Implementation Attacks on Block Ciphers: New Approaches and Countermeasures

Thesis submitted to the
Indian Institute of Technology Kharagpur
for award of the degree of

Bachelor of Technology

by

Ashrujit Ghoshal
(14CS10060)

under the guidance of

Dr. Debdeep Mukhopadhyay

Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

May 2018

To Maa and Baba

Statement of Originality

I certify that

1. The work contained in this report has been done by me under the guidance of my supervisor.
2. The work has not been submitