHSIC Hardware Description Language (VHDL). The proprietary Xilinx[®] or Altera[®] software then transforms the users VHDL circuit description specifications into a *bitstream* to be downloaded and programmed on the chip. Modern FPGAs use a combination of soft-core, embedded microcontroller, and Configurable Logic Blocks (CLBs) to provide a flexible platform for rapid system development.

While each manufacturer has a different nomenclature for the architectural elements of their FPGAs, the basic design is fairly standard. The terms used in this research are consistent with the Xilinx Virtex[®]-5 [15] and Virtex-6 [16] FPGAs. Each FPGA contains top-level logic elements/structures called CLBs. Each CLB contains two slices which contain equivalent circuitry such as function generators (i.e., Look-Up Table (LUT)), storage elements, arithmetic logic gates, multiplexers, and fast carry

look ahead chains. These CLBs are interconnected by a reconfigurable switch matrix for routing. Ultimately, all FPGA technology is based on LUT structures to give the user programmable logic function generators. LUTs are multiple input and single output structures that give a desired output based on specified input combinations. The Virtex-5 FPGAs is built using $65nm$ process technology and are implemented using 6-input LUT technology. The Virtex-6 FPGAs are also implemented using 6-input LUT technology, but are built on a smaller $40nm$ process technology to decrease power consumption.

Data encryption is a highly repetitive process where FPGAs and reconfigurable computing have been shown to increase performance of systems by leveraging hardware functions to execute these repetitive processes [29]. FPGAs enable rapid prototyping while exploiting the performance gains of using hardware functions. Typically to reconfigure an FPGA the user must take the system offline to download a new bitstream design.

2.1.2 Run-time and Partial Reconfiguration.

Reconfiguration in an FPGA can be executed in different ways. The simplest and most common way is to take the FPGA off-line while it is loaded with a new bitstream. The ability to reconfigure a circuit leads to an area of particular interest, the ability to reconfigure the system while it is still in operation. Altera defines run-time reconfiguration as the ability to modify or change the functional configuration of the device during operation, through hardware or software changes [7]. The concept of being able to change portions of a system's circuitry in run-time instead of taking the system offline to program the bitstream can have unique implications in information security.

Partial Reconfiguration (PR) for an FPGA is a Xilinx design flow that attempts to

create reconfiguration regions or zones in an FPGA device, so that one region can be reconfigured while the remainder of the FPGA continues to operate normally [7]. The ability to change only a portion of a design while leaving others portions untouched can be a powerful tool. This feature gives the designer the ability to perform repetitive processes on hardware to increase performance, while maintaining the flexibility of a software solution [29]. Stone et al., introduced a design where individual LUTs could be reconfigured while the FPGA was still in operation [61].

Stone's design has some promising applications, but comes with inherent limitations. The reconfiguration circuit is capable of operating at a higher frequency compared the rest of the FPGA, which minimizes downtime, but as the design gets larger the cost starts outweighing the benefit. This design allowed the design to be dynamically altered, but the routing cannot be changed. Additionally, each design has pre-compiled bitstreams that must be stored on the system. A solution that is somewhere in the middle would be optimal.

2.1.3 Polymorphic Reconfiguration.

Some polymorphic gate designs propose non-traditional control variables such as temperature or voltage to create polymorphic gates originally proposed by Stoica et al., [60]. These polymorphic gates were then combined to create polymorphic circuits [58]. These circuits are not very practical for rapid design and employment since the technology relies on a specific fabrication of the underlying gate structures. Also, the external conditions required to create these polymorphic changes in the circuits were not always easily controlled and could pose a serious threat to the standard operation of the circuit. Critical applications would require a more stable polymorphic design that is not as susceptible to random environmental impacts. In addition, the use of external conditions for directing the reconfiguration process also provides another

avenue of attack for adversaries.

Polymorphic reconfiguration, as defined by the Air Force Institute of Technology (AFIT), is a combination of polymorphic gates or decision circuits to create a dynamically changing circuit design that is capable of changing the way it functions based on an input key. A basic example of a polymorphic gate is created by replacing any 2-input gate with a 4 to 1 Multiplexer (MUX) as defined by [22]. This can be accomplished by connecting the gate inputs to the MUX input select lines and the additional input lines of the MUX are driven by the truth table of the gate function you wish to create. These additional inputs can be driven to a specific level manually or driven by external key generation circuitry causing the polymorphic gate to function like a combinational lock to unlock the gates function. An example of converting an AND gate into a polymorphic AND gate is shown in Figure 1.

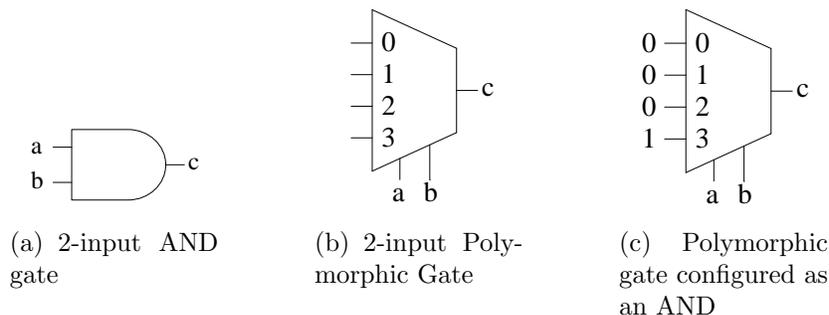


Figure 1. Polymorphic Gate Conversion

Other gates like polymorphic switches may also be used to create polymorphic circuits. A polymorphic switch is a 2-input, 2-output switch that uses a single control line to drive the input/output selection for the gate. An example of the polymorphic switch and its operation is shown in Figure 2. All these techniques can all be used to enhance the area of information security and at the heart of information security are the cryptographic algorithm that protect the information.

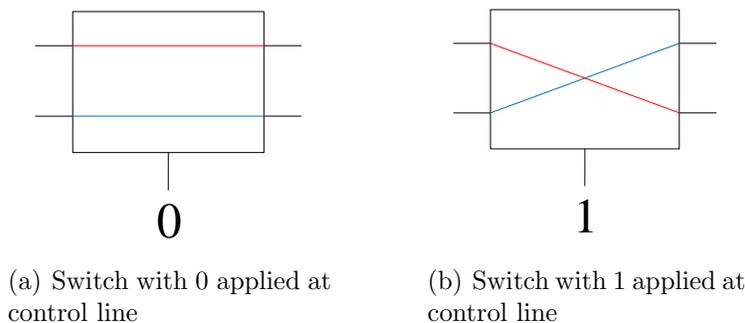


Figure 2. Polymorphic Switch

2.2 Cryptographic Algorithms

Cryptographic algorithms today are primarily used to protect the transmission and storage of information in digital form. All cryptographic algorithms in one way or another attempt to take the plaintext¹ and scramble it to create ciphertext² in a process called encryption or enciphering. The reverse is the act of recovering the plaintext from the ciphertext, called decryption or deciphering.

Information security is essential to not only national defense, but also the economic security of the nation. Many of the transactions conducted online require the use of public-key encryption and digital signature schemes. Federal employees and contractors throughout the DoD use PIV cards as a means of identification of users for facility access and for the use in secure communication. These cryptographic systems use public-key cryptographic algorithms with a set of encryption transformations and a set of decryption transformations, as in RSA, to ensure information security. Every type of cryptographic system requires some sort of private or secret key to enable the security of the system. In a symmetric-key cryptographic algorithm the encryption and decryption keys are the same and are known by both sender and receiver, but in a public-key algorithm the encryption and decryption keys are different. The public

¹The Plaintext is the original message to be sent or received.

²The Ciphertext is the scrambled or hidden message that may be transmitted over non-secure channels.

key is used for encryption while a secret key is used for decryption. In the next few sections, some methods of generating and protecting the key are discussed, followed by a popular symmetric-key algorithm (AES), and lastly one of the most commonly used public-key cryptographic algorithms (RSA) is discussed.

2.2.1 Key Generation and Protection.

All modern cryptographic algorithms rely on the use and protection of a secret key regardless of the type of cryptosystem. Encryption/decryption systems typically fall into two different categories: symmetric-key and public-key. Both algorithms require a secret key. In a symmetric-key cryptosystem it is the secret key and in a public-key cryptosystem it is the private key. If this key is compromised, the security of the entire system is compromised, regardless of the particular algorithm used [57]. If one could analyze a system and retrieve the secret key the security of the system would be eliminated.

One of the advantages of using a public-key cryptosystem like RSA over a symmetric-key cryptosystem is that a secure channel is not needed to distribute keys [57]. It is much easier to use a public-key cryptosystem since the public-key can be freely distributed and does not need to be kept secret. In a symmetric-key cryptosystem the key must not only be generated, but then it has to be distributed to all parties before secure communication can commence. Key generation requires a key of sufficient size and be “random” in the sense that the probability of any particular value being selected must be sufficiently small [45].

This section discusses two general techniques for key generation: random and pseudorandom. “In the classical (Kolmogorov) sense, a string of bits is random if it cannot be described by a shorter string than itself” [42]. A random bit generator can be used to generate a uniformly distributed random number that would force

an adversary to guess 2^n possible keys, where n is the number of bits in the key. A truly random key is difficult to produce, but would provide the best protection in a cryptosystem. However, generation of true random bits is an inefficient procedure in most practical environments [45].

Pseudorandom Number Generators (PRNGs) are typically used in cryptographic systems in the key generation process. If the appropriate PRNG is selected it can provide adequate protection in the creation of pseudo random seeds. PRNGs often use a Linear Feedback Shift Register (LSFR), shown in Figure 3, to create the pseudo-random bit stream. A LSFR is basically a shift register that has its input connected

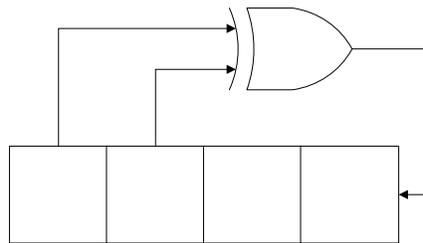


Figure 3. 4-bit LFSR

to some combination of bits already contained within the register. A simple example of a LFSR would be to have some of the register bits XORed together to form the input bit into the shift register. A LSFR can be described by an associated feedback polynomial that shows which coefficients are tapped to determine the input bit. Using the recommended tap locations in [3] to achieve a maximum-length LFSR counter for a 4-Bit LSFR there is $2^n - 1 = 2^4 - 1 = 15$ possible different numbers before repeating. This gives us a feedback polynomial shown in Equation 1.

$$x^4 + x^3 + 1 \tag{1}$$

2.2.2 AES.

The Advanced Encryption Standard (AES) is a symmetric-key cryptographic algorithm. A symmetric-key cryptographic algorithm uses the same key to encrypt and decrypt a message. This concept is shown in Figure 4.

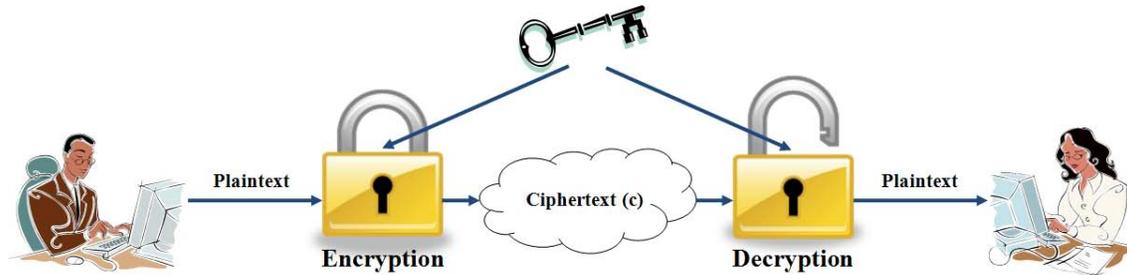


Figure 4. AES - Symmetric-Key Cryptographic Algorithm

AES is currently approved by the NIST to be used by Federal agencies to provide cryptographic protection for sensitive (unclassified) information. AES provides 128 bit block cipher that is capable of using 128, 192, or 256 bit keys to encrypt and decrypt [4]. If the circuit components that implement these functions can be identified, the key can be recovered. The AES algorithm bases its security on the ability to diffuse and confuse the message and key together to generate the ciphertext. The five functions used in AES are:

- KeyExpansion - This routine is used to generate a series of Round Keys from the Cipher Key. It is the first function to be performed when an encryption or decryption operation begins. It takes as an input the 128-, 192-, or 256-bit AES key and generates 10, 12, or 14 round keys depending on the key length.
- SubBytes - This is a transformation function that takes a 4×4 State³ array of bytes and uses a non-linear substitution table (S-Box) that operates on each of the state bytes independently.

³The State is a 4×4 array containing the intermediate state of the cipher.

- ShiftRows - This is a transformation function that processes cyclically shifts on the last three rows of a 4×4 State array of bytes . The second row is shifted to the left one, the third row is shifted left by two, and the fourth row is shifted left by three.
- MixColumns - This is a transformation function that takes all the columns of a 4×4 State array of bytes and mixes their data by multiplying each column of the array by a fixed polynomial.
- AddRoundKey - This is a transformation function that adds a Round Key to the 4×4 State array of bytes using an XOR operation. The length of the Round Key equals the size of the State.

2.2.3 RSA.

Rivest, Shamir, and Adleman (RSA), named after its inventors, is a public-key cryptographic algorithm that uses two keys and a modulus function to encrypt and decrypt messages. RSA is the most widely used a public-key cryptosystem [45] for information security, but can also be used for digital signatures. A pictorial representation of the RSA public-key cryptographic algorithm is shown in Figure 5. In

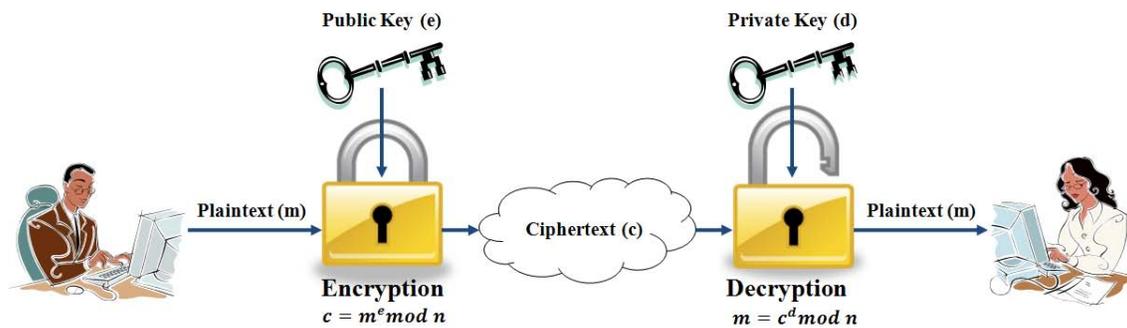


Figure 5. RSA - Public-Key Cryptographic Algorithm

RSA, two keys are generated e and d called the *encryption exponent* and the *decryp-*

tion exponent using the modulus n . One key is kept private (secret) while the other is made public. RSA's security is based on the intractability of factoring very large integers, which is closely related to the discrete logarithm problem. The key generation algorithm is illustrated in pseudocode below in Algorithm 1. Key generation is important for RSA because the security of the algorithm is based completely on the "hardness" of factoring a product of two large prime numbers.

Algorithm 1 Key Generation for RSA Public-Key Cryptosystem[45][57]

SUMMARY: each entity creates an RSA public key and a corresponding private key. Each entity A should do the following:

1. Generate two large random (and distinct) primes p and q , each roughly the same size.
 2. Compute $n = pq$ and $\phi(n) = (p - 1)(q - 1)$
 3. Select a random integer $e, 1 < e < \phi$, such that $\gcd(e, \phi(n)) = 1, f(p, q)$ (i.e., e and $\phi(n)$ share no divisors other than 1).
 4. Use the extended Euclidean algorithm to compute the unique integer $d, 1 < d < \phi(n)$, such that $ed \equiv 1 \pmod{\phi(n)}$
 5. A's public key is (n, e) ; A's private key is d .
-

A brief description of the RSA public-key cryptographic algorithm is presented in Algorithm 2. RSA is based on a mathematical function to generate the ciphertext.

Algorithm 2 The RSA Algorithm [45]

SUMMARY: B encrypts a message m for A , which A decrypts.

1. *Encryption.* B should do the following:
 - (a) Obtain A 's authentic public key (n, e) .
 - (b) Represent the message as an integer m in the interval $[0, n - 1]$.
 - (c) Compute $c = m^e \pmod n$ (e.g., using Algorithm 3).
 - (d) Send the ciphertext c to A .
 2. *Decryption* To recover the plaintext m from c , A should do the following:
 - (a) Use the private key d to recover $m = c^d \pmod n$.
-

The RSA discussed in this research uses an implementation based on modular exponentiation. Modular exponentiation is a fairly simple Square-and-Multiply Al-

gorithm shown in Algorithm 3, requiring repetition to implement. The Right-to-Left design is described in Algorithm 3. This repetition makes the operation easy to implement, but it also makes RSA particularly vulnerable to SCA attacks.

Algorithm 3 Square-and-Multiply Algorithm for Modular Exponentiation

SUMMARY: Algorithm for calculating plaintext message from ciphertext ($m = c^d \text{ mod } n$) in RSA. Where d_i is exponent bits ($0 \leq i < t$) whose binary representation is $d = \sum_{i=0}^t d_i 2^i$

```

Set  $M \leftarrow 1$  and  $C \leftarrow c$ 
if  $d = 0$  then
    Return ( $M$ )
end if
for  $i = 0$  up to  $t$  do
    if  $d_i = 1$  then
         $M = M * C \text{ mod } n$  (Multiply Operation)
    end if
     $C = C * C \text{ mod } n$  (Square Operation)
end for
Return ( $M$ )

```

The key component in modular exponentiation is the modular multiplication unit. Figure 6 shows a graphical representation of a Montgomery Modular Multiplication unit used in this research. This multiplier uses a series of shift and add operations followed by modulo reductions. The multiplicand is bitwise shifted to the left and added to the final product based on the Least Significant Bit (LSB) of the multiplier as it is bitwise shifted to the right. After each shift and add operation the modulus is subtracted from the sum either 0, 1, or 2 times to get the final product.

The basis for any RSA implementation is modular exponentiation. Given a modular multiplier instantiation the modular exponentiation operation becomes a simple repetitive process of squares and multiplies. However, this simplicity also makes the RSA software and hardware very vulnerable to attacks, particularly to SCA attacks.

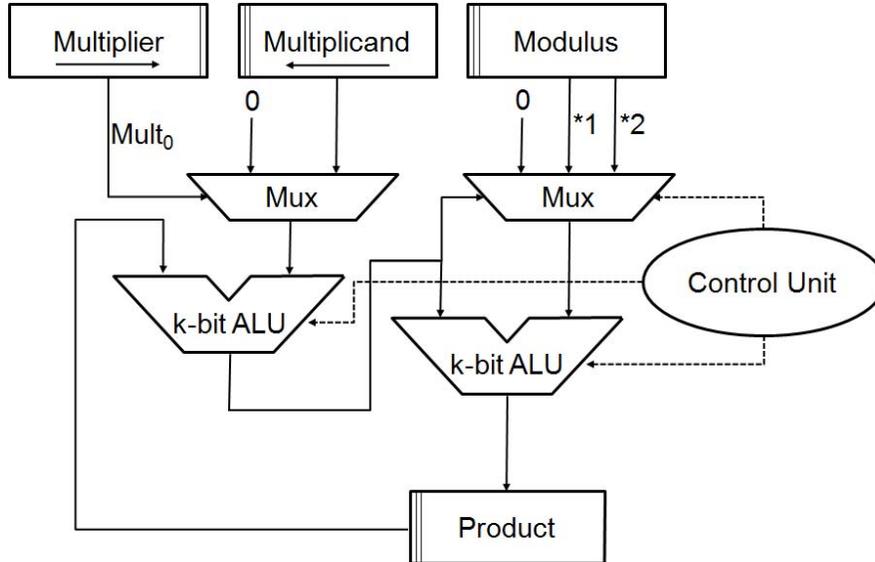


Figure 6. Montgomery Modular Multiplier

2.3 Hardware Obfuscation

Obfuscation is typically thought of as a technique that transforms a design to a functionally equivalent design, but resists reverse engineering [28], [66]. Hardware obfuscation is the modification of circuit hardware to intentionally conceal its functionality from an observer or adversary. Hardware obfuscation is conducted in an attempt to protect Intellectual Property (IP) from being stolen, or in the case of cryptographic systems protection of data being passed between systems [25]. There are two main categories to classify hardware obfuscation used as a part of this research: security through obscurity and authentication based obfuscation.

2.3.1 Security Through Obscurity.

The idea behind security through obscurity involves changing the white-box characteristics or underlying gate structure and signals of the system. Hardware obfuscation that creates security through obscurity can be executed several different ways. The first technique involves netlist obfuscation [24] while still maintaining the original

circuit semantics. This does not defend the circuit from being stolen and used as a “black-box” system, but it makes it more difficult for an attacker to determine and modify the circuit functions. The process of netlist obfuscation changes depending upon the technique, but typically involves some sort of smart selection and replacement of gates to create new paths within the design.

The second technique for obfuscation involves modifying the circuit hardware by adding extra circuitry at the design level to intentionally obfuscate white-box circuit functions. This can be accomplished dynamically by reprogramming without halting operations [61]. Alternately, a circuit may be fabricated with gates that change the logical flow of the Finite State Machine (FSM) and create multiple paths for circuit operation. This allows the circuit to dynamically change the way it operates in runtime. What if you use this logic to enable/disable circuit operations based on a key?

2.3.2 Authentication Based Obfuscation.

Authentication based obfuscation can be accomplished by inserting a FSM into the control logic of the function causing the system to only function properly if provided the proper key or authentication protocols. The authentication protocols require some sort of cryptographic algorithm and digital signatures embedded on the system. The system will only operate properly if a proper challenge and response is given. These digital signatures are only known to the creator of the IP core. Each system can potentially have a unique authentication signature, which would require the storage and control of these signatures.

The idea of having a hardware specific identifier that would not have to be centrally controlled led to the development of hardware signatures (digital fingerprints) that are tied to a specific piece of hardware. If a circuit has this type of functionality and the hardware is somehow stolen/cloned and placed on similar hardware it will

not operate properly. These hardware signatures stemmed from research in Digital Fingerprinting [30], [41], [52] and Physically Uncloneable Functions (PUF) [33], [62]. Both are similar in nature and involve the use of circuitry that utilizes subtle defects in the manufacturing process which cause two seemingly identical circuits to operate remarkably differently. These subtle differences can cause slightly different delays in the circuitry causing glitches that can be counted to create a hardware based key.

2.4 Circuit Vulnerabilities

MCTs and proprietary information systems need to be secured from potential attack. There are several classes of attacks that can be used against both ASICs and FPGAs alike. Four classes of attacks are discussed here: Reverse Engineering, Invasive Techniques, Semi-Invasive Techniques, and Side Channel Analysis Attacks. The following vulnerabilities stem from the common circuit structures used to create FPGAs are also common to ASICs.

2.4.1 Reverse Engineering.

The reverse engineering process is a process of analyzing a circuit to identify the individual components within a system and how they interact with each other in order to identify the circuit's intent. The ultimate goal of reverse engineering is to visualize the relationships between system components [27]. There are two classifications of reverse engineering black-box analysis and white-box analysis. The two methods correspond to the amount of information that is known about the circuit.

In black-box analysis little is known about the circuit structure. This method of analysis relies on the input/output relationship (i.e., the truth table) of the circuit to identify it. Inputs are applied to a circuit and the outputs are recorded to enumerate the truth table and identify the circuit function. For smaller circuits, the truth table

of a component can be compared to a library directly. For larger circuits, it is not possible to even enumerate the truth table let alone compare it to a library [54].

White-box analysis assumes that some information about the circuit structure is known either through a circuit netlist or imagery. Given a gate-level description of a circuit it is possible to identify components within a circuit design [64]. Libraries of optimized components are used to reduce the need to develop entire systems from scratch. These modern synthesis tools use common components to carry out specific operations while designers can focus on other areas of development. White box analysis exploits the use of these common components and hierarchical design methodologies.

2.4.2 Invasive Techniques.

While monitoring a circuit to gain knowledge about it has been somewhat passive in nature, sometimes other techniques may be required to discover more about the device. If access to the target device is unrestricted then the device may be physically tampered with. Invasive techniques require de-packaging components by removing the protective packaging around the circuit under investigation. These attacks require specialized tools to de-package and map the chips components. Even though these techniques have not been demonstrated against FPGA technology directly, the SRAM technology that FPGAs are built on have been shown to be susceptible to these types of attacks [65].

2.4.3 Semi-invasive Techniques.

Semi-invasive techniques are similar to invasive techniques in that they must also have the outer packaging removed, but require less effort and expense to execute. Semi-invasive techniques are categorized in the area of fault-injections. Fault injection using low cost items like a camera flash or laser pointer has been shown to temporarily

change the state of individual Complementary Metal-Oxide Semiconductor (CMOS) transistors [59]. This method exploits the fact that the CMOS transistor is susceptible to ionizing radiation. An attacker can use semi-invasive techniques to flip a bit in a cryptographic circuit allowing the system to be cracked [19].

2.4.4 Side Channel Analysis Attacks.

SCA attacks are used to non-invasively tamper with an Integrated Circuit (IC) and attempt to retrieve the secret key by analyzing the electrical emissions leaked from the circuits switching activity during normal operation. These attacks typically exploit information leaked from three different side channels: power consumption [40], timing delays [38], and EM emission [18]. The EM side channel remains the most viable avenue of attacking cryptographic devices when the power side channel is unavailable [18]. For the remainder of this research SCA attacks will be classified in the three main categories for development in Sections 2.5, 2.6, and 2.7:

- Power Analysis
- Timing Analysis
- Electromagnetic Analysis

2.5 Power Analysis

Power analysis attacks were discovered by Kocher [39] and measure the power dissipation of a circuit when a transistor is switched from 0 to 1, and conversely 1 to 0. Power analysis is conducted by placing a meter across a resistor between power and ground of the circuit to recover information regarding cryptographic operations being performed [39]. Power Analysis is divided into two techniques: Simple Power Analysis (SPA) and Differential Power Analysis (DPA). Both techniques are based on

measuring the amount of work the system is accomplishing based on the data being manipulated.

Simple Power Analysis (SPA): SPA involves directly interpreting power signals collected on cryptographic circuits operations. SPA can yield information about the cryptographic circuit's operations being conducted and secret key [39]. SPA is conducted using only a single power trace to extract the secret key based on conditional logic used in the cryptographic operations performed.

Differential Power Analysis (DPA): DPA involves the statistical analysis of the power signature to yield the entire key or partial information about the key. This process is designed to recover enough information about cryptographic circuits to reconstruct the secret key. DPA uses small-scale power effects based on the data being manipulated during cryptographic operations [39]. These analysis techniques use a random plaintext and a constant key to correlate the traces to the data being manipulated.

The first experimental results of power analysis attacks on FPGAs appeared in [51] where Örs et al. investigated the side channel of a hardware implementation of Elliptic Curve (EC) point multiplications. They conducted SCA to attack a Montgomery modular multiplication circuit on the Xilinx Virtex-800 FPGA by measuring the chip power supply directly. EC point multiplication is implemented using a simple double-and-add method similar to the multiply-and-square operations conducted in modular exponentiation (ref. Algorithm 3), but have significantly different side channel signatures. Wang et al., proposed an algorithm variant of Kim's Countermeasure [43], that could be resistant to DPA on a hardware implementation of RSA using a Montgomery modular multiplier[63]. Their design used a form of blinding⁴ within the

⁴Blinding involves a process of multiplying operands by a random number

modular operations to randomize the power signatures. Their results were all based on Synopsys[®]VHDL simulations and not validated on hardware.

FPGAs provide an optimal platform to design and test the SCA vulnerabilities of cryptographic systems prior to deployment. Currently there are several FPGA platforms that enable a user to conduct SCA attacks on RSA and other cryptographic algorithms. One such system used by Miyamoto et al., the Side-channel Attack Standard Evaluation Board (SASEBO) [47] [48] [49], is specifically design to exploit the power side channel. These commercially available systems rarely give the user full access to the source code to conduct trade studies or to create an optimal design to meet the security and performance needs of the user. In addition, these systems require the purchase of specialized hardware to operate. The creation of a hardware non-specific design capable of being implemented on any standard FPGA platform would be a perfect platform to conduct trade studies on power analysis attacks and countermeasures available.

2.5.1 Power Analysis Attacks.

2.5.1.1 Simple Power Analysis Attacks.

Kocher introduced SPA by implementing a SPA attack against the Data Encryption Standard (DES) [39]. Kocher analyzed the timing of branching operations to correlate power signatures directly to cryptographic operations being performed therefore revealing the secret key [39]. SPA attacks rely on the secret key being used in conditional logic operations which cause branching conditions based on the bit values of the key.

2.5.1.2 Differential Power Analysis Attacks.

Correlation Coefficient Attacks: DPA relies heavily on a statistical analysis to conduct attacks. DPA commonly uses the correlation coefficient⁵ to determine a linear relationship between data. The most common models used with correlation coefficient attacks are Hamming-Distance and Hamming-Weight Models.

Hamming-Distance and Hamming-Weight Models: The Hamming-Distance model uses the power signature of cryptographic operations to correlate the observed transitions of bit values within a register or on a data line. The Hamming-Weight model correlates the operands based on the Hamming Weight⁶ of the operands.

2.5.2 Power Analysis Countermeasures.

2.5.2.1 Simple Power Analysis Countermeasures.

SPA countermeasures are implemented simply by avoiding the use of procedures that use the secret key in conditional logic within cryptographic circuitry.

2.5.2.2 Differential Power Analysis Countermeasures.

Countermeasures to DPA have been researched since the inception of DPA. There are countless countermeasures to defend against DPA and they are presented in open literature. Yet, no countermeasure currently exists that completely protects a circuit from DPA. There are several classes of DPA countermeasures that are common throughout the community:

Signal Leakage Reduction: This class of countermeasures involve signal reduc-

⁵The Correlation Coefficient is a guess at the covariance between two random variables.

⁶The Hamming Weight of a binary number represents the number of 1 bits within the data sequence.

tion. This is accomplished by either balancing the hamming weight of the data or reducing the secret key dependence on conditional logic functions.

Noise Induction: This class of countermeasures involves inducing noise generators, such as oscillators, into the power consumption to effectively mask the original circuit's power. These added signals increase the number of signals required to conduct an attack. Ultimately we would like to increase the required samples to an unmanageable size.

Timing Randomization: This class of countermeasure involves randomizing circuit execution timing. This class is similar to a technique implemented by Kocher [39]. Timing randomization is accomplished by inserting random delays or clock modifications [44].

Kaps et al., proposed a countermeasure for DPA attacks on FPGA based AES implementations by using Dynamic Differential Logic (DDL) equalizing power consumption to increase the number of traces (i.e. encryption cycles) needed [36]. This technique worked, but at a large area penalty.

2.6 Timing Analysis

Timing analysis attacks exploit the slight differences in the amount of time required to perform encryption/decryption operations [38], [40]. These slight differences in encryption operations can leak information about the secret key being used. By carefully measuring the time required for a cryptographic system to conduct private key operations in algorithms like Diffie-Hellman and RSA, attackers are able to extract the private key [40].

2.6.1 Timing Analysis Attacks.

These algorithms are particularly vulnerable in the modular exponentiation portion of the cryptographic algorithms (see Algorithm 3). In modular exponentiation each bit of the exponent (private key) bits d_i are used to determine whether a square operation ($d_i = 0$), or a square and multiply operations ($d_i = 1$) will be performed. The goal of an attacker would be to monitor the loop iterations of the execution cycle timing to extrapolate the private key. Circuits are especially vulnerable when operations are conducted in a serial manner such as in software implementations of cryptographic systems because they typically execute one operation at a time. These attacks require a working knowledge of how the system is implemented in order to conduct a successful attack.

A more practical method of attack was detailed in [31] that does not require as much knowledge of the system as those presented by Kocher. This attack used a Montgomery multiplier unit that used a constant timing regardless of input factors plus a reduction phase for the modulus. This allowed them to create templates of the different plaintext input combinations that did and did not require reductions for each of the loops.

Messerges et al. [46] conducted SPA on implementation of modular exponentiation on a smart card. Using this implementation Messerges was unable to cross-correlate differences in the square and multiply power signatures, but was able to identify a slightly different execution time needed for each operation leading to a combined power and timing attack. The results of the attack are shown in Figure 7.

2.6.2 Timing Attack Countermeasures.

Kocher suggested that an obvious countermeasure to timing attacks is to force all operations to take the same amount of time [40]. However forcing software imple-

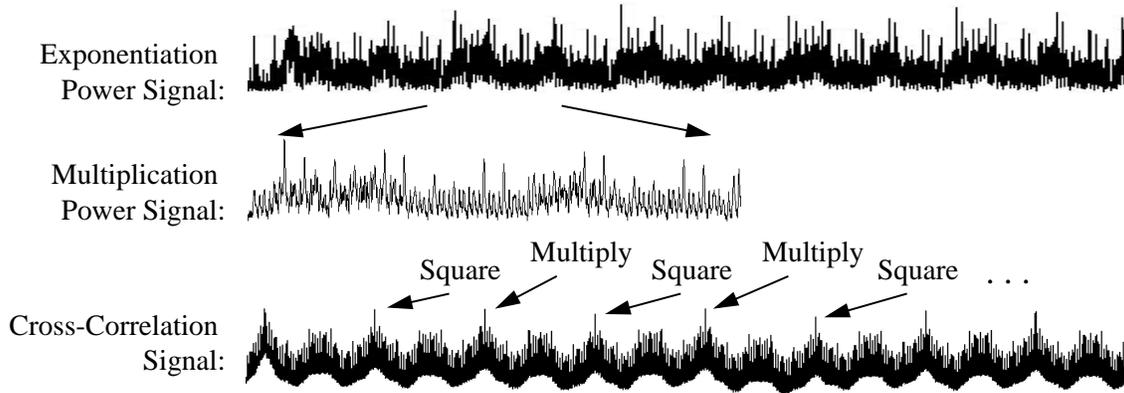


Figure 7. Cross-Correlation of Modular Exponentiation on a Smart Card [46]

mentations to run in constant time is difficult in a platform-independent environment because of compiler options [40]. Forcing a constant operating time can be easier to implement using hardware, but then the circuit will still leak information on the power channel based on switching activity.

Kocher [38] proposed that techniques for blinding signatures [26] could be adapted to Diffie-Hellman and RSA to prevent modular operations from being directly related to the secret key. Blinding the exponent can be used to defend against SCA attack and can be applied to either the message and/or exponent. For RSA, Kocher proposed that a random pair (v_f, v_i) could be added to the input message before the modular exponentiation operations. You first choose a random v_f relatively prime to n and $v_i = (v_f^{-1})^e \bmod n$. The input message is then multiplied by $v_i \bmod n$, and the result is then recovered by multiplying $v_f \bmod n$.

2.7 Electromagnetic Analysis

Electromagnetic Analysis (EMA) attacks are similar to power analysis except it does not require physical tampering of the device under attack. EMA exploits the TEMPEST (i.e., EM radiation) information leaked by electronic devices commonly called compromising emanations [55]. By using technology like inductive probes, the

adversary uses passive analysis to collect circuit power information. The EM side channel was proposed not long after power analysis and later was demonstrated on smart cards by Quisquater and Samyden [55]. Just like power analysis, there are two techniques for EMA introduced by Quisquater: Simple Electromagnetic Analysis (SEMA) and Differential Electromagnetic Analysis (DEMA).

Simple Electromagnetic Analysis (SEMA): This type of analysis involves the same processes as SPA except that it uses a different side channel to execute. A single trace is collected on a cryptographic operation to extract the secret key.

Differential Electromagnetic Analysis (DEMA): This type of analysis involves the same processes as DPA. Many traces are collected using random plaintext input and a constant key. Traces are then correlated and analyzed to extract the secret key.

EMA was first conducted on smart cards by Quisquater using a small coil of wire connected to an oscilloscope to measure the current [55]. Carrier et al., demonstrated that the secret key could be revealed by using DEMA on a FPGA implementation of AES. Carrier also found that the EM side channel emanations of an FPGA are of the same nature as those from a smart card [23]. They attacked the hardware implementation of AES successfully showing that the EM side channels of the FPGA can be exploited by an attacker to retrieve information about the secret key. Mulder et al., conducted SEMA and DEMA attack on an FPGA implementation of an Elliptic Curve Cryptosystem [50] executing EC point multiplication.

2.7.1 EMA Attacks.

Attacks conducted on cryptographic circuits using EMA are conducted in the same manner as power analysis regardless of whether you are conducting SEMA or DEMA. The only difference is the method of collection.

2.7.2 EMA Countermeasures.

The first commonplace countermeasure for EMA is the use of additional metal layers to reduce the electromagnetic fields on the chip [55]. Another countermeasure to EMA is to design the circuit for lower power consumption. Noise generators, such as oscillators, can be easily added to the circuitry to generate additional signals in an attempt to make EMA more difficult.

2.8 Literature Review Summary

Reconfigurable hardware simplifies and reduces the cost of the design process by allowing the customer to rapidly iterate through prototype designs. FPGAs have been found to be vulnerable to the same methods of attack as ASIC circuitry, but one benefit FPGAs have is the ability to test possible design iterations before locking in a final design. The AES and RSA cryptographic algorithms were presented as a basis for cryptographic circuits. Finally, several common side channel analysis attacks were presented along with possible countermeasures designed increase the required traces to an unmanageable size.

III. Methodology of the Side Channel Analysis “*CLOAK*”

Countermeasure

This chapter describes the methodology used to research/create a flexible dynamic protection for encryption circuits against Side Channel Analysis (SCA) attacks by creating the Critically Low Observable Anti-Tamper Keeping Circuit (CLOAK) countermeasure.

3.1 Problem Definition

Current public key encryption systems do not have adequate protections against side channel analysis attacks. Cryptographic circuits are particularly susceptible to EM and power SCA attacks. RSA [56] public key cryptographic algorithm is especially vulnerable to SCA attacks by timing of multiplication and squaring operations used during modular exponentiation of the private-key operations. Timing attack vulnerabilities affect many current public key encryption algorithms [38]. While static defensive countermeasures have been incorporated in more recent designs, they do not address all the techniques known to be used.

The key vulnerability a SCA attack exploits in cryptographic algorithms like RSA is the predictable time required to calculate the private key operation using modular exponentiation. Timing attacks can be conducted by physically connecting a meter between power and ground on the chip or by non physical means to measure the EM radiation leaked by the circuit.

3.1.1 Goals and Hypothesis.

The goal of this research is to determine whether a polymorphic circuit can protect a device from an EM timing analysis attack. The hypothesis of this research is that

the EM signature of a cryptographic algorithm can be varied in such a way that the observer/adversary cannot correlate side channel signature to the cryptographic functions being executed. The approach to accomplish this is discussed in the following sections.

3.1.2 Research Approach.

The desired outcome of this research is the creation and analysis of a Critically Low Observable Anti-Tamper Keeping Circuit (CLOAK) countermeasure that provides flexible dynamic protection for encryption circuits against SCA attacks. This research is accomplished in three parts the polymorphic circuit design, implementation of RSA with reconfiguration, and analysis of the side channel signatures all with the ultimate goal of protecting the secret key.

3.1.2.1 Polymorphic Circuit Design.

The key to providing a secure cryptographic system is the protection of the secret key from the adversary. This research leverages and extends previous work by Porter [54] [53] and Cady [22] by using polymorphic circuits. Polymorphic reconfiguration is a combination of polymorphic gates or decision circuits to create a dynamically changing circuit design that is capable of changing the way they function based on an input key. A polymorphic circuit design is developed and implemented on a FPGA. This polymorphic circuit design is capable of continuously changing the way it functions to obfuscate its side channel signature in both power consumption and timing. This effectively creates the *CLOAK* countermeasure for hardware obfuscation. The EM side channel of the circuit is then characterized to insure proper functionality by implementing it into the modular exponentiation circuitry of a hardware implementation of RSA. The entire polymorphic circuit development process using block

diagrams is shown in Figure 8.

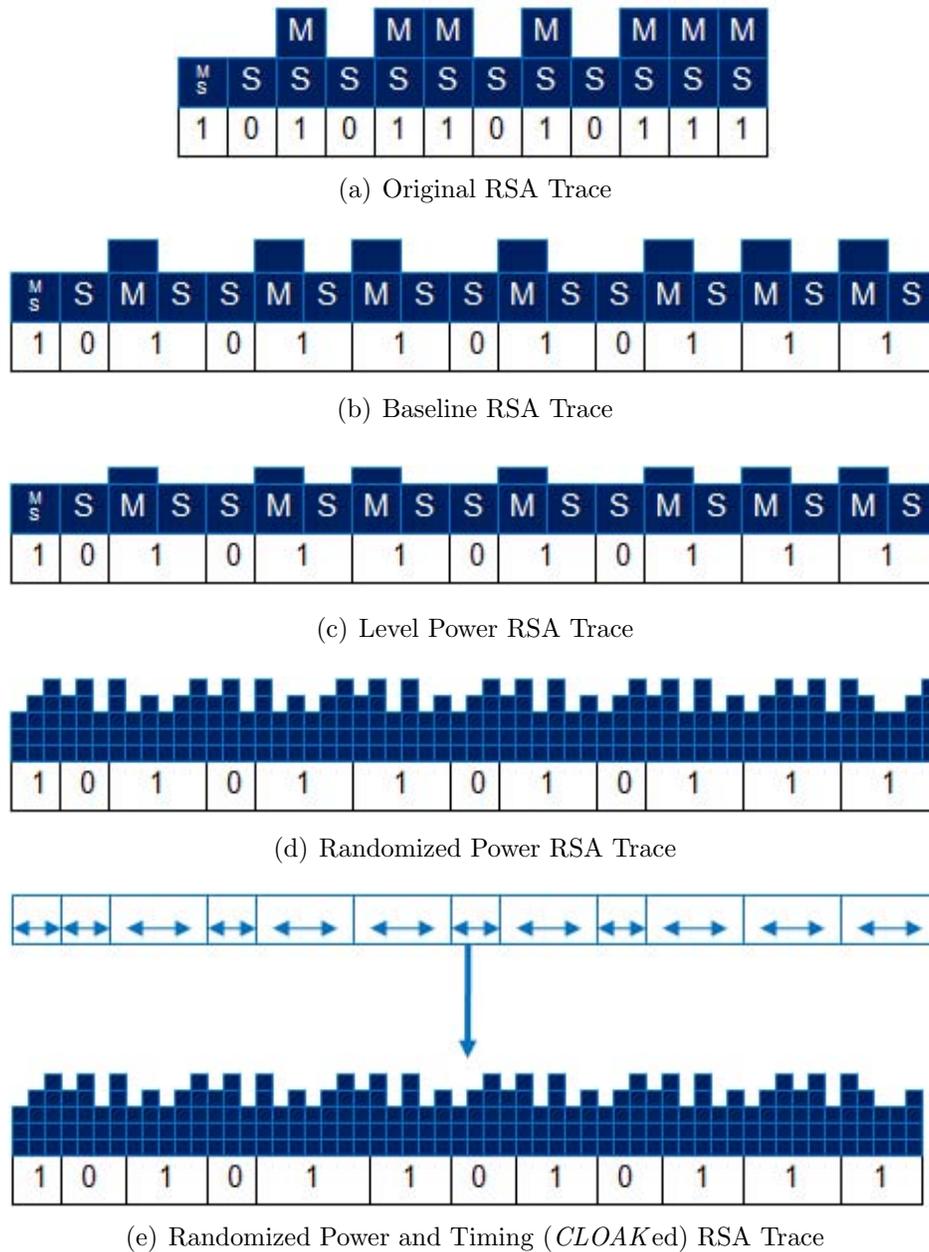


Figure 8. Development of the Polymorphic Circuit Design using Block Diagrams

The polymorphic circuit development is broken down into three primary steps:

1. **Flatten Power Signature:** The original EM trace for RSA is shown in Figure 8(a) where the square and multiply operations, shown with “S” and “M” blocks,

are executed at the same time. Flattening the power involved two iterations. The first iteration shown in 8(b) separated the square and multiply operations in time and became the baseline version of RSA. The second iteration shown in 8(c) reduced the two modular multiplier instantiations down to one instantiation for both multiply and squaring operations to become the level power version of RSA.

2. **Randomize Power Signature:** Once the circuit power consumption was relatively level the power consumption within each multiplier operation was randomized by iterating through multiple adders within each modular multiplier instantiation. The random power version of RSA is shown in Figure 8(d).
3. **Randomize Circuit Timing:** The third and final step in the polymorphic circuit development randomized the circuit power consumption and timing in order to create the final *CLOAK*ed RSA design, shown in Figure 8(e).

3.1.2.2 Implement RSA with Reconfiguration.

An RSA implementation is also designed and implemented on an FPGA using a structural approach, which is a design based on the components and their interconnects. This design approach gives the developer more control over the implementation of the modular exponentiation portion of the cryptographic algorithm. The first implementation of RSA uses unobfuscated modular exponentiation circuitry for baseline analysis. The second implementation uses the polymorphic circuit design for the modular exponentiation circuitry.

3.1.2.3 Analyze Side Channel Signatures.

The EM side channel of the original RSA implementation is characterized to obtain a baseline for subsequent analysis of the side channel. The baseline is compared to the

modified RSA implementation to characterize the extent the circuit is able to resist timing attacks and protect the secret key. It is expected that the combination of elements in this approach culminate in a proof-of-concept polymorphic circuit design that enhances system protective countermeasures by obfuscating operations.

3.2 System Boundaries

The System Under Test (SUT) is the Polymorphic *CLOAK* Encryption System (CES) as shown in Figure 9. The Polymorphic CES consists of four primary components: RSA encryption algorithm, an FPGA, a processor, and the *CLOAK* countermeasure.

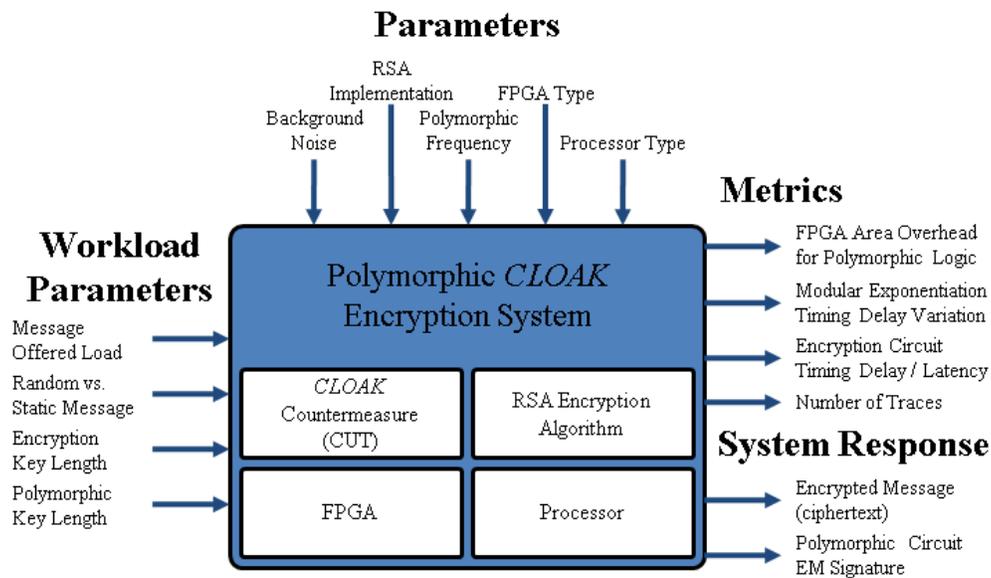


Figure 9. Polymorphic *CLOAK* Encryption System

The Component Under Test (CUT) is the *CLOAK* countermeasure integrated into a hardware implementation of the RSA cryptographic algorithm on a Xilinx Virtex-5 FPGA. The system also contains a PowerPC[®]440 Processor Core to handle all the data I/O interfacing for the system. These components are discussed in more detail

below.

3.2.1 RSA Encryption Algorithm.

The RSA encryption algorithm is a public-key cryptographic algorithm that uses two keys and a modulus to encrypt and decrypt messages. This algorithm is implemented using a structural hardware implementation programmed on the FPGA.

3.2.2 Xilinx Virtex-5 and Virtex-6 FPGAs.

The Xilinx[®]Virtex-5 and Virtex-6 FPGAs consist of an array of reprogrammable gates capable of being reprogrammed by the user. The configuration process starts by defining the Very-High-Speed Integrated Circuit (VHSIC) hardware circuit using the VHSIC Hardware Description Language (VHDL) [2]. The proprietary Xilinx[®]software transforms the users VHDL circuit description into specifications into a *bitstream* to be downloaded and programmed on the chip.

3.2.3 Processor Core.

The PowerPC 440 processor core is a 32-bit processor [14] that serves as the central control unit for the system implemented on the Virtex-5. The MicroBlaze soft-core¹ processor core is also a 32-bit processor that serves as the central control unit for the system implemented on the Virtex-6. Both processor cores handle all the data I/O interfacing for the SUT for each FPGA respectively.

3.2.4 CLOAK Countermeasure.

The *CLOAK* countermeasure is a polymorphic circuit design implemented in the modular exponentiation component within the RSA encryption algorithm. This poly-

¹A soft-core processor is implemented using VHDL on an FPGA

morphic circuit design continuously adjusts the way in which the circuit performs its operations to obfuscate the EM side channel from the adversary.

3.3 System Services

The Polymorphic *CLOAK* Encryption System provides two services.

RSA Block Encryption Service: The block encryption service either encrypts the message or not.

Obfuscation of (*CLOAK*) EM Side Channel: The obfuscation service either works or not, and any amount of partial obfuscation is considered a successful obfuscation. The amount of partial obfuscation depends on the increased number of traces required to successfully retrieve the key.

3.4 Workload Parameters

The workload parameters of the system characterize the requests for service to the system. These parameters describe the quantity and characteristics of the data the system operates on.

3.4.1 Message Offered Load.

Since the system encrypts messages to ciphertext, the number of messages offered to the system drives the system workload. An increase in the number of messages is directly proportional to the workload and the output ciphertext of the system. This trend holds until the capacity of the system is reached. For the experiment, the number of messages offered to the system is limited to the number of traces required in a collection. Initially this value is set to a 1,000 since that is sufficient for the SCA software.

3.4.2 Random versus Static Message.

If the system is using a static message, the system only has to receive and store the message in memory once per data collection. If the system is using a random message each encryption cycle, the workload of the system is reduced by the amount of time it takes to transfer and store the message to memory. For the experiment, a random message is used each encryption cycle.

3.4.3 Encryption Key Length.

The encryption key length changes the number of operations required to complete an encryption cycle. A longer key requires more modular exponentiation operations to produce the resulting ciphertext. An increase in this workload parameter would decrease the amount of data the system can operate on by increasing the encryption circuit execution time. For this experiment, the encryption key length is set to 512 bits. A key length of 512-bits gives the system sufficient workload and security, while still being able to fit on the hardware.

3.4.4 Polymorphic Key Length.

The polymorphic key length determines the circuit change frequency for the polymorphic circuit function. Ultimately the circuit should pseudo randomly change its function each time it is used. That is to say that the encryption circuit should not only vary function timing between the polymorphic function used, but each iteration of the encryption algorithm should be seeded randomly. This randomness in the execution time changes the amount of data the system may operate on. For this experiment, the polymorphic key length is set to 64 bits.

3.5 Performance Metrics

The SUT is evaluated based on FPGA area overhead for polymorphic logic, modular exponentiation timing delay variation, encryption circuit timing delay/latency, and number of traces.

3.5.1 FPGA Area Overhead for Polymorphic Logic.

In every system, area is a limited commodity and should not be wasted. On a typical ASIC, an increase in circuit area equates to an increase in chip area, and therefore, increased cost. On an FPGA, the available programmable space is defined by the manufacturer. To maximize the functionality of a system on a given FPGA, care should be taken not to use all the available space for a given circuit. The cryptographic circuitry is implemented with and without the *CLOAK* countermeasure to determine the increased area usage for the polymorphic logic. System utilization metrics are reported by the proprietary Xilinx software. This metric highlights the feasibility of implementing Polymorphic CES in embedded or space-limited applications.

3.5.2 Modular Exponentiation Timing Delay Variation.

The key to obfuscating the modular exponentiation function of RSA is to vary the timing of the square and multiply functions in such a way that an observer cannot determine which function is being executed. The randomness and frequency of the timing variations within modular exponentiation functions is measured from the time the data is available to when the result is produced for each operation. These results are compared to timing characteristics of normal circuit operations. In stages of modular exponentiation where the circuit executes only a square operation or a square and multiply operation, an observer gains insight into determining the key.

The timing delays for both square and multiply are varied in such a way that an observer cannot determine which one is being executed. This metric highlights the ability of the system to obfuscate its EM side channel signature to reduce the amount of information the adversary can determine about the system.

3.5.3 Encryption Circuit Timing Delay/Latency.

For any encryption system the major bottleneck is the speed of encryptions. Therefore, circuit delay/latency is measured. The delay is defined to be the moment the key and message are available to the encryption circuit to the moment the ciphertext becomes available. The critical path timing delay of the encryption circuit is measured using two methods. First, the maximum delay for the encryption circuitry is estimated and reported by the Xilinx software upon implementation on the chip. Second, the EM side channel is monitored using an oscilloscope triggered by a signal indicating the beginning and end of the encryption cycle. This metric reveals the time required for an encryption cycle.

3.5.4 Number of Traces.

SCA software, whether it be Inspector[®] or Matlab[®], require traces to characterize the systems side channel signature. A trace is the EM or power waveform collected by the oscilloscope and passed to the SCA software for collection and analysis. Once this side channel signature is determined the key can then be extracted. The amount of information an observer can extract about circuit operations increases with the number and quality of traces collected. The number of traces required to determine the key is an indication of the level of circuit obfuscation.

3.6 System Parameters

The system parameters are the characteristics of the system that if changed affects the metrics or the response of the system.

3.6.1 Background Noise.

The emanations given off by electronic equipment within the range of the SUT (hereafter referred to as “Background Noise”) can affect the EM readings. Some of the background noise, such as power lines or clock circuits, occur at regular frequencies and can be identified and filtered out of the resulting traces. Background noise sources are identified and limited to reduce their effect on the SUT.

3.6.2 Implementation of RSA.

The RSA public key algorithm is implemented on a hardware FPGA using structural rather than behavioral VHDL. A structural approach represents the system design in terms of its components and interconnections which allows more control over implementation and how the system will respond. In addition, it enables the polymorphic circuit design to be integrated into the overall implementation of RSA. The characteristics of the EM side channel differs greatly depending upon the implementation method used.

3.6.3 Polymorphic Frequency.

A frequency of the circuit changes is controlled by the length of the polymorphic key. The polymorphic frequency changes the timing of the modular exponentiation function in a pseudo random manner. In turn, this will change the system response and effectively *CLOAK* the function from SCA.

3.6.4 Xilinx FPGAs.

The Xilinx Virtex-5 FPGA is used to implement the hardware portion of the RSA algorithm. The FPGA allows rapid prototyping and testing of the system. Each version and family of FPGAs have slightly different EM side channel signature and timing characteristics. The Virtex-5 chip package consists of a flip-chip design, as shown in Figure 10, can also affect the EM Side channel signature. These differences may change the system speed, power consumption, or area, which may also affect the system response to inputs.

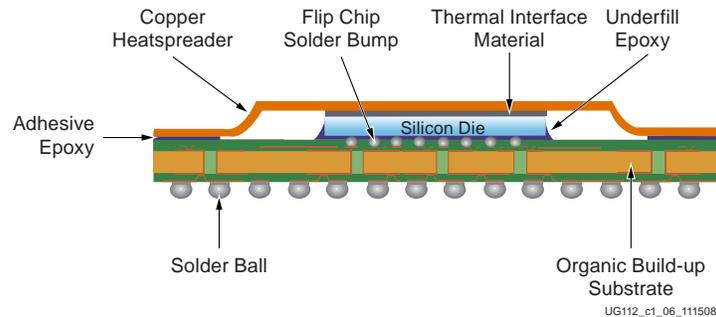


Figure 10. Cross Section for Virtex-5 Flip-Chip Package [11]

The Xilinx Virtex-6 FPGA is also used after initial development to characterize the differences in side channel between the FPGA families. The Virtex-6 chip package also consists of a similar flip-chip design as the Virtex-5. However, the Virtex-6 is built using a smaller $40nm$ process technology enabling decreased power consumption.

3.6.5 Processor Core.

The PowerPC 440 processor core is embedded in the Virtex-5 FPGA and provides data I/O and controls how the system will operate and respond to inputs. The processor core also contributes to the background EM noise of the system. The Virtex-6 FPGA does not contain an integrated PowerPC processor core so a MicroBlaze

soft-core processor core will be used.

3.7 Factors

Table 1 contains a summary of the factors and levels used in this methodology. Each of these factors are described in detail in the following sections.

Table 1. Factors and Levels

<i>Factors</i>	<i>Levels</i>
Polymorphic Frequency	None/Full
FPGA Version for RSA	Virtex-5/Virtex-6
Random vs. Static Message	Random/Static

3.7.1 Polymorphic Frequency.

The ability of the system to adjust its function timing establishes the system's ability to obfuscate the circuit function from the adversary. This research uses two levels of polymorphism none and full polymorphism, where full polymorphism is changing the circuit function for each step of the modular exponentiation algorithm. The existence of timing variations and the ability to obfuscate the circuit are dependent on each other and are said to interact with each other.

3.7.2 FPGA Version for RSA.

RSA can be implemented on any FPGA with suitable resources available to fit the design. Each hardware version will create a different EM side channel signature. The FPGA version is varied to determine its impact on a successful attack. The hardware implementations of RSA uses VHDL to define its structure. The FPGA is varied between Virtex-5 and Virtex-6 designs.

3.7.3 Random versus Static Message.

The message is varied between static (fixed message) and random (varied message) to determine an impact on an attack. A static versus random message may change the side channel signature by changing the loading on the encryption circuit. Varying message content increases the difficulty of characterizing the side channel signature and therefore the ability to attack the system.

3.8 Evaluation Technique

The evaluation technique used is measurement on real hardware since a simulation can only get you so close to actual values. A real system can more accurately evaluate the functions of a circuit. Figure 11 shows the experimental configuration. The system is developed and tested on a Xilinx Virtex-5 board. Measurements (traces) are taken using an Lecroy WavePro 725Zi oscilloscope with a Willtek[®]1207 Inductive EM probe and triggered by pulse signal from the SUT. EM traces from the Polymorphic CES are compiled and analyzed using Riscure's Inspector SCA Test Software version 4.1.1. The SUT will receive the message, key, and control signals from and provide ciphertext to the Inspector software through the serial port.

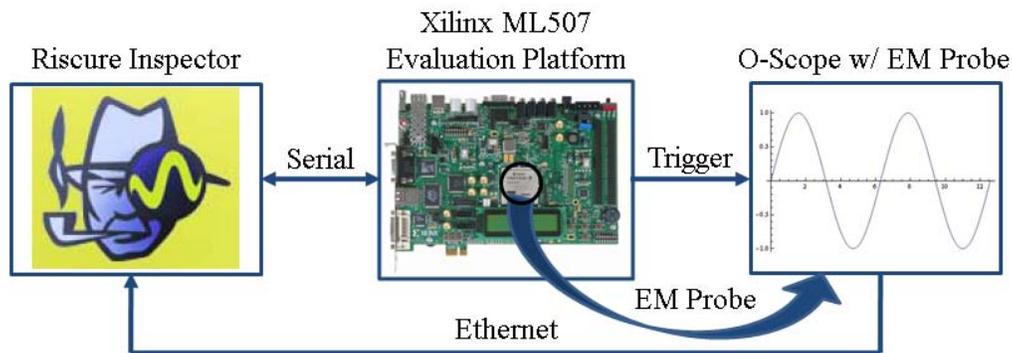


Figure 11. Experimental System Configuration

Section 3.1.2 discusses the baseline traces collected with the Inspector software

throughout the design process. The traces collected from the obfuscated circuit design are validated against the baseline traces to ensure proper functionality. Inspector’s cryptographic analysis software is also used to analyze the circuit’s ability to obfuscate its function by comparing the number of traces needed for baseline tests.

The effectiveness of polymorphic reconfiguration in the modular exponentiation design will be evaluated in three parts, the first being the verification of the individual polymorphic subcircuit side channel signatures. Secondly, the effect on the side channel signature of the polymorphic modular exponentiation operations compared to the baseline signatures. Thirdly, characterize the RSA side channel vulnerability signature of baseline RSA circuit. Finally, a practical case study implementation of the RSA public key cryptography algorithm using the polymorphic modular exponentiation component. The modular exponentiation component of RSA is chosen as the implementation test bench due to its vulnerability to timing attacks using the EM side channel. Evaluation is based on how many traces are required to recover the secret key and how the results compare to the baseline circuit.

3.9 Experimental Design

A full factorial design is conducted. There are 3 factors with 2 levels each (i.e. 2^k , where $k = 3$) resulting in 8 experiments. Two replications of each experiment are conducted for a total of 16 experiments. A 95% confidence level is used for subsequent analysis of the data. This is sufficient to show that the Polymorphic CES can protect a cryptographic circuit from EMA attacks.

3.10 Methodology Summary

This chapter defines the experimental methodology for the Polymorphic CES. The goal of this research is to determine the effectiveness of the polymorphic *CLOAK*

circuit as a countermeasure to EM SCA attacks on RSA. This research focuses on the timing vulnerabilities in the modular exponentiation portions of the RSA algorithm to secure the secret key from possible attack. The SUT is defined and bounded by clearly defining its components. The system services are listed with possible outcomes. Performance metrics are defined based on the system services. System and workload parameters are identified and associated with their sensitivity to the SUT. A list of factors and their levels are chosen from the selected parameters. The experimental design measures the EM side channel signature using an oscilloscope connected to the Inspector SCA software to collect and analyze traces. These traces are validated against baseline measurements to evaluate the effectiveness of the SUT. The factors list is compiled into a full factorial design of a real system resulting in 16 experiments.

IV. Results

This chapter first discusses the experimental setup and the methodology used for attacking the system in Sections 4.1 and 4.2. The chapter continues in Section 4.3 and 4.4 by progressively describing the development of the RSA Encryption System and polymorphic circuit within the design. Results are given throughout the process of development and final results for the Polymorphic CES are given in Section 4.5. Section 4.6 expands the research by adapting the Polymorphic CES design to run on the Virtex-6 board for comparison. Finally, all the designs are compared to each other based on running time and size metrics in Section 4.7. These designs are also compared to comparable industry implementations currently available.

4.1 Experimental Setup



Figure 12. Experimental System Setup

Figure 12 depicts the hardware realization of the system design depicted in the experimental system configuration shown in Figure 11. On the left is the Lecroy oscilloscope, in the middle is the Inspector software used for side channel analysis,

and on the right is the Electromagnetic (EM) probe measuring the unintended EM emissions from the SUT.

Figure 13 shows a close up view of the FPGA with an inductive probe placed on top of the Virtex-5 FPGA with all the connections labeled. Data was not only collected from the top of the FPGA, but it was also collected from the bottom of the board as well.

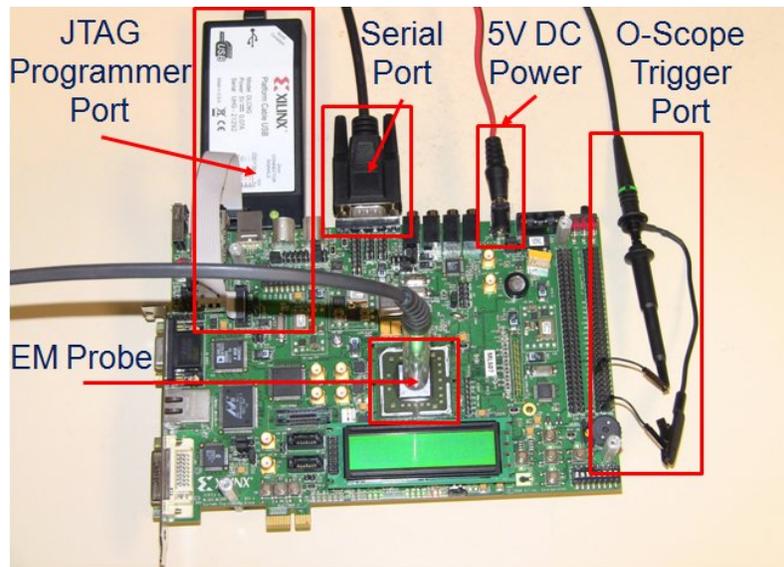


Figure 13. RSA Encryption System Connections

Figure 14 shows the typical Inspector console output for one cycle of RSA Encryption with fixed key, modulus, and plaintext. The first line contains the “set key” command (40h), data length (00 40h = 64 bytes), and the remaining 64 bytes of data is the 512-bit key value. The second line is the echo from the RSA Encryption System. Line three contains the “set modulus” command (42h), data length (00 40h = 64 bytes), and the remaining 64 bytes of data is the 512-bit modulus value. The fourth line is the echo from the RSA Encryption System. Line five contains the “hardware RSA encryption” command (45h), data length (00 40h = 64 bytes), and the remaining 64 bytes of data is the 512-bit plaintext value. The sixth line is the

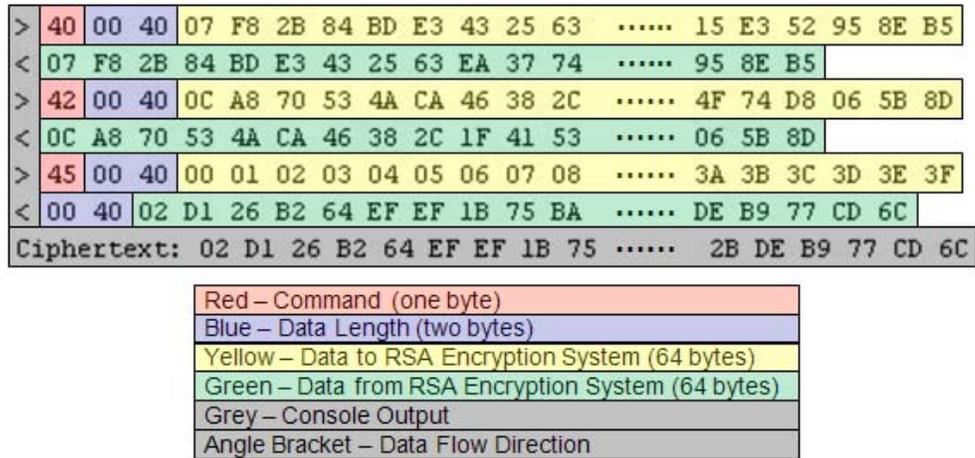


Figure 14. Inspector console output for a round of RSA Encryption using sample key and modulus from [37]

ciphertext response from the RSA Encryption System. The seventh line is an echo from Inspector to the console indicating the ciphertext received from the FPGA via the serial port. Inspector has the capability to randomize the plaintext or key value before sending it to the RSA Encryption System. Retransmission of the key and modulus is not required if every execution cycle is using the same key and modulus.

4.2 Attack Methodology (Top vs. Bottom and Probe Type)

Initially, a small key and plaintext were used to ensure the circuitry worked properly. It was soon discovered that using a characteristic key and plaintext size (i.e., 512-Bits or more) creates considerably larger EM power levels since the circuitry conducts more switching. The flux EM signal of a hardware circuit implementation is directly proportional to the amplitude of the input voltage, which is directly proportional to the switching activity of the circuit [35]. When using a small key the circuit has less switching activity and is easier to hack [45] due to the reduced set of possible key guesses. For this research a realistic full length 512-bit key is used.

During initial attacks on the FPGA it was discovered that the bottom of the

Virtex-5 FPGA contains an array of internal power coupling capacitors (VCCINT), shown in Figure 15. These capacitors not only provide primary internal power to the FPGA, but tend to leak EM information. To compound this issue the System Monitor coupling capacitor is found in the direct center of this array. The System Monitor function is capable of measuring physical operating parameters like on-chip power supply voltages. The EM information leaked by these capacitors relate directly to the operations being performed by the FPGA and provide an optimal avenue for passive EMA of the circuit's power consumption.

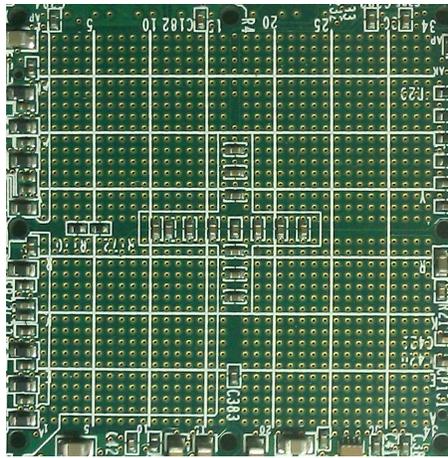


Figure 15. Capacitors on the bottom of the Virtex-5 FPGA

The Virtex-5 FPGA is designed using a Flip-Chip design, shown in Figure 10, in order to reduce the EM emissions and effects from noise [11]. In an attempt to increase the EM signal quality collected from the top of the FPGA the copper heatspreader was removed from the top of the chip. Figure 16 shows the Virtex-5 chip before and after the copper heatspreader was removed. In a typical passive attack, the chip would not likely be depackaged it was necessary to make an informed decision on how to attack the circuit. The removal of the heatspreader did not provide a very noticeable change in signal strength for the Willtek probe. The reduced signal strength on top is partially due to the chip design.

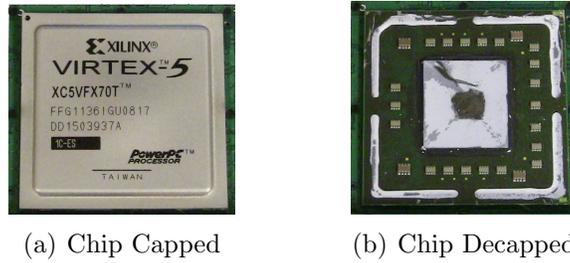
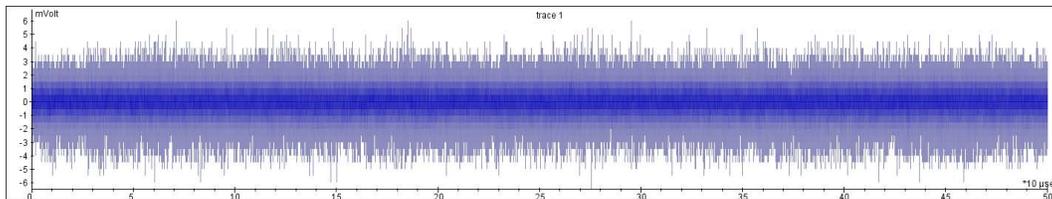
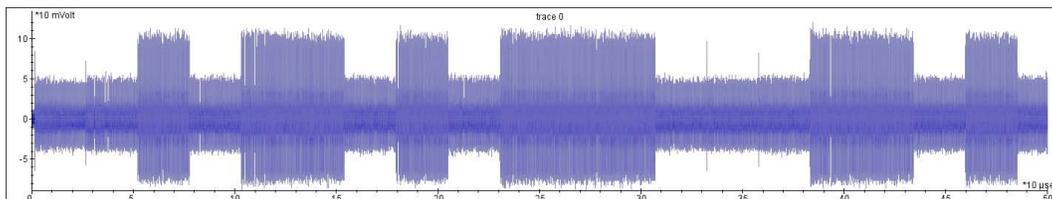


Figure 16. Depackaging the Virtex-5 Chip

As shown in Figure 17, the signal strength between top and bottom of the FPGA is considerably different. The signal from the bottom is much more defined and allowing SEMA to be easily conducted. Due to the lower sensitivity and reduced gain of the Willtek probe it was better suited for attacking the larger signals on the bottom of the FPGA. The Riscure[®] probe is more sensitive than the Willtek probe and becomes



(a) EM signal taken from top of board

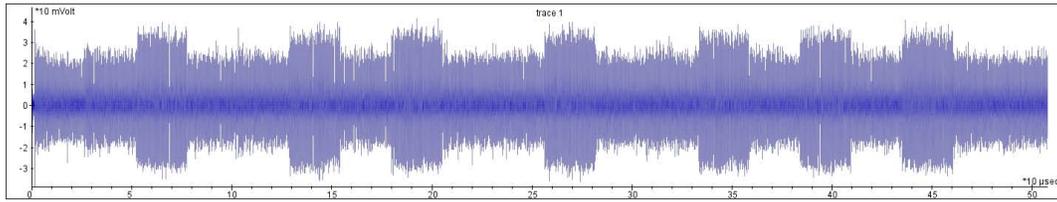


(b) EM signal taken from bottom of board

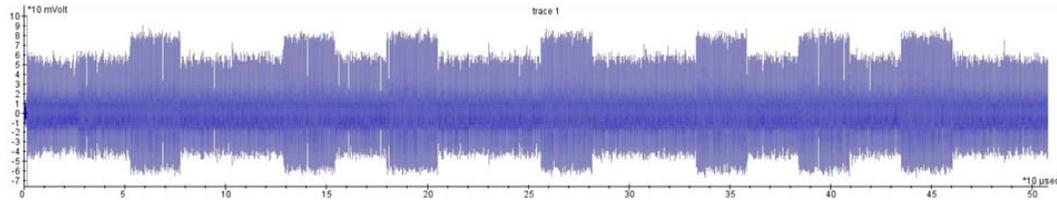
Figure 17. Comparison of EM signal strength using Willtek probe

saturated if used to collect data from the bottom of the board. The Riscure probe is more adapted for smaller and more localized signals. When used to attack the circuit from the top it gets a much better signal than the Willtek probe. Figure 18 shows the difference in EM signal power levels between the two probes. The Willtek probe from the bottom gets almost the double the power amplitude when compared to the

Riscure probe from the top.



(a) EM signal taken with the Riscure probe from top of board



(b) EM signal taken with the Willtek probe from bottom of board

Figure 18. Comparison of EM signal strength between probes

The majority of the data was collected by the Willtek probe from the bottom of the FPGA, which is shown in Figure 19.



Figure 19. EM Inductive Probe on Virtex-5 FPGA

4.3 RSA Encryption Development

The RSA Encryption System was developed using behavioral VHDL and implemented on a Virtex-5 FPGA. The 512-Bit RSA algorithm was designed using the Right-to-Left Square-and-Multiply Algorithm (see Algorithm 3) to implement modular exponentiation. The system components for my RSA Encryption System are shown in Figure 20 and are briefly described below. Figure 50 in Appendix A shows a flowchart of how my RSA Encryption System works.

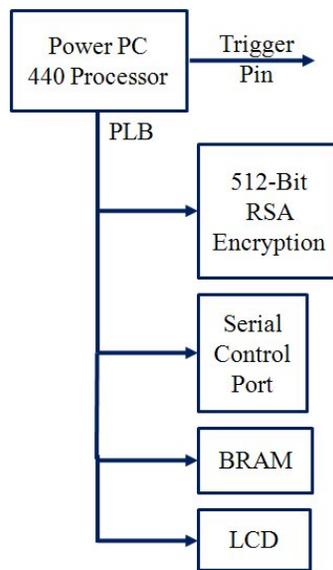


Figure 20. RSA Encryption System

PowerPC 440 Processor: The on-board PowerPC in the FPGA handles the simple I/O functions to transfer the plaintext, modulus, and key inputs via a serial communication to a PC. In addition, the PowerPC handles synchronization of Riscure’s Inspector software with the RSA Encryption System.

Processor Local Bus(PLB): 32-Bit interface between the PowerPC and the IP cores.

512-Bit RSA Encryption: VHDL description of a 512-Bit RSA Algorithm implemented on the FPGA.

Serial Control Port: The serial I/O is used to transfer externally generated key, modulus, and plaintext to the PowerPC to start the encryption cycle. At the end of each encryption cycle, serial I/O is used to transfer the ciphertext back to a commercial SCA software package called Inspector.

Block RAM (BRAM): BRAM is a configurable memory module on the FPGA where the PowerPC instructions are stored.

LCD: The LCD displays the number of encryption cycles that have occurred during system operation.

RSA was initially developed with two equivalent Montgomery modular multiplier instantiations, shown in Figure 6, to execute the square and multiply operations for the circuit. This circuit became Version A as described in Appendix C. The two modular multipliers were programmed to execute simultaneously in time for each loop of the modular exponentiation process in order to save time in the execution process. Once data was collected from the EM side channel of the circuit it was noticed that this circuit structure caused a very dynamic difference in EM signal levels. Figure 21 shows the first $500\mu s$ of the 512-bit trace, shown in Figure 53 of Appendix B. This trace represents the first design iteration that executed the square and multiply operations at the same time and caused a signal that leaked a lot of sensitive information. The SEMA of this trace is also shown in Figure 21 and shows how the private key can simply be read directly from the trace (right to left). The first bit in the Right-to-Left Square-and-Multiply Algorithm appears as though it only conducts a square operation resulting in a 0-bit, but that is only because the message is multiplied by a “1” so that the first multiply operation can be skipped.

Additionally, RSA keys are relatively prime to $\phi(n)$ and will always be odd. This means the LSB is assumed to be “1”.

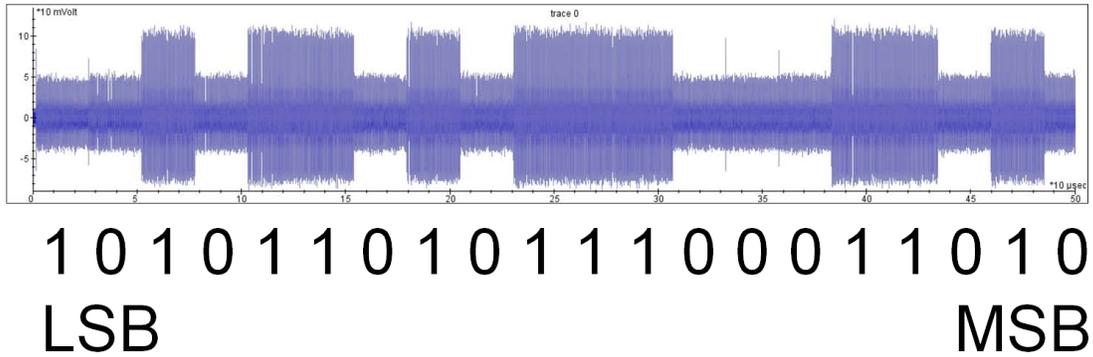


Figure 21. SEMA of Combined Square and Multiply Trace with key (5 8E B5) - RSA Version A

In the second iteration of Hardware RSA code, the square and multiply operations were changed to operate separately in time, but using the same two separate modular multiplier instantiations. This version became the baseline circuit for all future testing and was named Version B as described in Appendix C. Figure 22 shows the first $500\mu s$ of the 512-bit trace, shown in Figure 54 of Appendix B. This trace represents

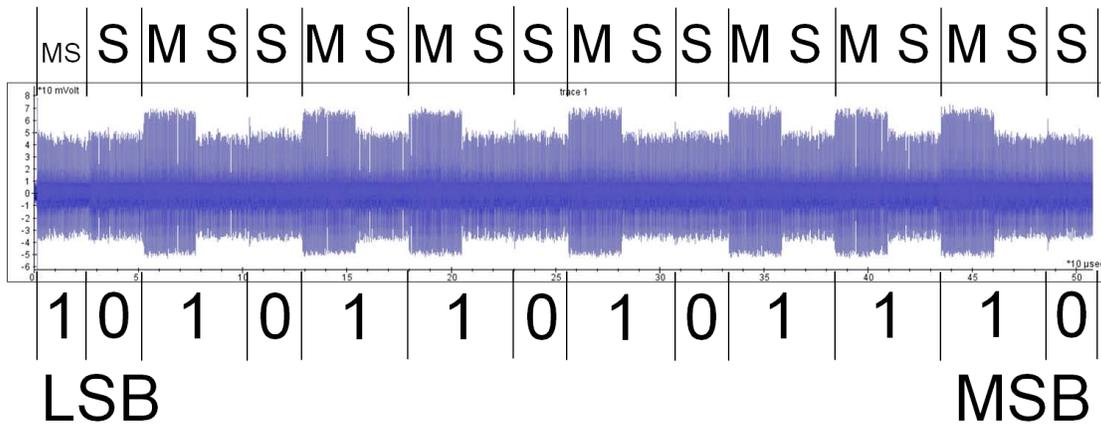


Figure 22. SEMA of Separate Square and Multiply Trace with key (E B5) - RSA Version B

the second design iteration where the square and multiply operations were separated

in time. In this circuit version, a square operation ($d_i = 0$) and a square followed by a multiply operation ($d_i = 1$) can be easily identified since the multiply operations leak approximately $25mV$ more power at peak amplitude. The SEMA of this trace is also shown in Figure 22 and shows how the private key can be easily extrapolated from the trace. The apparent leakage of the trace can also be seen after some signal processing to accentuate the dips in the signal in Figure 23.

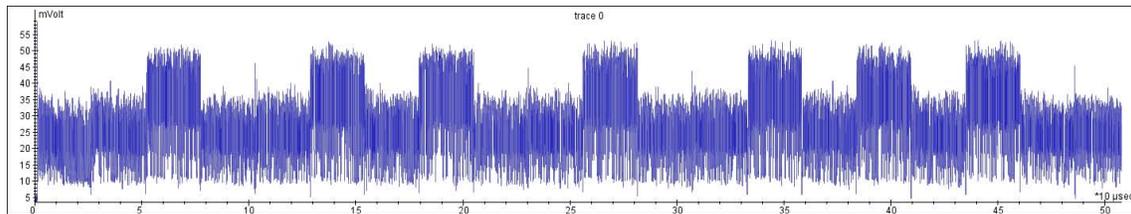


Figure 23. Separate Square and Multiply Trace after signal processing - RSA Version B

The second iteration of Hardware RSA had some inherent timing variations due to how the multiply operations were performed. The timing of the multiply operations were based on the number of bits between the first and last 1-bit in the multiplier input and varied a few clock cycles between each operation. This misalignment in the timing of a randomized input caused some difficulty in conducting DEMA to extract the key from the baseline circuit version. Due to the difficulty in conducting DEMA an alternate version was created that had a constant timing based on a fixed length counter. Figure 24 shows the first $500\mu s$ of the 512-bit trace, shown in Figure 55 of Appendix B. This trace shows the statistical average of 1,000 resampled and signal processed traces using a random plaintext input. The statistical average package removes all the differences between the 1,000 traces and creates a trace containing the average power level for them all. Since the multiply operations use constant timing the SCA software can easily align and extract the statistical differences between the traces. The next step after characterizing the baseline EM signal was to develop the

polymorphic circuit to create the *CLOAK* countermeasure.

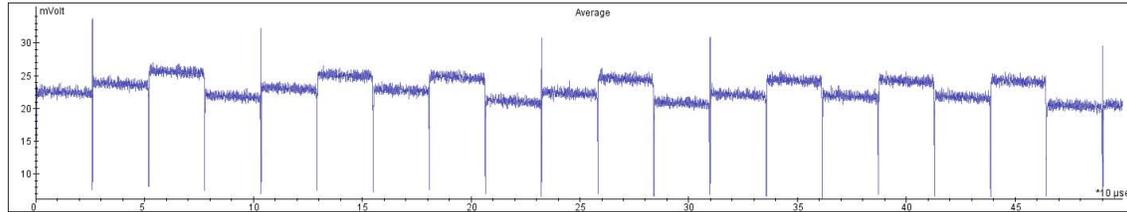


Figure 24. Separate Square and Multiply Trace after signal processing using constant timing - RSA Version 2B

4.4 Polymorphic Circuit Development

The polymorphic circuit development to create the Polymorphic CES was broken down into three steps:

- Flatten Power Signature
- Randomize Power Signature
- Randomize Circuit Timing

4.4.1 Flatten Power Signature.

The first step in the development of the polymorphic circuit design is to flatten the EM power signature of the RSA circuit. Even though the square and multiply operations were separated in time they still were using two separate, but equivalent modular multiplier instantiations. This enabled the circuit to still leak valuable information about the functions being executed. Therefore, the third iteration of hardware RSA was an attempt to flatten the power signature between the square and multiply operations. This was accomplished by creating one modular multiplier instantiation within the circuitry to execute both the square and multiply operations. This circuit became Version C as described in Appendix C.

Figure 25 shows the first $500\mu s$ of the 512-bit trace, shown in Figure 56 of Appendix B. This trace represents the third design iteration that uses one modular multiplier instantiation to execute the square and multiply operations in order to level the circuit's power consumption. The timing boundaries for the square and

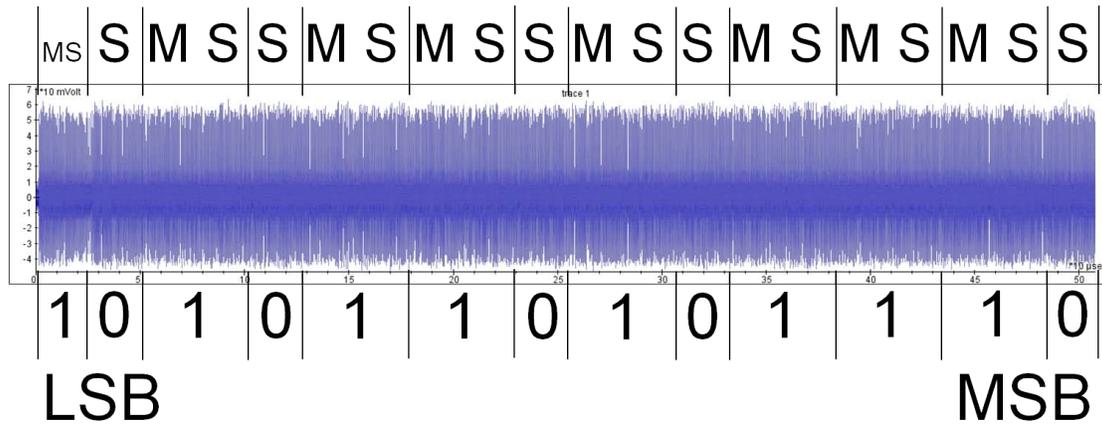


Figure 25. SEMA overlaid on Level Power Consumption Trace with key (E B5) - RSA Version C

multiply operations of this trace are also shown in Figure 25, which shows how the private key can no longer easily be extrapolated from the original trace using SEMA.

However, after some signal processing to accentuate the dips in the signal between the square and multiply operations, shown in Figure 26, more information about the trace can be extrapolated. Signal processing allows the trace to be compartmented, but DEMA needs to be conducted to identify a square versus a multiply operation. The timing boundaries for this trace can also be seen in Figure 26. SEMA can no longer be conducted on the trace so the square and multiply labels are superimposed on the trace as a courtesy. As you can see from the trace, none of the modular operations correlate with any others in order to identify a square from a multiply.

This circuit design provides a compact and low power cryptographic solution capable of providing some protection against side channel attacks. Due to a decreased hardware utilization of 24.6% over the previous implementation; the clock speed was

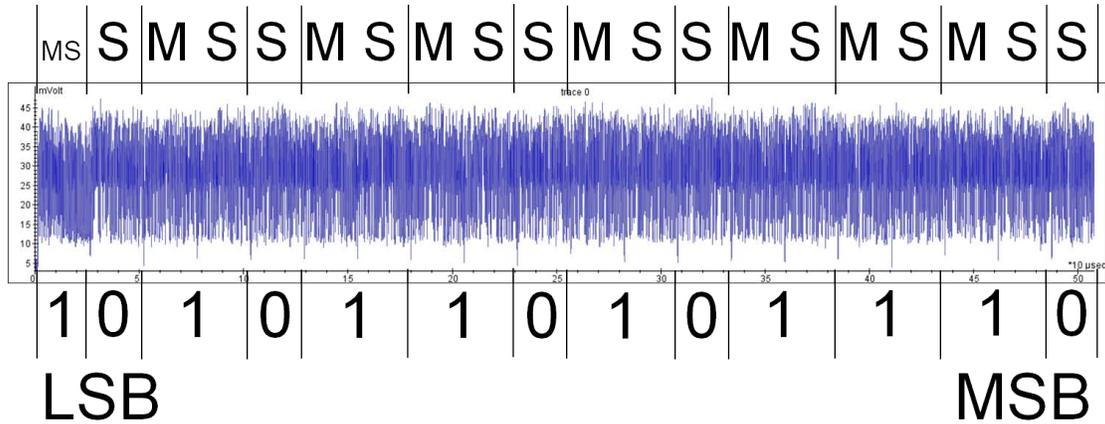


Figure 26. SEMA overlaid on Level Power Consumption Trace with key (E B5) after signal processing - RSA Version C

doubled to $40MHz$. What if the adversary capable of aligning the trace based on the dips in the power trace? In order to characterize this possibility the circuit design was modified to have a constant timing based on a fixed length counter. Figure 27 shows the first $500\mu s$ of the 512-bit trace, shown in Figure 58 of Appendix B. This trace shows the statistical average of 1,000 resampled and signal processed traces using a random plaintext input. Figure 27 shows a trace with a potential $1.5mV$ difference in power levels between a square and multiply operations. This trace is only made possible by using a multiplier with constant timing. In actuality, the adversary would not be able to modify the circuitry to collect these traces.

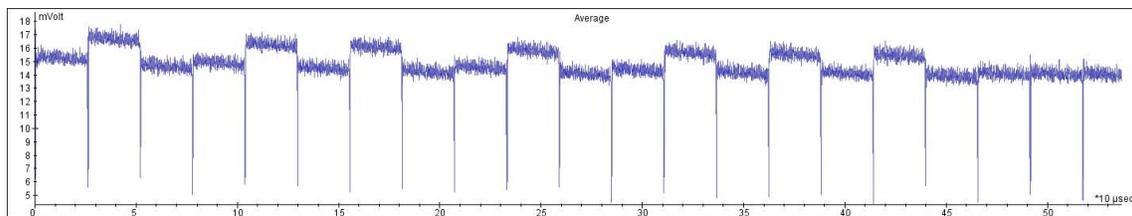


Figure 27. Level Power Consumption Trace after signal processing using constant timing - RSA Version 2C

Instead, the adversary would use a function similar to an Elastic Alignment¹

¹An Elastic Alignment is a trace alignment module contained in Inspector capable of local stretch-

to attempt to align the traces. Using an elastic alignment in conjunction with a statistical average on approximately 1,000 random plaintext traces yields the first $500\mu s$ of the 512-bit trace, shown in Figure 28. The full 512-bit trace can be found in Appendix B, Figure 59. Now that the power signatures are relatively level they can be randomized.

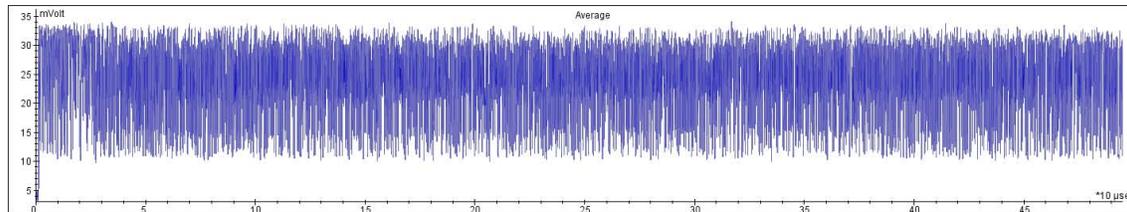


Figure 28. Level Power Consumption Trace after elastic alignment and average - RSA Version C

4.4.2 Randomize Power Signature.

The second step in the development of the polymorphic circuit design is to randomize the EM power signature. Once the power signature was relatively flattened between multiply and square operations, the power levels within each modular multiply operation was then varied in a pseudo-random manner. The fourth and fifth iteration of the hardware RSA implemented two and three different adders, respectively. Figure 29(a) shows the polymorphic multiplier with a blue box where the polymorphic adder in Figure 29(b) is placed in the design. The first adder used the standard VHDL add operator, the second a ripple carry adder, and the third a carry look-ahead (CLA) adder. This changed the possible power levels for each addition operation to three different levels. Therefore, using DEMA would be more difficult since the adders used in each of the operations have three possible correlation levels

ing and compression of traces at multiple points based on a reference trace for synchronization necessary for DPA or DEMA.

for each possible input tripling the complexity. These circuits became Version D and E respectively as described in Appendix C.

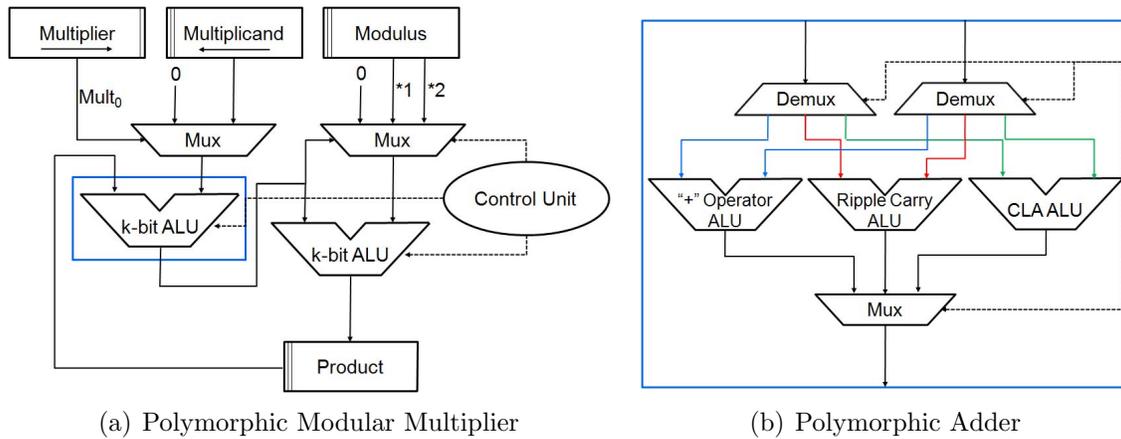


Figure 29. Polymorphic Modular Multiplier with Polymorphic Adder

Figure 30 shows the first $500\mu s$ of the 512-bit trace shown in Figure 60 of Appendix B. This trace represents the fifth design iteration that uses one modular multiplier instantiation that contains multiple adders in order to randomize the circuit’s power consumption. The timing boundaries for the square and multiply operations of this

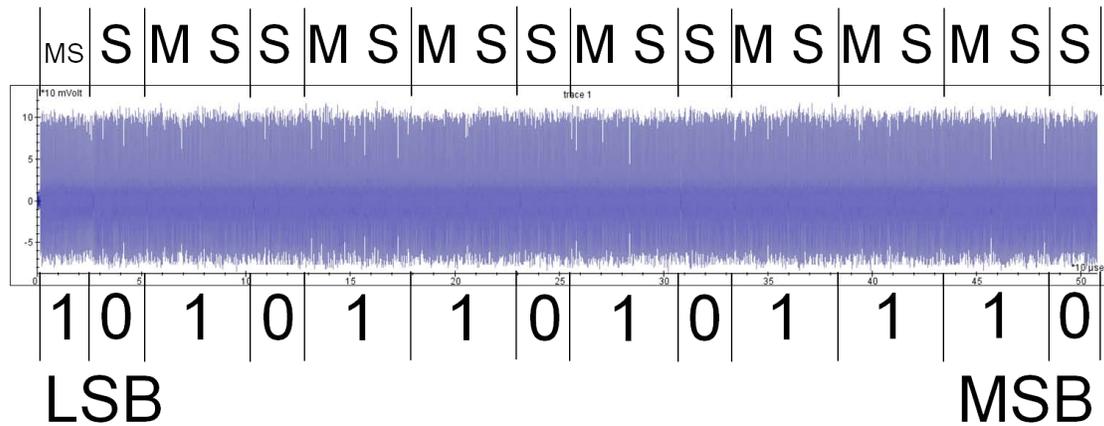


Figure 30. SEMA overlaid on Randomized Power Consumption Trace with key (E B5) - RSA Version E

trace are also shown in Figure 30, which shows how the private key can no longer

easily be extrapolated from the trace using SEMA. The square and multiply labels are superimposed on the trace as a courtesy to the reader.

After signal processing to accentuate the dips in the signal, shown in Figure 31, more information is revealed about the trace. Signal processing allows the trace to be compartmented, but DEMA needs to be conducted to identify a square versus a multiply operation. The timing boundaries for this trace can also be seen in Figure 31, though SEMA can no longer easily be conducted. As you can see, each multiply and square operation has a randomized power signature and each operation is different from the next revealing little about the operation being performed. With a software implementation, the operations would have regular patterns, but in this hardware implementation the operations are dependent on the inputs to the multiplier instantiation.

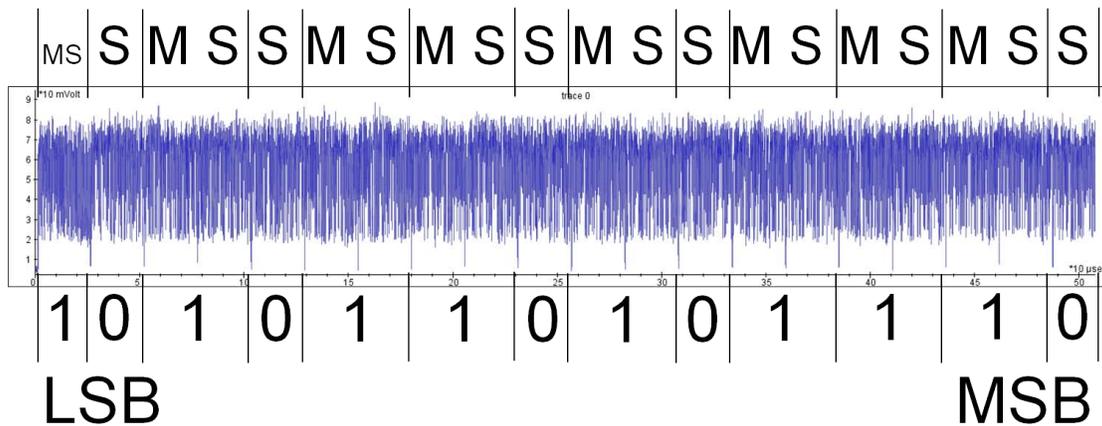


Figure 31. SEMA overlaid on Randomized Power Consumption Trace with key (E B5) after signal processing - RSA Version E

Using an elastic alignment in conjunction with a statistical average on approximately 1,000 random plaintext traces yields the first 500 μ s of the 512-bit trace, shown in Figure 32. For this design the timing is not randomized enough at the beginning of the trace so in the first 500 μ s we can differentiate the squares from the multiplies. The full 512-bit trace can be found in Appendix B, Figure 62. The full trace shows

that after the first $500\mu s$ the dips in the signal are still apparent, but a square cannot be distinguished from a multiply operation. Naturally, these results leads to the next logical step, which is randomizing circuit timing.

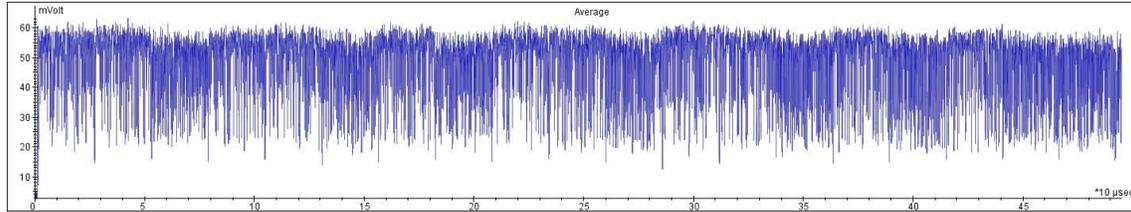


Figure 32. Randomized Power Consumption Trace after elastic alignment and average - RSA Version E

4.4.3 Randomize Circuit Timing.

The third step in the development of the polymorphic circuit design is to use a 64-bit LFSR to pseudo-randomly adjust the execution times of each of the multiply operations. Initially the multiplier execution had a slight randomness since the execution time ended after the multiplier input shifted past the most significant 1-bit, but only offered a couple of clock cycles of variation between each multiply operation. First, a 64-bit LFSR was created using the feedback polynomial shown in Equation 2. This PRNG generates approximately $2^n - 1 = 2^{64} - 1 = 18.5 \times 10^{18}$ possible different numbers before repeating. Selected bits of the generated number were then used to make execution timing decisions shown in Figure 33.

$$x^{64} + x^{63} + x^{61} + x^{60} + 1 \quad (2)$$

The timing flowchart shown in Figure 33 shows that at the beginning of each multiply or square operation, the first decision (**green decision**) is executed. This

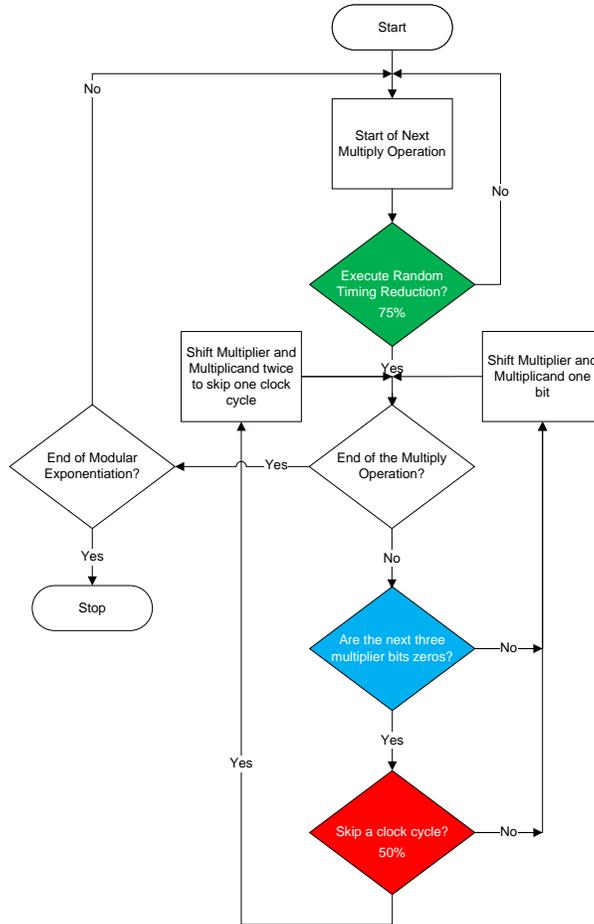


Figure 33. Pseudo-Random Timing Flow Chart

decision is based on the two LSBs of the LFSR and will execute an increased pseudo-random timing reduction approximately 75% of the time (i.e., both bits are 0-bits). Otherwise, the original timing reductions based on the number of bits between the first and last 1-bit in the multiplier input will be executed. The increased timing reductions involve with the second and third decisions. The second decision (blue decision) checks if the three LSBs of the multiplier input are zeros. If the three LSBs are not zeros there is no change in the timing, but if they are zeros then the circuitry moves to the third decision (red decision). The third decision makes a binary decision on whether or not to skip a clock cycle based on the LSB of the LFSR and executes

50% of the time. A clock cycle in the multiply operation is skipped by simply shifting the multiplier right two bits and the multiplicand left two bits instead of the normal one bit.

The final *CLOAK* modular multiplier is shown in Figure 34 with a red box showing where the circuitry that randomizes the timing of the multiplier operations. Using the logic shown in the timing flow chart, the execution time of each multiply operation was able to be randomized from approximately 490 ± 25 clock cycles. Given the clock frequency of the RSA circuit is running at $20MHz$, the timing of each multiply operation is randomized from $24 \pm 1.25\mu s$. In the grand scheme of things that does not seem like much time, but this makes it very difficult to align multiple traces even if the attacker is able to acquire multiple traces using the same key, modulus, and plaintext.

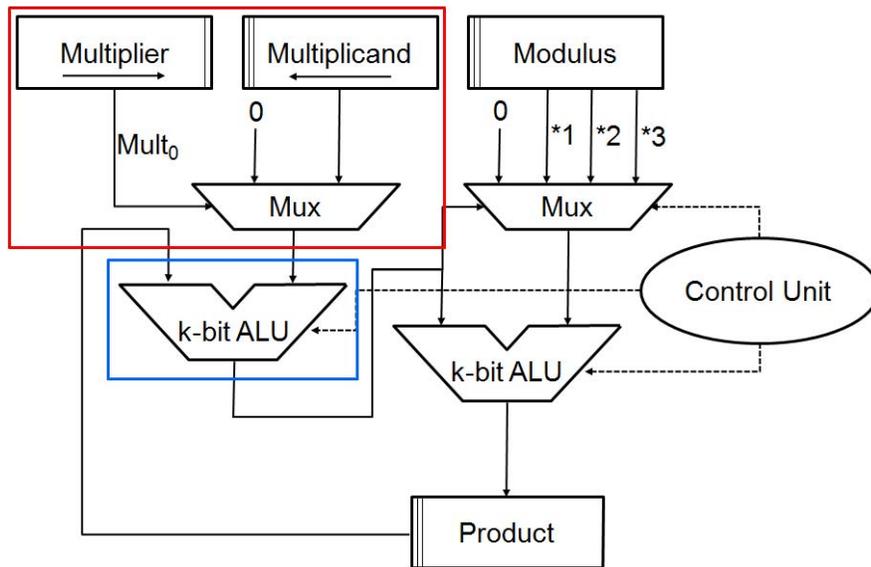


Figure 34. *CLOAK* Modular Multiplier

Figure 35 shows the first $500\mu s$ of the 512-bit trace shown in Figure 63 of Appendix B. This trace represents the sixth design iteration that uses one modular multiplier instantiation that contains multiple adders in order to randomize the circuit's power

consumption as well as circuitry to randomize the multiplier execution time. This circuit became Version F as described in Appendix C.

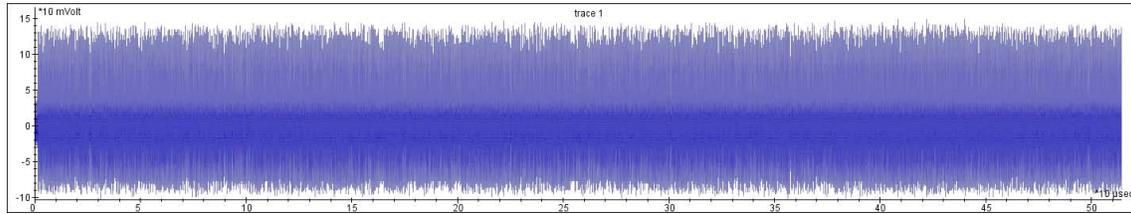


Figure 35. Randomized Power and Timing Trace - RSA Version F

The timing boundaries for the square and multiply operations of this trace are also randomized in order to reduce the ability for an attacker to align the traces for DEMA. The timing boundaries for the square and multiply operations cannot be determined from the trace shown in Figure 35. The timing boundaries for each trace is slightly different from each other and also differs from previous versions of RSA. This trace shows how the private key can no longer easily be extrapolated from the trace using SEMA and if you took several traces side by side there is no way to effectively align the traces for DEMA.

After signal processing to accentuate the dips in the signal, shown in Figure 36, more information is revealed about the trace. Signal processing allows the trace to become more compartmented, but as you can see the dips in this trace are not as defined as in previous versions. We know that DEMA needs to be conducted to identify a square versus a multiply operation, but the SCA software has a difficult time aligning the traces since the multiplier execution time has been randomized at different points within each multiply cycle. The timing boundaries for this trace can also be seen in Figure 36 and square and multiply labels are superimposed for convenience, though SEMA can no longer easily be conducted. As you can see, each multiply and square operation has a randomized power signature as well as randomized circuit timing. Each operation is different from the next, revealing very

little about the operation being performed.

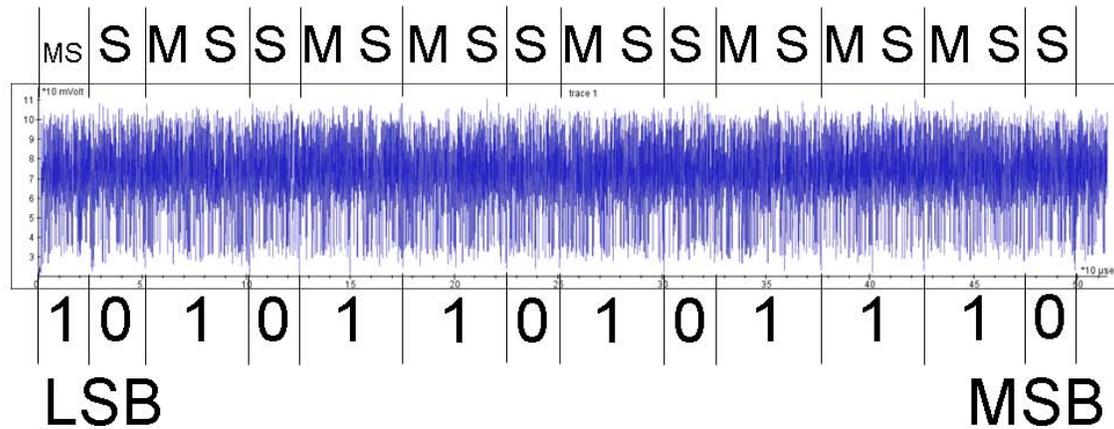


Figure 36. SEMA overlaid on Randomized Power and Timing Trace with key (E B5) after signal processing - RSA Version F

Using an elastic alignment in conjunction with a statistical average on approximately 1,000 traces with a random plaintext applied to a fixed key and modulus yields the first $500\mu s$ of the 512-bit trace, shown in Figure 37. For this design, the

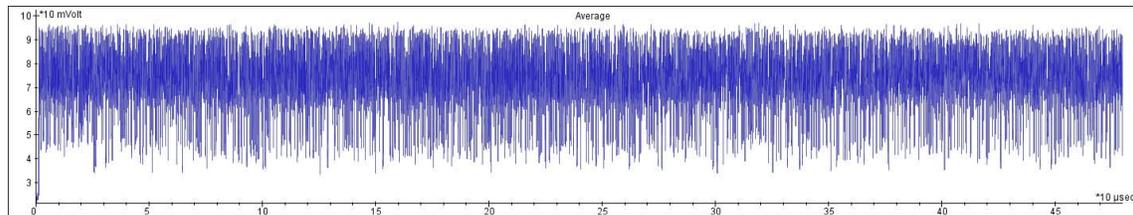


Figure 37. Randomized Power and Timing Trace after elastic alignment and average - RSA Version F

timing is randomized more than before so that in the first $500\mu s$ we cannot align the signals enough to differentiate the squares from the multiplies. The full 512-bit trace can be found in Appendix B, Figure 65. Once the Polymorphic CES development was completed the results were then analyzed.

4.5 Protected RSA SCA Results

Figure 38 shows a comparison of EM signals from three circuit versions for a typical square and multiply operation after signal processing. An example comparison of EM signals before signal processing can be seen in Figure 66 in Appendix B. The

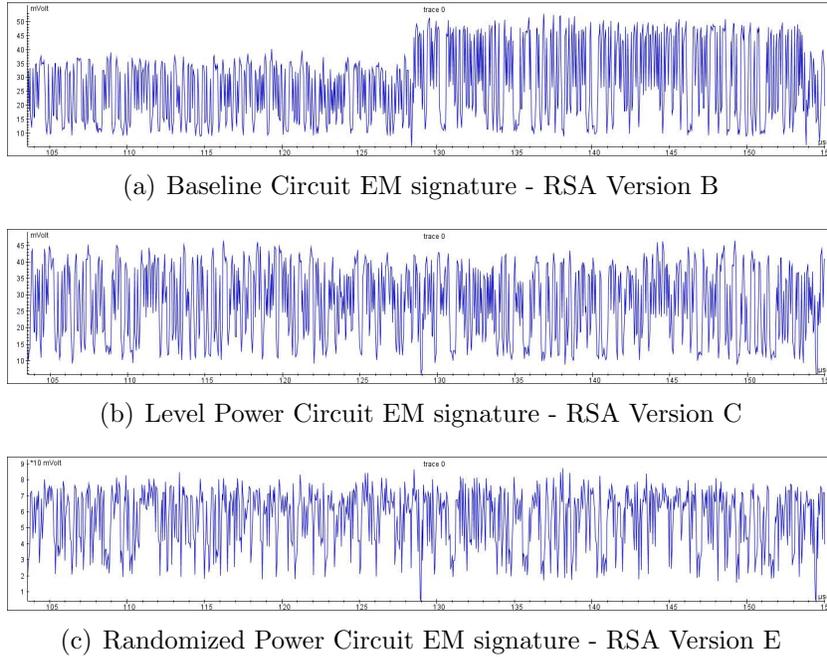
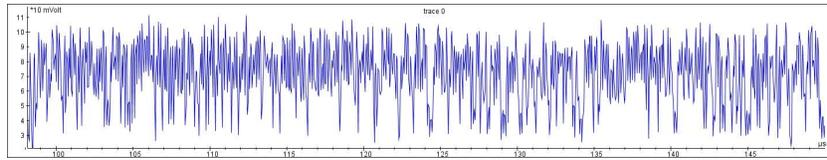


Figure 38. Comparison of EM signals for square and multiply operations after signal processing

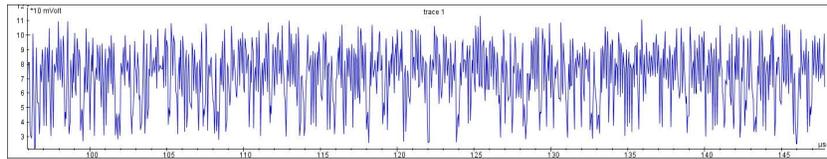
three sample square and multiply operations were taken from traces at the same time period using the same key, modulus, and plaintext so the operations are identical for each trace. These traces show the differences in the signatures of a characteristic square and multiply operation. In actuality, no multiply or square operation has the same signature as any other within the same trace since the multiplier operation is a function of its inputs. This characteristic is exploited in the polymorphic circuit design by randomizing power and then randomizing the timing within the multiply function and therefore the overall encryption cycle.

Once these randomized power signatures are coupled with randomized circuit tim-

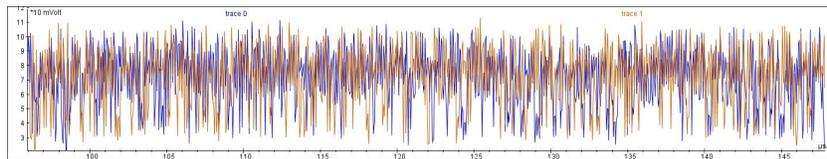
ing, the complexity of DEMA is compounded. A square is virtually indistinguishable from a multiply. Figure 39 shows a comparison of EM signals from the Polymorphic Circuit using randomized power and timing. These traces show the same multiply



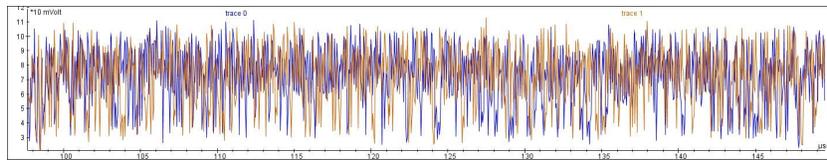
(a) Randomized Power and Timing Trace 0



(b) Randomized Power and Timing Trace 1



(c) Randomized Power and Timing two traces overlaid



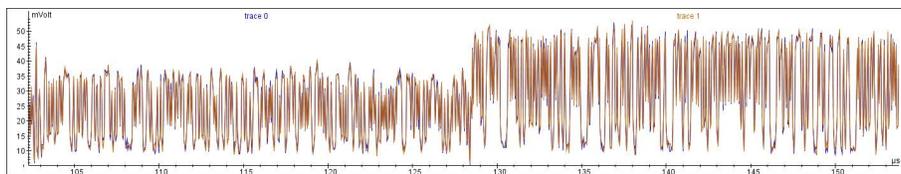
(d) Randomized Power and Timing two aligned traces overlaid

Figure 39. Comparison of EM signals for square and multiply operations using identical inputs - RSA Version F

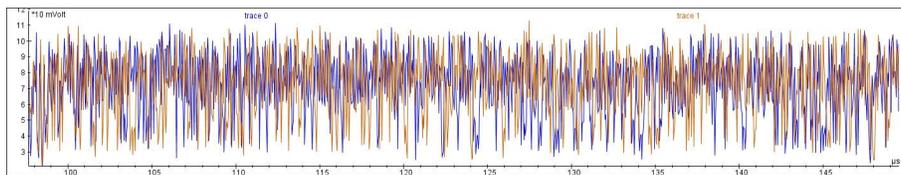
and square operation with the same inputs as the traces in shown in Figure 38. The traces show the same square and multiply operations from two different traces using the same circuitry and identical inputs. However, the two traces are not that similar. Trace 0 and Trace 1 are shown in Figure 39(a) and 39(b) respectively. Figure 39(c) shows the two traces overlaid in real time and Figure 39(d) shows the two traces after they were aligned to have the same start time. The two traces show the two very different power signatures and timing even though they are both operating on iden-

tical inputs. With hardware circuitry operating on the same inputs the trace would typically have virtually identical signals.

Figure 40 shows the same square and multiply operations from the protected and unprotected circuits. With the unprotected circuit in Figure 40(a) the identical inputs produce two virtually identical signals. However, with the protected circuit in Figure 40(b) the two traces are very different and we are unable to correlate a square from a multiply operation. This shows visually how different the two traces are, but next we will find out how different they really are.



(a) Two Separate Square and Multiply Traces (Baseline) - RSA Version B

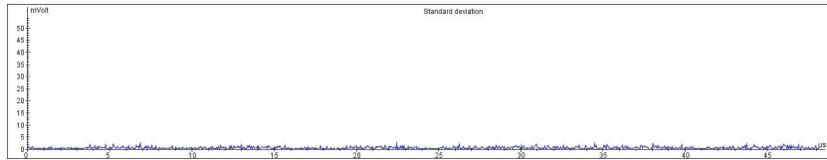


(b) Two Randomized Power and Timing Traces (*CLOAK*ed) - RSA Version F

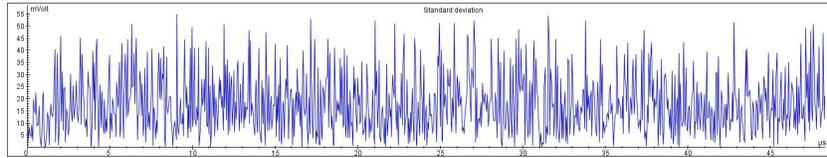
Figure 40. Comparison of square and multiply operations using identical inputs

Figure 41 shows the difference in standard deviation for the same protected and unprotected square and multiply operations using identical inputs. The trace from the *CLOAK*ed circuit in Figure 41(b) shows 23 times or 2300% increase in standard deviation over the baseline trace shown in Figure 41(a).

To decrease the noise in the signal a band pass filter was applied to $20\text{MHz} \pm 1\text{MHz}$ to remove all signals above and below the hardware operating frequency. Figure 42 shows the difference in standard deviation for the same protected and unprotected square and multiply operations using identical inputs. The trace from the *CLOAK*ed circuit in Figure 42(b) now shows 28 times or 2800% increase in standard deviation



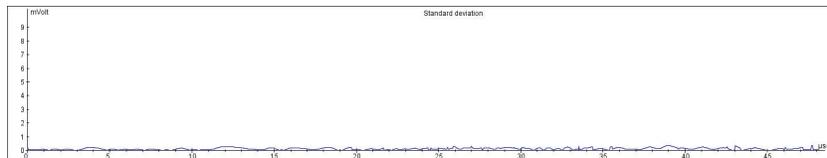
(a) Separate Square and Multiply Trace (Baseline) - RSA Version B



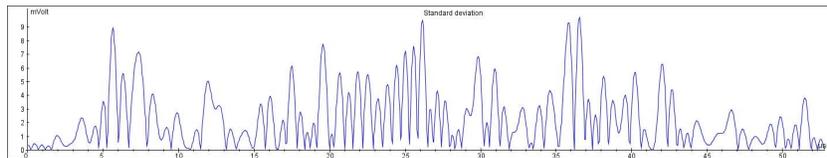
(b) Randomized Power and Timing Trace (*CLOAK*ed) - RSA Version F

Figure 41. Comparison of standard deviation of square and multiply operations using identical inputs

over the baseline trace shown in Figure 42(a). So after we applied a filter to reduce noise the standard deviation increased even more.



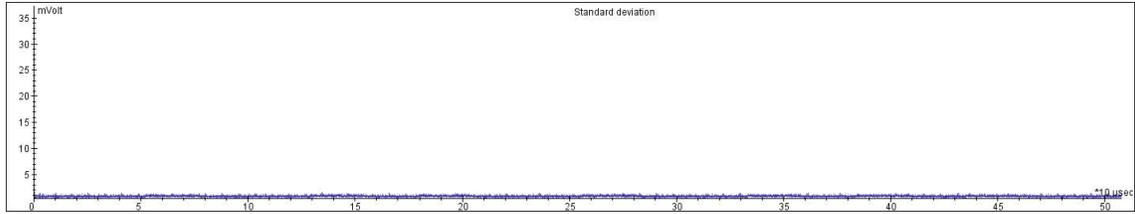
(a) Separate Square and Multiply Trace (Baseline) - RSA Version B



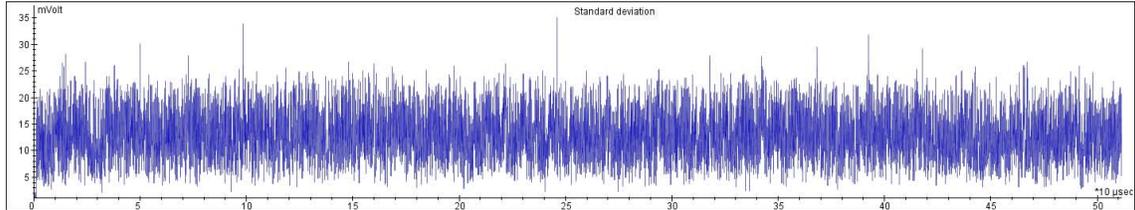
(b) Randomized Power and Timing Trace (*CLOAK*ed) - RSA Version F

Figure 42. Comparison of standard deviation of square and multiply operations after frequency filtering

Additionally, a standard deviation of the entire trace was calculated across 100 traces using identical inputs. Figure 43 shows the difference in standard deviation for the first $500\mu s$ of the full protected and unprotected traces. The trace from the *CLOAK*ed circuit in Figure 43(b) shows 18 times or 1800% increase in standard deviation over the baseline trace shown in Figure 43(a).



(a) Separate Square and Multiply Trace (Baseline) - RSA Version B



(b) Randomized Power and Timing Trace (*CLOAK*ed) - RSA Version F

Figure 43. Comparison of standard deviation of 100 traces using identical inputs

4.6 Virtex-6 FPGA Investigations

The Polymorphic CES design was designed, implemented, and tested thoroughly on the Virtex-5 FPGA, but due to size limitations the hardware implementations of both AES and RSA were not able to coexist on the same chip. The CES was then adapted to run on the newer 40nm process technology the Virtex-6 FPGA is built on. This modification was conducted not only to characterize the difference in side channel signatures, but also to create a fully functional Encryption System testbed. The system was initially implemented with the baseline RSA circuit named Version B as described in Appendix C. The system also contained a hardware and software implementation of AES. Figure 51 in Appendix A shows a flowchart of how the Encryption System works on the Virtex-6 FPGA. Even with hardware versions of AES and 512-bit RSA both implemented on the FPGA the Virtex-6 was only 24% utilized.

The Virtex-6 FPGA has a similar Flip-Chip design as the Virtex-5 FPGA aside from being build using smaller process technology. In addition, the Virtex-6 FPGA

comes equipped with a cooling fan and heat sink, shown in Figure 44, installed on the copper heatspreader. The presence of this fan assembly on top of the FPGA limits the ability of a non-invasive attack on the circuit from the top. It was also discovered that this fan while powered up adds an average of $15mV$ of noise to the already reduced signal strength so for initial data collection the fan was turned off.

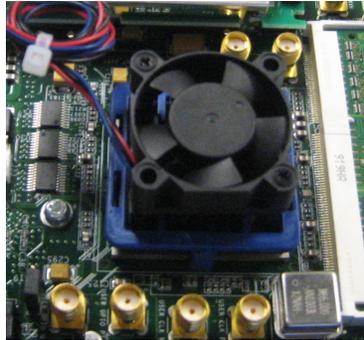


Figure 44. Fan Assembly on Virtex-6 FPGA

Figure 45 shows the EM probe on the bottom of the Virtex-6 FPGA where all data collection occurred. Although the Willtek probe is shown in the picture the Riscure probe was also used for data collection from the bottom of the Virtex-6 FPGA.

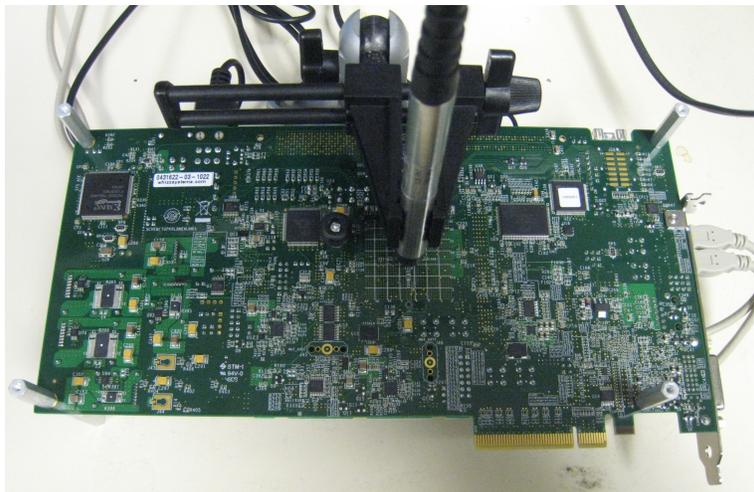
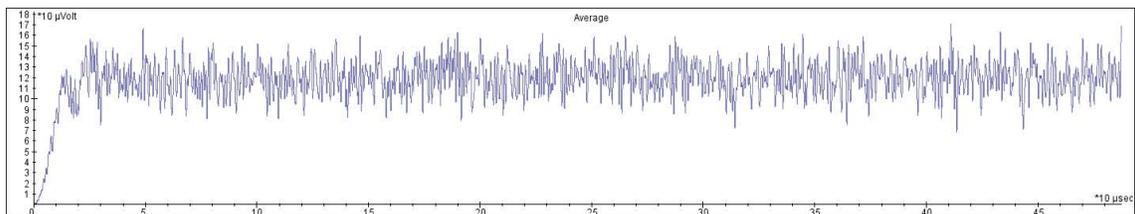


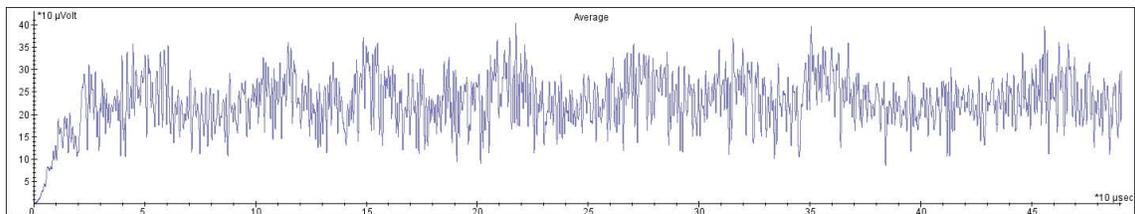
Figure 45. EM Inductive Probe on Virtex-6 FPGA

4.6.1 Unprotected RSA SCA Results.

Initially the Encryption System was implemented on the Virtex-6 FPGA so that the hardware RSA operated at the same $25MHz$ frequency as the MicroBlaze and the hardware AES. The traces shown in Figure 46 show a comparison of the first $500\mu s$ of the 512-bit trace shown in Figures 67 and 68 in Appendix B. These traces represent the baseline RSA Version B circuit, where the square and multiply operations operate separately in time, but use two separate modular multiplier instantiations. These traces show the difference in signals when other circuitry is running on the system at the same clock frequency and cannot be filtered out of the trace. Due to the added noise generated by the MicroBlaze running at the same frequency as the RSA hardware in Figure 46(a) the square and multiply operations are not differentiable with the naked eye. Using the Willtek probe the EM signal strength was considerably lower than with the Riscure probe, but was able to filter out the noise and focus more on the FPGA operations due to the wider resolution of the Willtek probe.



(a) Separate Square and Multiply Trace using a same clock - RSA Version B

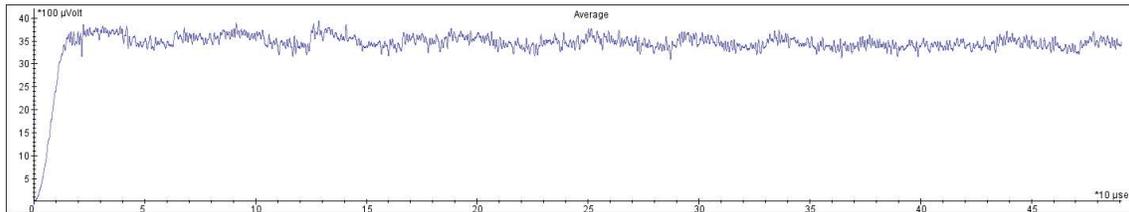


(b) Separate Square and Multiply Trace using a different clock - RSA Version 4B

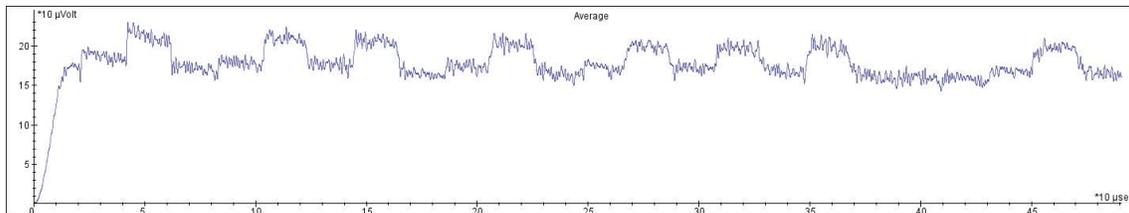
Figure 46. Comparison of EM signals for Separate Square and Multiply Trace after signal processing using fixed plaintext on a Virtex-6 FPGA

Using a statistical average on approximately 100 random plaintext traces yields

the first $500\mu s$ of the 512-bit trace, shown in Figure 47. The full 512-bit trace can be found in Appendix B, Figure 69 and 70. The full trace shows that after the first $500\mu s$ the square and multiply operations are still apparent on the Virtex-6 signals though the dips in the signal are not as apparent.



(a) Separate Square and Multiply Trace using a same clock - RSA Version B



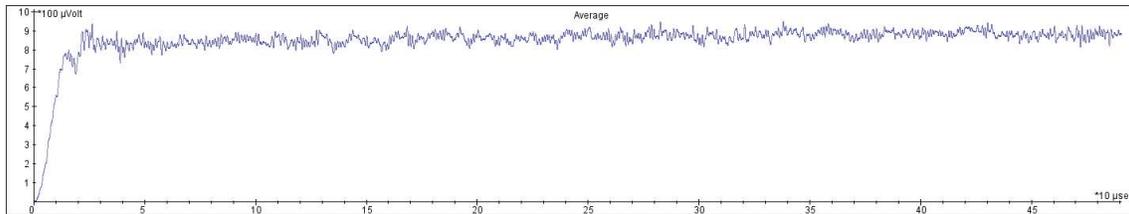
(b) Separate Square and Multiply Trace using a different clock - RSA Version 4B

Figure 47. Comparison of EM signals for Separate Square and Multiply Trace after signal processing using random plaintext on a Virtex-6 FPGA

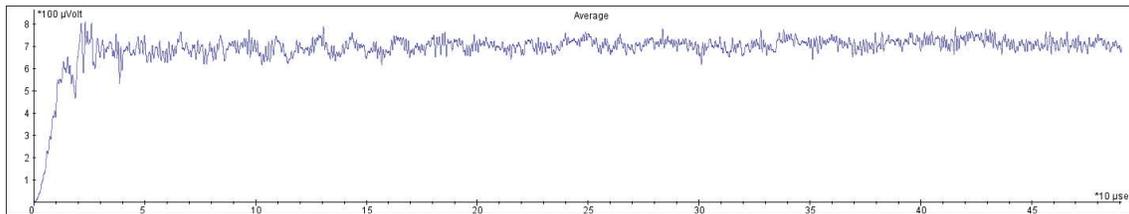
4.6.2 Protected RSA SCA Results.

The initial design for the Polymorphic CES on the Virtex-6 FPGA was implemented so that the hardware RSA operated at the same $25MHz$ frequency as the MicroBlaze and the hardware AES. The traces shown in Figure 48 show a comparison of the first $500\mu s$ of the 512-bit trace shown in Figures 71 and 72 in Appendix B. These traces represent the RSA Version F circuit, where the *CLOAK* countermeasure randomizes power and timing for the RSA circuit. These traces show the difference in signals when other circuitry is running on the system at the same clock frequency and cannot be filtered out of the trace. Even after the noise generated by the MicroBlaze hardware was removed from the signal shown in Figure 48(b) the square and multiply

operations are not differentiable with the naked eye and therefore SEMA cannot be conducted.



(a) Randomized Power and Timing Trace using a same clock - RSA Version F

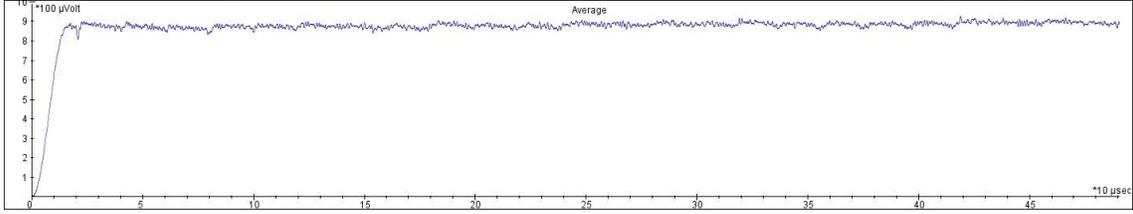


(b) Randomized Power and Timing Trace using a different clock - RSA Version 4F

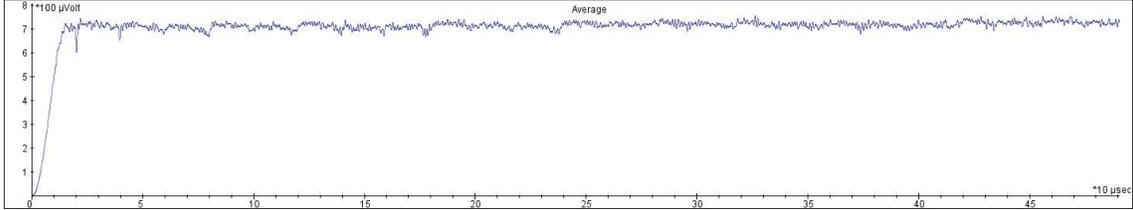
Figure 48. Comparison of EM signals for Randomized Power and Timing Trace after signal processing using fixed plaintext on a Virtex-6 FPGA

Using a statistical average on approximately 100 random plaintext traces yields the first $500\mu s$ of the 512-bit trace, shown in Figure 49. The full 512-bit trace can be found in Appendix B, Figure 73 and 74. The full trace shows that after the first $500\mu s$ the dips in the signal are still apparent, but a square cannot be distinguished from a multiply operation.

The Virtex-6 FPGA had an overall lower power signal than the Virtex-5 FPGA. This lower power signal made it more difficult to process and differentiate the traces, but the results were still the same. The square and multiply operations are not differentiable. Therefore, Polymorphic CES successfully obfuscates the EM side-channel on both the Virtex-5 and Virtex-6 FPGAs.



(a) Randomized Power and Timing Trace using a same clock - RSA Version F



(b) Randomized Power and Timing Trace using a different clock - RSA Version 4F

Figure 49. Comparison of EM signals for Randomized Power and Timing Trace after signal processing using random plaintext on a Virtex-6 FPGA

4.7 Design Comparison

Table 2 shows a comparison of design execution time and size for each of the primary versions of RSA used on the Virtex-5 FPGA in this research. Version descriptions are detailed in Appendix C. The separate square and multiply circuit (Version B) is used as the baseline circuit for all design comparisons after increasing execution time by 51% over the original. The level power consumption circuit (Version C) performed very well by providing a decent side channel signature that can not be easily attacked while still providing a 24.6% decrease in size and a 50% decrease in execution time. After the *CLOAK* countermeasure was implemented the circuit execution time decreased by 3.7% and increased size by 11.1%, flip flops by 10.1%, and LUTs by 34.3%. Table 3 shows how the designs in this research compare to similar hardware designs being used. When the designs are compared on equal ground the only design that compares was Blum in [21], where the execution times were only estimated and were not validated on an FPGA with a fully functional implementation of RSA.

Table 4 shows a comparison of execution time and size for the primary versions of

Table 2. RSA Design Execution Time and Size for Virtex-5 FPGA

	<i>Version</i>	<i>Execution Time</i> ¹	<i>Utilization</i>	<i>Flip Flops</i>	<i>LUTs</i>
<i>Combined Sq & Mult</i>	A	12.92ms/128μs	80%	10871	21008
<i>Separate Sq & Mult</i>	B	19.5ms/180μs	81%	10873	21014
<i>Level Power</i>	C	9.75ms/180μs ²	65%	9336	14299
<i>Randomized Power 1</i>	D	19.5ms/180μs	70%	10365	17470
<i>Randomized Power 2</i>	E	19.5ms/180μs	85%	11395	24619
<i>CLOAKed</i>	F	18.8ms/174μs	90%	11973	28214

¹ All circuits were tested with a 20MHz clock rate. First execution time uses a normal 512-bit key size and the second uses a short key of 15h.

² RSA Version C was capable of an increased clock rate of 40MHz reducing the execution time by half (19.5ms @ 20MHz).

Table 3. Comparison of RSA Design Execution Time

	<i>Execution Time</i> ¹	<i>Clock Speed</i>	<i>FPGA Type</i>
<i>Hardware w/o CLOAK</i>	19.5ms/180μs	20MHz	Virtex-5
<i>Hardware w/ CLOAK</i>	18.8ms/174μs	20MHz	Virtex-5
<i>Hardware w/o CLOAK</i>	11.7ms/108μs	33.3MHz	Virtex-6
<i>Hardware w/ CLOAK</i>	11.3ms/104.3μs	33.3MHz	Virtex-6
<i>SASEBO-R [47]</i>	1,689ms/16.5ms ²	2MHz	Virtex-II Pro
<i>SASEBO-R [49]</i>	138ms/1.35ms ³	24MHz	Virtex-II Pro
<i>Blum High Radix [21]</i>	2.93ms/110μs ⁴	48MHz	Virtex-5
<i>Blum [21]</i>	9.38ms/350μs ⁴	48MHz	Virtex-5

¹ Execution time given for normal 512-bit key size and small key size.

² Total running time is not given for SASEBO-R so running time is estimated based on modular multiplication execution time of $2.2ms \times 1.5n$ ($n = 512\text{-bits}/n = 5\text{-bits}$)

³ Total running time is not given for SASEBO-R so running time is estimated based on modular multiplication execution time of $\sim 180\mu s \times 1.5n$ ($n = 512\text{-bits}/n = 5\text{-bits}$)

⁴ RSA implementation execution times are estimated and were not validated on an actual FPGA.

RSA used on the Virtex-6 FPGA. Version descriptions are also included in Appendix C. The table shows that the Virtex-6 FPGA design is capable of implementing larger designs, but also at higher frequencies.

Table 4. RSA Design Execution Time and Size for Virtex-6 FPGA

	<i>Version</i>	<i>Execution Time</i> ¹	<i>Utilization</i> ²	<i>Flip Flops</i>	<i>LUTs</i>
<i>Separate Sq & Mult</i>	B	11.7ms/108μs	24%	10876	19307
<i>Level Power</i>	C	11.7ms/108μs	20%	9854	14117
<i>CLOAKed</i>	F	11.3ms/104.3μs	28%	11978	26384

¹ All circuit times are given at 33.3MHz. Initial tests used a 25MHz clock rate, but all circuits were capable of 33.3MHz. First execution time uses a normal 512-bit key size and the second uses a short key of 15h.

² Device utilization shows the number of occupied slices in the entire system to include the AES hardware.

4.8 Results Summary

The initial RSA design implementation (Version A) executed square and multiply operations at the same time, but this caused a dynamic contrast of 50mV between a 1-bit and 0-bit in the key. This extreme circuit leakage is not acceptable in encryption/decryption circuitry. The baseline RSA circuit (Version B) used separate square and multiply operations using equal, but separate multiplier instantiations. This multiply operation in this baseline circuit still used 25mV more peak power than square operations. Efforts were then conducted to implement the polymorphic circuit design using three basic steps.

The first step in polymorphic circuit design development was to flatten the power signature (Version C). This step involved the use of one hardware multiplier instantiation to conduct both multiply and square operations. In this circuit design none of the modular operations correlate with any others in order to identify a square from a multiply, at least not using SEMA. The second step in polymorphic circuit design development took this idea further by randomizing the power signature (Version D and E). This step exploited the fact that Montgomery modular multipliers use a series of add and subtract operations to conduct modular multiplication. By implementing three adders within the single multiplier instantiation the power levels within each

modular multiply operation are varied between the three adders thus complicating the DEMA process by adding three additional power levels for each addition operation. The third and final step in polymorphic circuit design was to randomize the circuit timing (Version F) to eliminate the ability to successfully align the square and multiply operations for DEMA. In this design the square and multiply operations could not be successfully aligned in order to differentiate a square from a multiply operation successfully creation a polymorphic circuit design. Even if the attacker is able to acquire multiple traces using a fixed key, modulus, and plaintext each trace will have have a different side channel signature.

After the Polymorphic CES design was tested thoroughly on the Virtex-5 FPGA the CES was then adapted to run on the Virtex-6 FPGA. This modification was conducted not only to characterize the difference in side channel signatures, but also to create a fully functional Encryption System testbed. The Virtex-6 FPGA implementations had a lower overall signal strength due to the larger chip size and lower power levels, but the *CLOAK* countermeasure was able to successfully obfuscate the EM side channel.

V. Conclusion

This research effort has determined that a polymorphic circuit design can be created that varies circuit power consumption and timing can protect a cryptographic device from EMA attacks. This chapter summarizes the research effort by identifying and summarizing the objectives that were met, presents conclusions that were determined, discusses contributions to the field of study, and establishes the foundations for future achievements.

5.1 Completed Objectives

- ✓ Polymorphic Circuit Design: A Polymorphic Montgomery Modular Multiplier was developed and implemented as a *CLOAK* countermeasure within the modular exponentiation circuitry of a hardware implementation of RSA cryptographic algorithm that was capable of changing the way they function in both power consumption level and circuit timing.
- ✓ Implement RSA with Reconfiguration: The RSA encryption/decryption algorithm was implemented on the Virtex-5 and Virtex-6 FPGAs using VHDL. The FPGA based Encryption System designs were designed as testbed systems on their respective platforms to analyze RSA side channel signatures with and without the polymorphic circuit design implemented.
- ✓ Analyze Side Channel Signatures: The EM side channel signatures of the RSA algorithm was analyzed before and after implementation of the *CLOAK* countermeasure in order to characterize the circuit's ability to resist timing attacks. Culminate in a proof of concept polymorphic circuit design that enhances the systems protective countermeasures by obfuscating circuit operations from SCA.

5.2 Conclusions

This research has determined that a polymorphic circuit design that varies circuit power consumption and timing can protect a cryptographic device from an EMA attacks. This goal was achieved by varying the EM signature of the RSA cryptographic algorithm in such a way that the observer/adversary would be unable to correlate side channel signature to the specific cryptographic functions being executed. Specifically this research designed and implemented a 512-bit hardware version of RSA using modular exponentiation on the Virtex-5 and Virtex-6 FPGAs using VHDL. This cryptographic circuit is capable of obfuscating the vulnerable square and multiply operations within the algorithm in order to effectively *CLOAK* their function from side channel attack. As you can see in Figure 40(b), given two traces using identical inputs on the *CLOAK*ed circuit gives us two completely different side channel signatures for the given square and multiply operations. The standard deviation of the square and multiply operation for the *CLOAK*ed circuit was shown to be 23 times or 2300% greater than the baseline trace. Additionally, the standard deviation of 100 full traces of the *CLOAK*ed circuit were shown to be 18 times or 1800% greater than 100 baseline traces. Hardware obfuscation is very difficult to accomplish and in doing so there can be considerable size and performance penalty to a more secure design. For this research there was an increase in execution time of 51% when the square and multiply operations were separated in time, but this measure was pivotal in creating a baseline circuit to obfuscate circuit functions. Once the *CLOAK* countermeasure was implemented the execution time decreased by 3.7% and size increased by 11.1%. In the end we were successfully able to obfuscate the hardware functions of the cryptographic algorithm.

5.3 Contributions

- Developed a 512-bit hardware implementation of the RSA public-key encryption/decryption algorithm using modular exponentiation to be used on the Virtex-5 or Virtex-6 FPGAs.
- Incorporated a hardware implementation of the AES symmetric-key encryption algorithm on the Virtex-5 and Virtex-6 FPGAs.
- Created an FPGA based Encryption System testbed, named Polymorphic CES, to be used for side channel analysis on hardware and software based cryptographic algorithms. Baseline system incorporates a hardware implementation of AES and RSA as well as a software implementation of AES. On the Virtex-5 Encryption System version the hardware instantiations of RSA and AES are implemented separately. The Virtex-6 Encryption System version combines both hardware instantiations of RSA and AES into one fully functional design capable of running multiple versions of each.
- Designed, implemented, and tested the *CLOAK* countermeasure.
- Conducted EM side channel analysis of hardware based RSA before and after implementation of the *CLOAK* countermeasure.

5.4 Future Work

- Randomizing the order of execution for square and multiply operations within each step of the modular exponentiation process.
- Development of a random RSA key generation module using Java within Inspector or C-code on the PowerPC in order to be incorporated in the trace acquisition process of our FPGA based Encryption System testbed.

- Development of custom DPA/DEMA module within Inspector or Matlab capable of automating the key extraction process for a hardware based RSA trace.
- Addition of circuitry for blinding the exponent.
- Addition of noise maker circuitry, such as oscillators, to further obfuscate the EM signal from the adversary. The MicroBlaze served this function in the Virtex-6 investigations. Note: Oscillators need to operate at the same frequency as the RSA circuit otherwise they can be easily filtered out of the trace during signal processing.
- The latest research in Digital Fingerprinting of hardware allows for the generation of hardware specific keys linked to specific pieces of hardware. These keys can be used to provide a private key for decryption operations or digital signatures in algorithms such as AES or RSA. Using such a hardware specific key for RSA would require a little more signal processing due to the RSA key generation requirements. This key could also be used to drive a polymorphic function that would essentially disable the circuit functionality if attempts were made to copy the system *bitstream* to another FPGA.

5.5 Summary

This research provides contributions in the information protection and tamper protection of MCTs. The documented results and analysis of the *CLOAK* countermeasure confirms that it successfully obfuscates the EM side channel of the hardware based RSA circuitry on the Virtex-5 as well as the Virtex-6 FPGA. Also, the development of the experimental testbed unit provides the functionality to test encryption system vulnerabilities and countermeasures on an FPGA based system while having complete access to the source code.

Appendix A. Encryption System Flowcharts

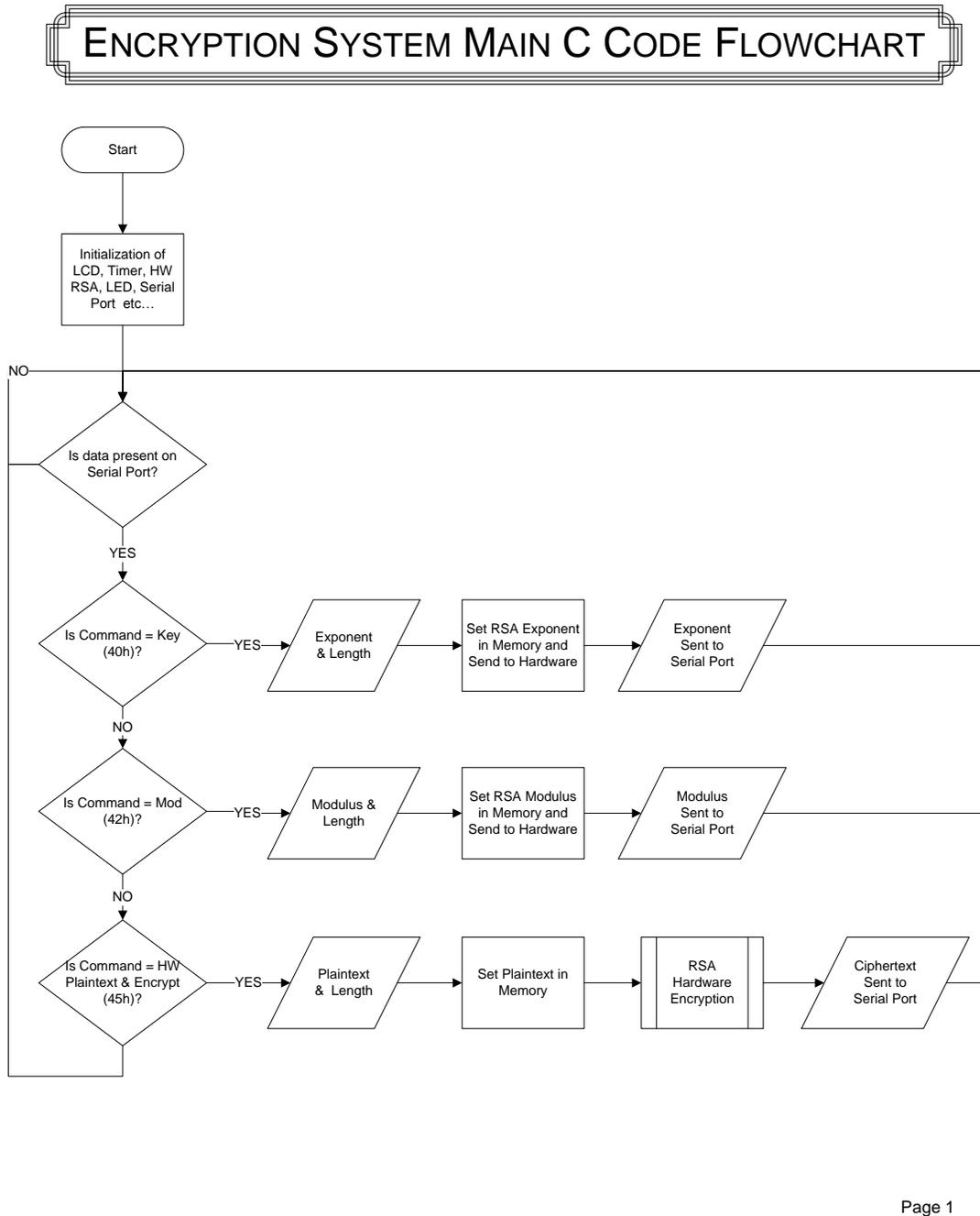


Figure 50. RSA Encryption System Flowchart on the Virtex-5 FPGA

ENCRYPTION SYSTEM MAIN C CODE FLOWCHART

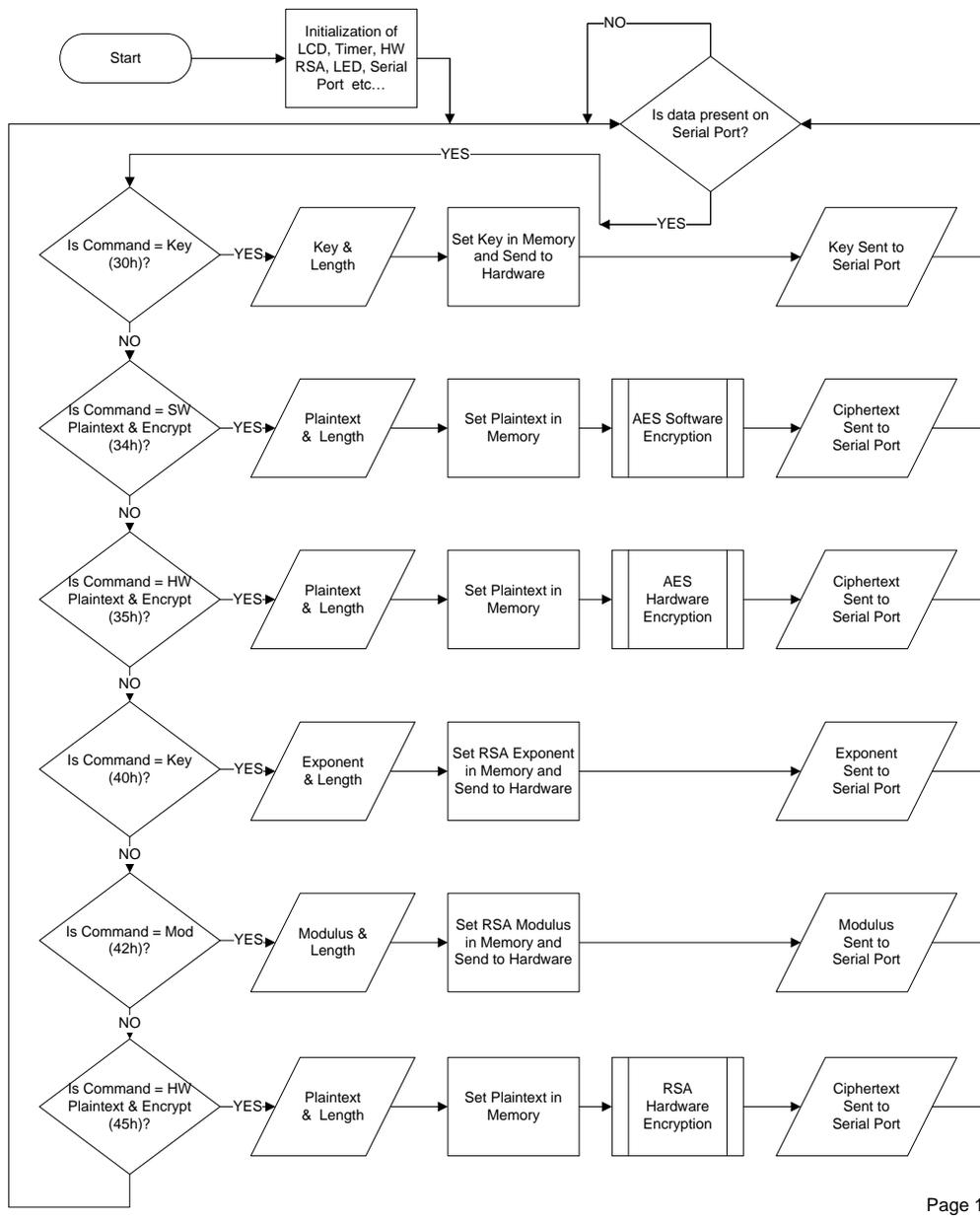


Figure 51. Encryption System Flowchart on the Virtex-6 FPGA

AES MAIN C CODE FLOWCHART

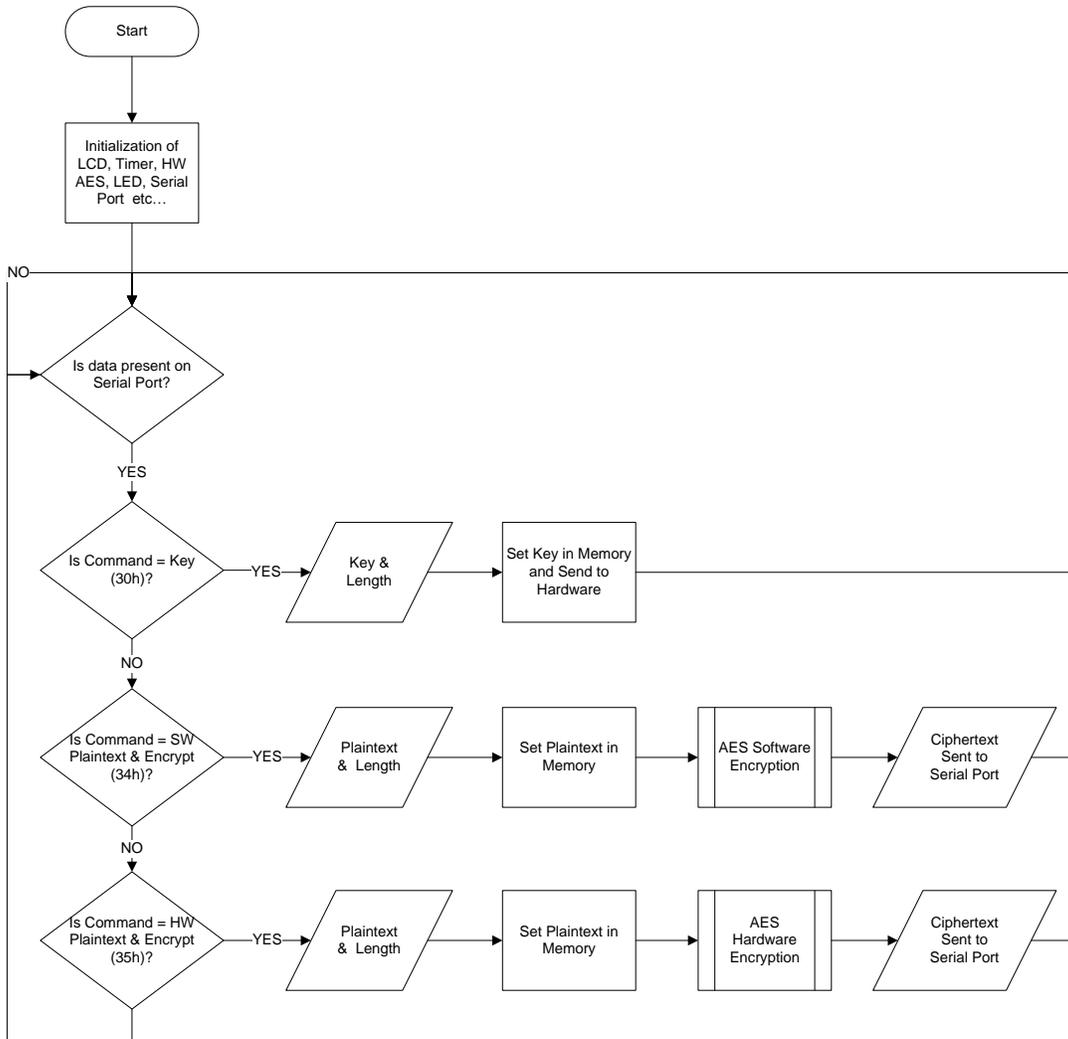


Figure 52. AES Encryption System Flowchart on the Virtex-5 FPGA

Appendix B. RSA Traces

2.1 Virtex-5 FPGA Traces

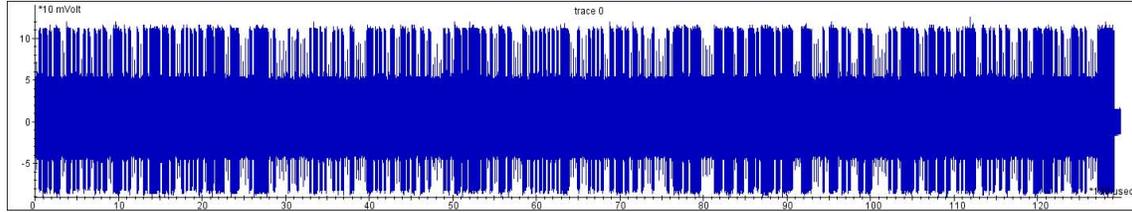


Figure 53. Full 512-bit Combined Square and Multiply Trace - RSA Version A

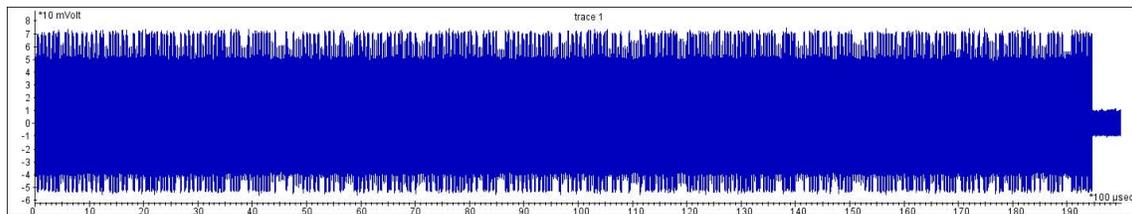


Figure 54. Full 512-bit Separate Square and Multiply Trace - RSA Version B

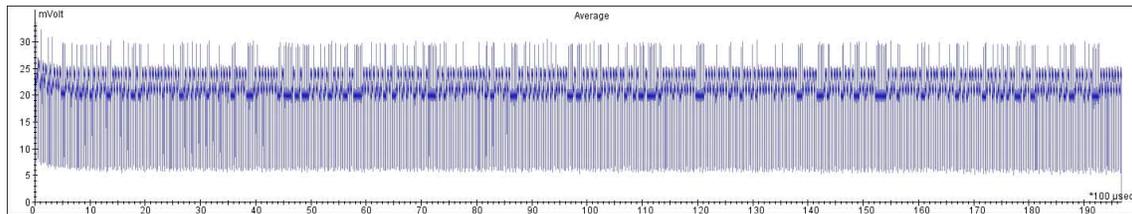


Figure 55. Full 512-bit Separate Square and Multiply Trace using constant timing - RSA Version 2B

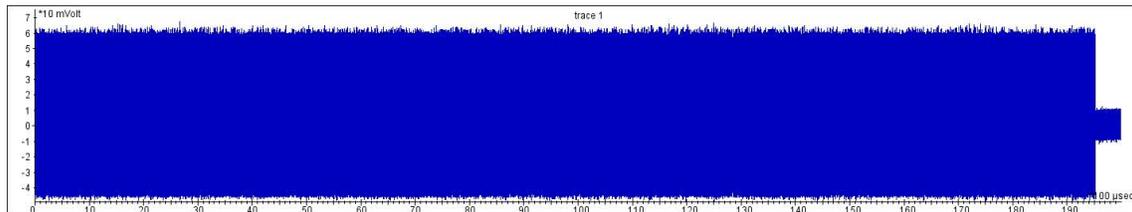


Figure 56. Full 512-bit Level Power Consumption Trace - RSA Version C

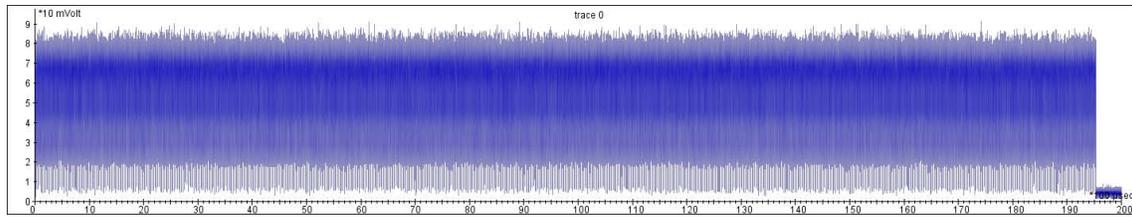


Figure 57. Full 512-bit Level Power Consumption Trace after signal processing - RSA Version C

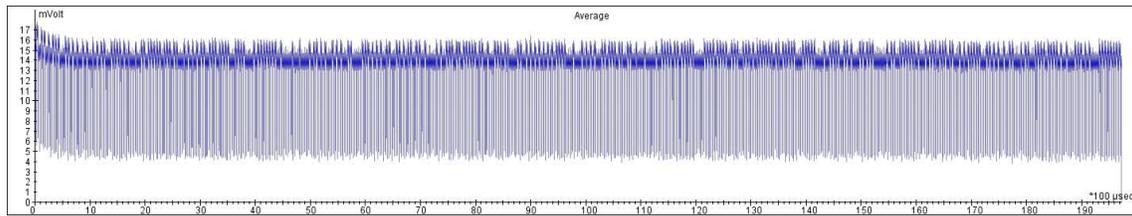


Figure 58. Full 512-bit Level Power Consumption using constant timing - RSA Version 2C

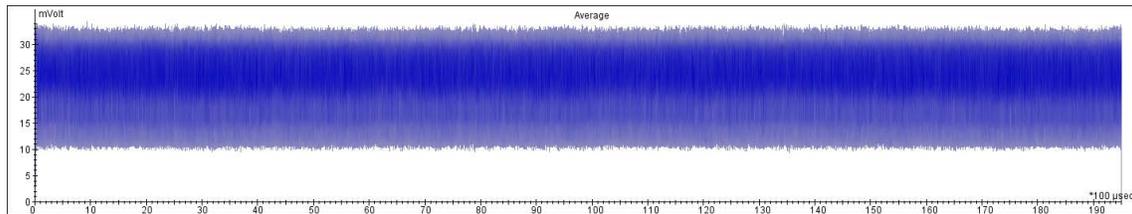


Figure 59. Full 512-bit Level Power Consumption Trace after elastic alignment and average - RSA Version C

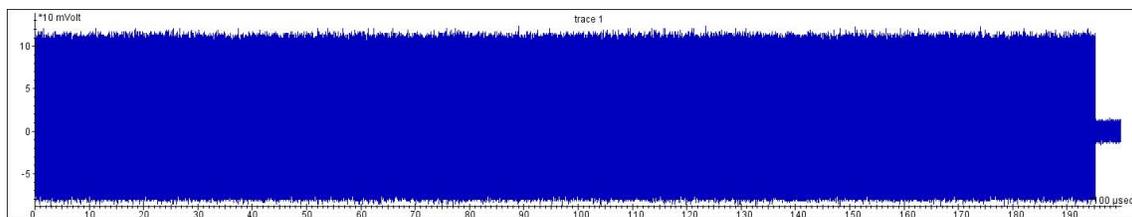


Figure 60. Full 512-bit Randomized Power Consumption Trace - RSA Version E

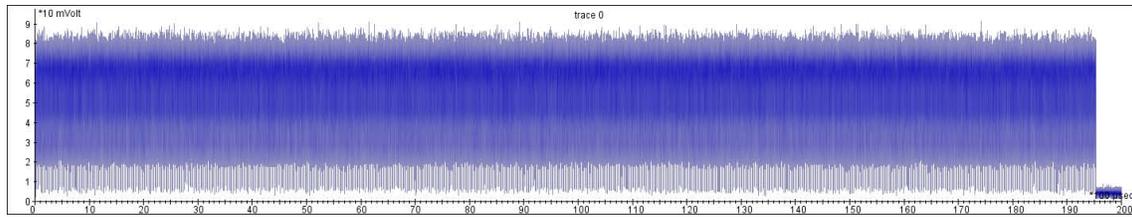


Figure 61. Full 512-bit Randomized Power Consumption Trace after signal processing - RSA Version E

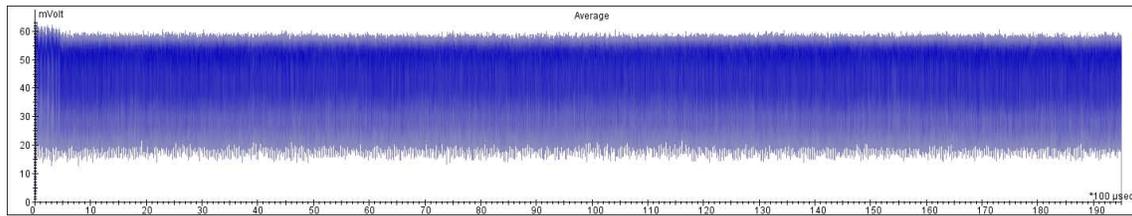


Figure 62. Full 512-bit Randomized Power Consumption Trace after elastic alignment and average - RSA Version E

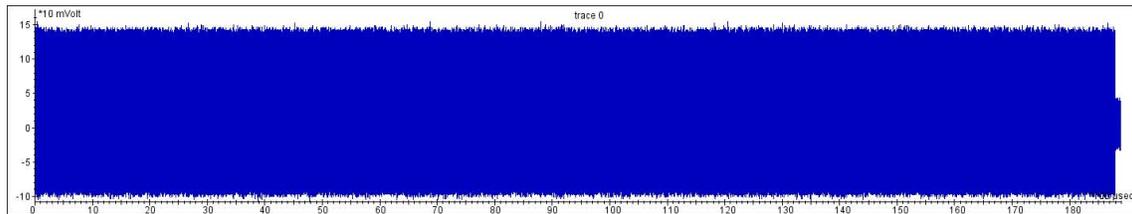


Figure 63. Full 512-bit Randomized Power and Timing Trace - RSA Version F

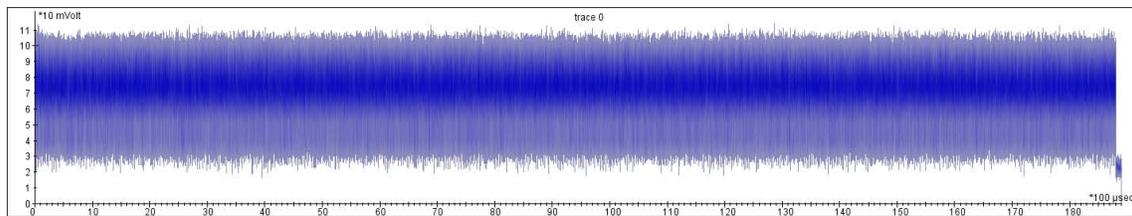


Figure 64. Full 512-bit Randomized Power and Timing Trace after signal processing - RSA Version F

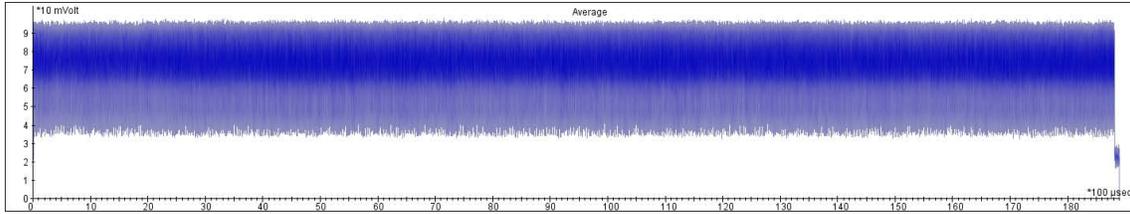
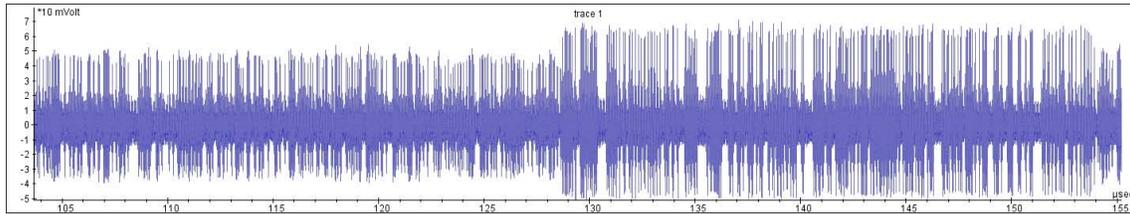
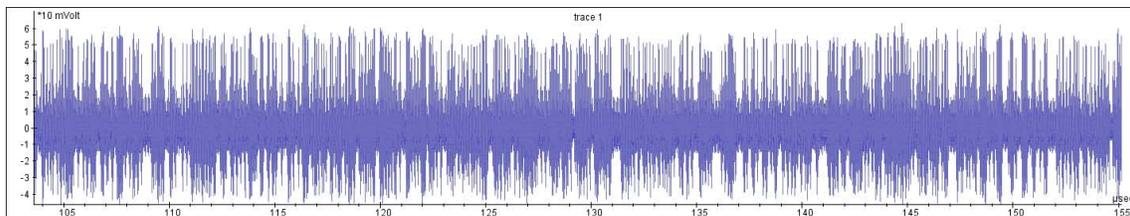


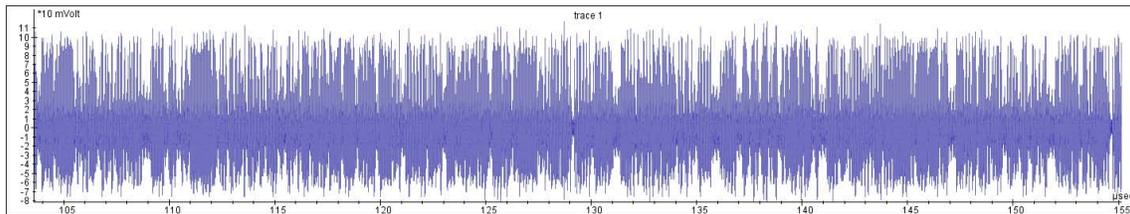
Figure 65. Full 512-bit Randomized Power and Timing Trace after elastic alignment and average - RSA Version F



(a) Baseline Circuit EM signature - RSA Version B



(b) Level Power Circuit EM signature - RSA Version C



(c) Randomized Power Circuit EM signature - RSA Version E

Figure 66. Comparison of EM signals for square and multiply operations

2.2 Virtex-6 FPGA Traces

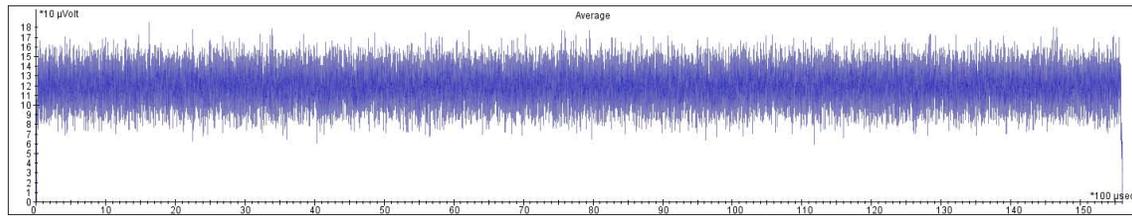


Figure 67. Full 512-bit Separate Square and Multiply Trace after signal processing using fixed plaintext - RSA Version B on a Virtex-6 FPGA

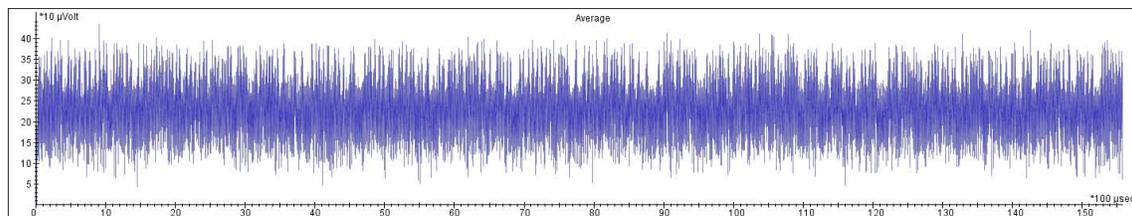


Figure 68. Full 512-bit Separate Square and Multiply Trace after signal processing using a different clock and fixed plaintext - RSA Version 4B on a Virtex-6 FPGA

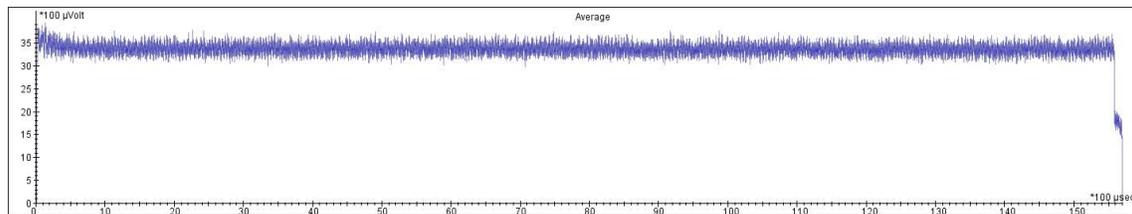


Figure 69. Full 512-bit Separate Square and Multiply Trace after signal processing using random plaintext - RSA Version B on a Virtex-6 FPGA

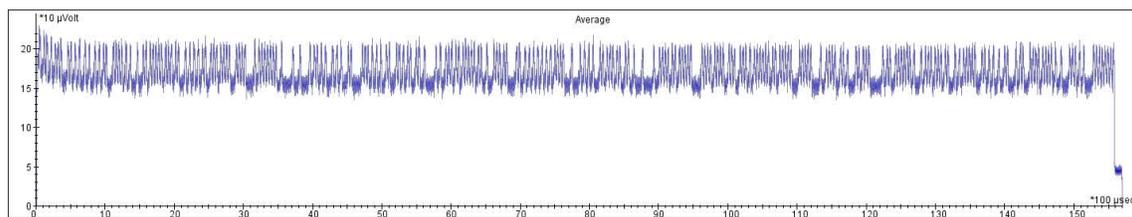


Figure 70. Full 512-bit Separate Square and Multiply Trace after signal processing using a different clock and random plaintext - RSA Version 4B on a Virtex-6 FPGA

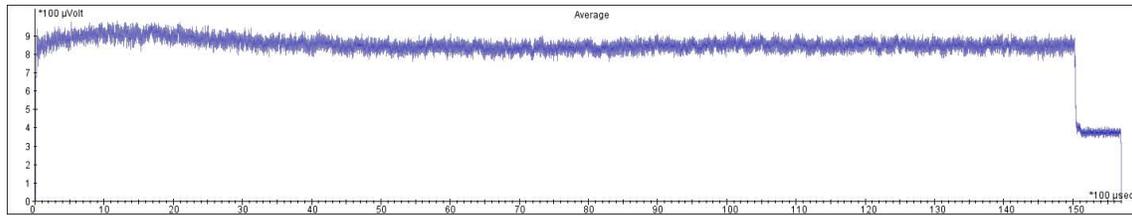


Figure 71. Full 512-bit Separate Square and Multiply Trace after signal processing using fixed plaintext - RSA Version F on a Virtex-6 FPGA

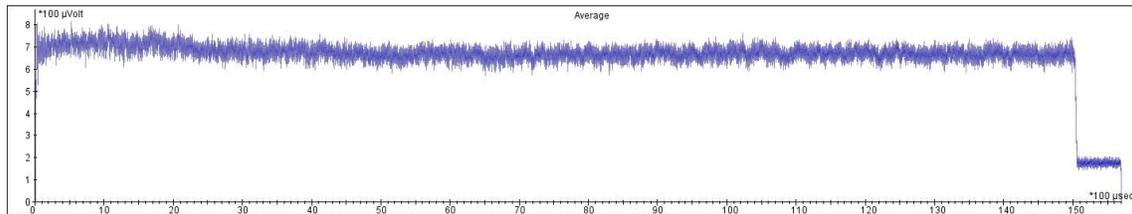


Figure 72. Full 512-bit Separate Square and Multiply Trace after signal processing using a different clock and fixed plaintext - RSA Version 4F on a Virtex-6 FPGA

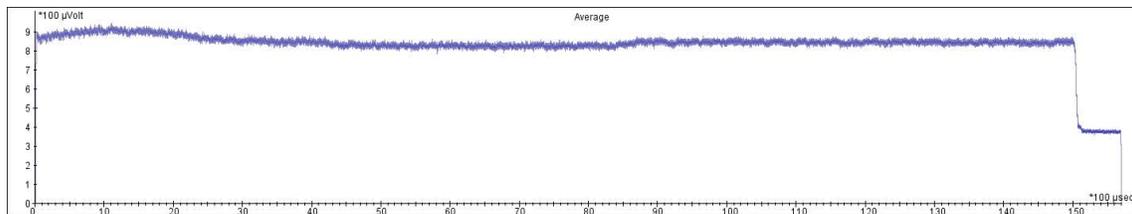


Figure 73. Full 512-bit Separate Square and Multiply Trace after signal processing using random plaintext - RSA Version F on a Virtex-6 FPGA

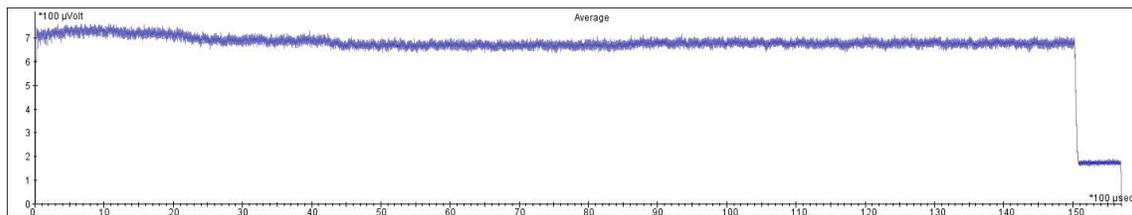


Figure 74. Full 512-bit Separate Square and Multiply Trace after signal processing using a different clock and random plaintext - RSA Version 4F on a Virtex-6 FPGA

Appendix C. Version Control

AES Version A (aes_hw_v1_00_a): Working version of the non-iterative 128-bit Hardware AES code.

AES Version B (aes_hw_v1_00_b): Working version of the iterative 128-bit Hardware AES code.

RSA Version A (rsa_hw_v1_00_a): First working version of 512-bit Hardware RSA code. Square and Multiply are executed at the same time.

RSA Version B (rsa_hw_v1_00_b): Second iteration of the 512-bit Hardware RSA code. Square and Multiply are executed at the different times, but using two different instantiations for the square and multiply operations.

RSA Version 2B (rsa_hw_v2_00_b): Same as the previous except the multiply operations all have the same execution time.

RSA Version 4B (rsa_hw_v4_00_b): Same as Version B except it uses an external clock input so the MicroBlaze on the Virtex-6 can run at a different clock frequency.

RSA Version C (rsa_hw_v1_00_c): Third iteration of the 512-bit Hardware RSA code. Square and Multiply are executed at the different times, and uses only one instantiation of the multiplier for the square and multiply operations. This is the baseline starting point for implementing the CLOAK countermeasure.

RSA Version 2C (rsa_hw_v2_00_c): Same as the previous except the multiply operations all have the same execution time.

RSA Version 3C (rsa_hw_v3_00_c): Same as Version C except it uses runs 1024-Bit Hardware RSA Code on the Virtex-6.

RSA Version 4C (rsa_hw_v4_00_c): Same as Version C except it uses an external clock input so the MicroBlaze on the Virtex-6 can run at a different clock frequency.

RSA Version D (rsa_hw_v1_00_d): Fourth iteration of the 512-bit Hardware RSA code. Square and Multiply are executed at the different times, and uses only one instantiation of the multiplier for the square and multiply operations. Each multiplier implements two adder circuits (one being the VHDL add operator and the second being a ripple carry adder). This is the first implementation of the CLOAK countermeasure

RSA Version E (rsa_hw_v1_00_e): Fifth iteration of the 512-bit Hardware RSA code. Square and Multiply are executed at the different times, and uses only one instantiation of the multiplier for the square and multiply operations. Each multiplier implements three adder circuits (one being the VHDL add operator, second being a ripple carry adder, and third being a carry look-ahead adder). This is the first implementation of the CLOAK countermeasure.

RSA Version 2E (rsa_hw_v2_00_e): Same as the previous except the multiply operations all have the same execution time.

RSA Version F (rsa_hw_v1_00_f): Sixth iteration of the 512-bit Hardware RSA code. Square and Multiply are executed at the different times, and uses only one instantiation of the multiplier for the square and multiply operations. Each multiplier implements three adder circuits (one being the VHDL add operator, second being a ripple carry adder, and third being a carry look-ahead adder). Additionally, this implementation randomizes the timing of each multiply operation to a much greater extent than previous versions. Previous versions varied execution based on the distance between the first and last 1 bit within

the multiplier. This is the second and final implementation of the CLOAK countermeasure.

RSA Version 4F (rsa_hw_v4_00_f): Same as Version F except it uses an external clock input so the MicroBlaze on the Virtex-6 can run at a different clock frequency.

Appendix D. Data Sheets

This section contains selected data sheet information for primary data collection devices.

4.1 Willtek 1207 Inductive Probe

The Willtek 1207 Inductive Probe [17] is an active probe for contactless measurements. The probe is designed to the following specifications:

- Frequency Range - 50 *MHz* to 4 *GHz* Specified, 10 *MHz* to 6 *GHz* Usable
- Gain - 20 *dB* @ 1 *GHz*

4.2 Inspector EM Probe

The Inspector EM Probe [12] is an active probe for contactless measurements. Device also contains a hardware device, called an *EM Shield or Field Deflector*, designed to reduce unwanted EM Signals. The probe is designed to the following specifications:

- Sensitivity @ 1 *MHz* - 20 *MHz*/1 μT
- Bandwidth - 1 *GHz*
- Resolution - 1 mm^2

4.3 Lecroy WavePro 725Zi Oscilloscope

The Lecroy WavePro 725Zi Digital Storage Oscilloscope (DSO) [13] is a 2.5 *GHz*, 20 *GS/s*, 4 *Ch*, 10 *Mpts/Ch* DSO with a 15.4" WXGA Color Display.

4.4 Agilent E3631A DC Power Supply

The Agilent E3631A [10] is a triple output DC Power Supply.

4.5 Virtex-5 FPGA

The Virtex-5 FXT FPGA ML507 Evaluation Platform [8].

4.6 Virtex-6 FPGA

The Virtex-6 LXT FPGA ML605 Evaluation Platform [9].

Bibliography

- [1] “Military Critical Technologies List”. URL <http://www.dtic.mil/mctl/MCTL.html>.
- [2] “IEEE Standard VHDL Language Reference Manual.” *ANSI/IEEE Std 1076-1993*, 1994.
- [3] “Efficient Shift Registers, LFSR Counters, and Long Pseudo-Random Sequence Generators”. XAPP 052 (v1.1), 1996. URL http://www.xilinx.com/support/documentation/application_notes/xapp052.pdf.
- [4] “Specification for the Advanced Encryption Standard (AES)”. Federal Information Processing Standards Publication 197, 2001. URL <http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf>.
- [5] “Federal Information Security Management Act of 2002 FISMA (2002)”. Title III of the E-Government Act - Information Security, 44 U.S.C. Sec. 3541, December 2002.
- [6] “Personal Identity Verification (PIV) of Federal Employees and Contractors”. Federal Information Processing Standards Publication 201-1, March 2006.
- [7] “FPGA Run-Time Reconfiguration: Two Approaches”. WP-01055-1.0, March 2008. URL <http://www.altera.com/literature/wp/wp-01055-fpga-run-time-reconfiguration.pdf>.
- [8] “ML505/ML506/ML507 Evaluation Platform User Guide”. UG347 (v3.1.1), October 2009. URL http://www.xilinx.com/support/documentation/boards_and_kits/ug347.pdf.
- [9] “ML605 Reference Design User Guide”. UG535 (v1.0), September 2009. URL http://www.xilinx.com/support/documentation/boards_and_kits/ug535.pdf.
- [10] “Agilent E363xA Series DC Power Supply”, 2010. URL <http://cp.literature.agilent.com/litweb/pdf/5968-9726EN.pdf>.
- [11] “Device Package User Guide”. UG112 (v3.6), September 2010. URL http://www.xilinx.com/support/documentation/user_guides/ug112.pdf.
- [12] “Inspector Brochure”, 2010. URL http://www.riscure.com/fileadmin/images/Docs/Inspector_brochure.pdf.
- [13] “Lecroy WavePro 7 Zi Series Oscilloscope”, 2010. URL http://www.lecroy.com/files/pdf/LeCroy_WavePro_7_Zi_Datasheet.pdf.

- [14] “PowerPC Processor Reference Guide”. UG011 (v1.3), January 2010. URL http://www.xilinx.com/support/documentation/user_guides/ug011.pdf.
- [15] “Virtex-5 FPGA Users Guide”. UG190 (v5.3), May 2010. URL http://www.xilinx.com/support/documentation/user_guides/ug190.pdf.
- [16] “Virtex-6 FPGA Users Guides”, 2010. URL <http://www.xilinx.com/support/documentation/virtex-6.htm>.
- [17] “Willtek 1207 Inductive Probe Data Sheet”, 2010. URL http://www.willtek.com/mediadb/global/download/literature/datasheets/9100/1207_ds_1-1008_en.pdf.
- [18] Agrawal, Dakshi, Bruce Archambeault, Josyula R. Rao, and Pankaj Rohatgi. “The EM Side-Channel(s)”. *Cryptographic Hardware and Embedded Systems, CHES’02*, volume LNCS 2523, 29–45. Springer-Verlag Berlin Heidelberg, 2002.
- [19] Aumuller, C, P Bier, W Fischer, P Hofreiter, and JP Seifert. “Fault attacks on RSA with CRT: Concrete results and practical countermeasures”. *CHES 2002*, LNCS 2523, 260–275. Springer-Verlag Berlin Heidelberg, 2002.
- [20] Benson, Pam and Michael Sefanov. “Iraqi insurgents hacked Predator drone feeds, U.S. official indicates”, december 2009. URL <http://edition.cnn.com/2009/US/12/17/drone.video.hacked/index.html>.
- [21] Blum, Thomas and Christof Paar. “High-Radix Montgomery Modular Exponentiation on Reconfigurable Hardware”. *IEEE Trans. Comput.*, 50:759–764, July 2001.
- [22] Cady, Camdon. *Static and Dynamic Component Obfuscation on Reconfigurable Devices*. Master’s thesis, Air Force Institute of Technology, 2010.
- [23] Carlier, Vincent, Hervé Chabanne, Emmanuelle Dottax, and Hervé Pelletier. “Electromagnetic side channels of an FPGA implementation of AES”. *Cryptology ePrint Archive*, (145), 2004. URL <http://eprint.iacr.org/2004/145.pdf>.
- [24] Chakraborty, R.S. and S. Bhunia. “Hardware protection and authentication through netlist level obfuscation”. *Computer-Aided Design, 2008. ICCAD 2008. IEEE/ACM International Conference on*, 674–677. 2008.
- [25] Chakraborty, R.S. and S. Bhunia. “HARPOON: An Obfuscation-Based SoC Design Methodology for Hardware Protection”. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 28(10):1493–1502, 2009.
- [26] Chaum, David. “Blind Signatures for Untraceable Payments”. *Advances in Cryptology Proceedings of Crypto 82*, 199–203. Springer-Verlag Berlin Heidelberg, 1983.

- [27] Chikofsky, E.J. and II Cross, J.H. “Reverse engineering and design recovery: a taxonomy”. *Software, IEEE*, 7(1):13–17, Jan 1990.
- [28] Collberg, Christian S., Ieee Computer Society, Clark Thomborson, and Senior Member. “Watermarking, Tamper-Proofing, and Obfuscation - Tools for Software Protection”. *IEEE Transactions on Software Engineering*, 28:735–746, 2002.
- [29] Compton, Katherine and Scott Hauck. “Reconfigurable computing: a survey of systems and software”. *ACM Computing Surveys*, 34(2):171–210, 2002.
- [30] Crouch, J.W., H.J. Patel, Y.C. Kim, J.T. McDonald, and T.C. Kim. “Creating digital fingerprints on commercial field programmable gate arrays”. *ICECE Technology, 2008. FPT 2008. International Conference on*, 345 –348. 2008.
- [31] f. Dhem, J., F. Koeune, P.-A. Leroux, P. Mestr, J.-J. Quisquater, and J. l. Willems. “A practical implementation of the timing attack”. *Smart Card Research and Applications. Third International Conference, CARDIS'98*, volume LNCS 1820, 104–113. Springer-Verlag Berlin Heidelberg, 2000.
- [32] Estrin, G., B. Bussell, R. Turn, and J. Bibb. “Parallel Processing in a Re-structurable Computer System”. *Electronic Computers, IEEE Transactions on*, EC-12(6):747 –755, 1963.
- [33] Gassend, Blaise, Daihyun Lim, Dwaine Clarke, Marten van Dijk, and Srinivas Devadas. “Identification and authentication of integrated circuits: Research Articles”. *Concurr. Comput. : Pract. Exper.*, 16:1077–1098, September 2004. ISSN 1532-0626.
- [34] Gorman, Siobhan, August Cole, Yochi Drezen, and Evan Perez. “Computer Spies Breach Fighter-Jet Project.” *Wall Street Journal - Eastern Edition*, 92:A1 – A2, 2009. URL <http://online.wsj.com/article/SB124027491029837401.html>.
- [35] Kaiser, K.L. *Electromagnetic compatibility handbook*. Electrical engineering handbook series. CRC Press, 2005.
- [36] Kaps, Jens-Peter and Rajesh Velegalati. “DPA resistant AES on FPGA using partial DDL”. *Proceedings - IEEE Symposium on Field-Programmable Custom Computing Machines, FCCM 2010*, 273 – 280. 2010.
- [37] Keromitis, Aggelos. “Generating RSA keys”, August 2010. URL www.cipherspace.org/rsa/rsa-keygen.html.
- [38] Kocher, P. C. and N. Koblitz. “Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems”. *Advances in Cryptology - CRYPTO'96*, volume LNCS 1109, 104–113. Springer-Verlag Berlin Heidelberg, 1996.

- [39] Kocher, Paul, Joshua Jaffe, and Benjamin Jun. “Differential Power Analysis”. *Advances in Cryptology - CRYPTO’99*, volume LNCS 1666, 388–397. Springer-Verlag Berlin Heidelberg, 1999.
- [40] Kocher, Paul C. “Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems”. *CRYPTO*, 104–113. 1996.
- [41] Lach, J., W.H. Mangione-Smith, and M. Potkonjak. “Fingerprinting techniques for field-programmable gate array intellectual property protection”. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 20(10):1253–1261, October 2001.
- [42] L’Ecuyer, Pierre. “Random Numbers for Simulation”. *Communications of the ACM*, 33(10):85–97, 1990.
- [43] Mamiya, Hideyo, Atsuko Miyaji, and Hiroaki Morimoto. “Efficient Countermeasures against RPA, DPA, and SPA”. *Cryptographic Hardware and Embedded Systems - CHES 2004*, volume 3156 of *Lecture Notes in Computer Science*, 243–319. Springer-Verlag Berlin Heidelberg, 2004.
- [44] Mangard, Stefan. “Hardware Countermeasures against DPA A Statistical Analysis of Their Effectiveness”. *Topics in Cryptology CT-RSA 2004*, volume 2964 of *LNCS 2964*, 222–235. Springer-Verlag Berlin Heidelberg, 2004.
- [45] Menezes, Alfred J., Scott A. Vanstone, and Paul C. Van Oorschot. *Handbook of Applied Cryptography*. CRC Press, Inc., 1996.
- [46] Messerges, Thomas, Ezzy Dabbish, and Robert Sloan. “Power Analysis Attacks of Modular Exponentiation in Smartcards”. *Cryptographic Hardware and Embedded Systems - CHES 1999*, volume 1717 of *Lecture Notes in Computer Science*, 724–724. Springer Berlin / Heidelberg, 1999.
- [47] Miyamoto, A., N. Homma, T. Aoki, and A. Satoh. “SPA against an FPGA-Based RSA Implementation with a High-Radix Montgomery Multiplier”. *Circuits and Systems, 2007. ISCAS 2007. IEEE International Symposium on*, 1847–1850. May 2007.
- [48] Miyamoto, A., N. Homma, T. Aoki, and A. Satoh. “Enhanced power analysis attack using chosen message against RSA hardware implementations”. *Circuits and Systems, 2008. ISCAS 2008. IEEE International Symposium on*, 3282–3285. may 2008.
- [49] Miyamoto, A., N. Homma, T. Aoki, and A. Satoh. “Evaluation of Simple/Comparative Power Analysis against an RSA ASIC implementation”. *Circuits and Systems, 2009. ISCAS 2009. IEEE International Symposium on*, 2918–2921. May 2009.

- [50] Mulder, E. De, P. Buysschaert, S. B. rs, P. Delmotte, B. Preneel, and I. Verbauwhede. “Electromagnetic analysis attack on an FPGA implementation of an elliptic curve cryptosystem”. In *EUROCON: Proceedings of the International Conference on “Computer as a tool”, 1879–1882*. 2005.
- [51] Örs, Siddika, Elisabeth Oswald, and Bart Preneel. “Power-Analysis Attacks on an FPGA First Experimental Results”. *Cryptographic Hardware and Embedded Systems - CHES 2003*, volume 2779 of *Lecture Notes in Computer Science*, 35–50. Springer Berlin / Heidelberg, 2003.
- [52] Patel, H., Yong Kim, J.T. McDonald, and L. Starman. “Increasing stability and distinguishability of the digital fingerprint in FPGAs through input word analysis”. *Field Programmable Logic and Applications, 2009. FPL 2009. International Conference on*. 31 2009.
- [53] Porter, R., S.J. Stone, Y.C. Kim, J.T. McDonald, and L.A. Starman. “Dynamic Polymorphic Reconfiguration for anti-tamper circuits”. *Field Programmable Logic and Applications, 2009. FPL 2009. International Conference on*, 493–497. Sept. 2009.
- [54] Porter, Roy. *Critical Technology Tamper Protection Through Dynamic Polymorphic Reconfiguration*. Master’s thesis, Air Force Institute of Technology, 2009.
- [55] Quisquater, Jean-Jacques and David Samyde. “ElectroMagnetic Analysis (EMA): Measures and Counter-Measures for Smart Cards”. *Proceedings of the International Conference on Research in Smart Cards, E-SMART ’01*, volume LNCS 2140, 200–210. Springer-Verlag Berlin Heidelberg, 2001.
- [56] Rivest, R. L., A. Shamir, and L. Adleman. “A Method for Obtaining Digital Signatures and Public-Key Cryptosystems”. *Communications of the ACM*, 21:120–126, 1978.
- [57] Schneier, Bruce. *Applied cryptography (2nd ed.): protocols, algorithms, and source code in C*. John Wiley & Sons, Inc., New York, NY, USA, 1995. ISBN 0-471-11709-9.
- [58] Sekanina, L. “Evolutionary design of gate-level polymorphic digital circuits”. *Applications of Evolutionary Computing*, volume LNCS 3449, 185–194. Springer-Verlag Berlin Heidelberg, 2005.
- [59] Skorobogatov, Sergei. *Semi-invasive attacks - A new approach to hardware security analysis*. Technical report, University of Cambridge, Computer Laboratory, Technical Report UCAM-CL-TR-630, 2005.
- [60] Stoica, A., R.S. Zebulum, X. Guo, D. Keymeulen, M.I. Ferguson, and V. Duong. “Taking evolutionary circuit design from experimentation to implementation: some useful techniques and a silicon demonstration.” *IEE Proceedings-Computers and Digital Techniques*, 151(4):295 – 300, 2004.

- [61] Stone, S.J., R. Porter, Y.C. Kim, and J.V. Paul. “A dynamically reconfigurable Field Programmable Gate Array hardware foundation for security applications”. *ICECE Technology, 2008. FPT 2008. International Conference on*, 305–308. Dec. 2008.
- [62] Suh, G. Edward and Srinivas Devadas. “Physical unclonable functions for device authentication and secret key generation”. *Proceedings of the 44th annual Design Automation Conference, DAC '07*, 9–14. ACM, 2007. ISBN 978-1-59593-627-1.
- [63] Wang, Yi, Jussipekka Leiwo, Thambipillai Srikanthan, and Luo Jianwen. “An Efficient Algorithm for DPA-resistant RSA”. *Circuits and Systems, 2006. APC-CAS 2006. IEEE Asia Pacific Conference on*, 1659–1662. 2006.
- [64] White, J.L., M.-J. Chung, A.S. Wojcik, and T.E. Doom. “Efficient algorithms for subcircuit enumeration and classification for the module identification problem”. 519–522. 2001.
- [65] Wollinger, T., C. Paar, Y.K. Cheung, P., G.A. Constantinides, and J.T. de Sousa. “How secure are FPGAs in cryptographic applications?.” volume LNCS 2778, 91–100. Springer-Verlag Berlin Heidelberg, 2003.
- [66] Zhuang, Xiaotong, Tao Zhang, Hsien-Hsin S. Lee, and Santosh Pande. “Hardware assisted control flow obfuscation for embedded processors”. *CASES 2004: International Conference on Compilers, Architecture, and Synthesis for Embedded Systems*, 292–302. 2004.

Vita

First Lieutenant Jeffrey L. Falkinburg entered the Air Force in 1998 as an Airman First Class. After completing Basic Military Training he proceeded to Keesler AFB, MS for Technical Training School. He began his first job as a Computer Maintenance Technician in the 34th Combat Communications Squadron, Tinker AFB, OK. There he was deployed to Eskan Village, Kingdom of Saudi Arabia in support of Operation Southern Watch in 1999. He graduated from the Community College of the Air Force in 2001 with an Associates of Applied Science in Electronic Systems Technology. In 2002, he went back to Keesler AFB, MS to retrain into Communications - Computer Systems Control where he then transferred to Offutt AFB, NE. He worked as a Circuit Actions and Activations Technician in the 55th Communications Squadron, Offutt AFB, NE. He graduated from Airmen Leadership School and was inducted as a Non-Commissioned Officer in the grade of Staff Sergeant. He then applied and was accepted into the Airmen Education and Commissioning Program in 2004 and was transferred to Detachment 470, Omaha, NE where he entered the Air Force Reserve Officer Training Corps program. He graduated with his Bachelor of Science in Computer Engineering from the University of Nebraska - Lincoln and received his Commission in May 2007 and became a Computer Systems Developmental Engineer.

Second Lieutenant Jeffrey L. Falkinburg then transferred to Eglin AFB, FL where he worked for Air Force Research Laboratories (AFRL) - Munitions Directorate as a Scene Generations Computer Engineer. While working at AFRL he was a program manager in charge of \$2M+ effort creating the “first ever” maritime simulation framework to support the Navy Non-line of Sight Precision attack Munition (NLOS PAM) missile system. In addition, while stationed at Eglin he was competitively chosen for AngelFire deployment to Al Asad, Iraq in support of Operation Iraqi Freedom

providing real-time wide area persistent ISR for the U.S. Marine Corp. Finally, First Lieutenant Jeffrey L. Falkinburg then applied and was accepted into the Air Force Institute of Technology (AFIT) in 2009. He was then transferred to Wright Patterson AFB, OH where he then graduated with a Master's of Science Degree in Computer Engineering in March 2011. Upon graduation, Lt Falkinburg will be assigned to AFRL - Human Effectiveness Directorate at Wright Patterson AFB, OH.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. **PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.**

1. REPORT DATE (DD-MM-YYYY) 24-03-2011		2. REPORT TYPE Master's Thesis		3. DATES COVERED (From — To) Aug 2009 — Mar 2011			
4. TITLE AND SUBTITLE DYNAMIC POLYMORPHIC RECONFIGURATION TO EFFECTIVELY "CLOAK" A CIRCUIT'S FUNCTION				5a. CONTRACT NUMBER			
				5b. GRANT NUMBER			
				5c. PROGRAM ELEMENT NUMBER			
				5d. PROJECT NUMBER ENG 10-326			
				5e. TASK NUMBER			
6. AUTHOR(S) Jeffrey L. Falkinburg, 1st, USAF; jeffrey.falkinburg@us.af.mil				5f. WORK UNIT NUMBER			
				7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/EN) 2950 Hobson Way WPAFB OH 45433-7765		8. PERFORMING ORGANIZATION REPORT NUMBER AFIT/GCE/ENG/11-03	
				9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Dr. Robert L. Herklotz Program Manager - Information Operations and Security Air Force Office of Scientific Research (AFOSR/RSL) 875 N. Randolph Street, Suite 325, Room 3112 Arlington, VA 22203-1768 (703) 696-6565; robert.herklotz@afosr.af.mil		10. SPONSOR/MONITOR'S ACRONYM(S) AFOSR/RSL	
12. DISTRIBUTION / AVAILABILITY STATEMENT APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED. This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.							
13. SUPPLEMENTARY NOTES							
14. ABSTRACT Today's society has become more dependent on the integrity and protection of digital information used in daily transactions resulting in an ever increasing need for information security. Additionally, the need for faster and more secure cryptographic algorithms to provide this information security has become paramount. Hardware implementations of cryptographic algorithms provide the necessary increase in throughput, but at a cost of leaking critical information. Side Channel Analysis (SCA) attacks allow an attacker to exploit the regular and predictable power signatures leaked by cryptographic functions used in algorithms such as RSA. In this research the focus on a means to counteract this vulnerability by creating a Critically Low Observable Anti-Tamper Keeping Circuit (CLOAK) capable of continuously changing the way it functions in both power and timing. This research has determined that a polymorphic circuit design capable of varying circuit power consumption and timing can protect a cryptographic device from an Electromagnetic Analysis (EMA) attacks. In essence, we are effectively CLOAKing the circuit functions from an attacker.							
15. SUBJECT TERMS side channel analysis, polymorphic, DEMA, SEMA, RSA, obfuscation, timing analysis, FPGA, countermeasure							
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON		
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr. Yong C. Kim		
U	U	U	UU	126	19b. TELEPHONE NUMBER (include area code) (937) 255-3636, x4620; yong.kim@afit.edu		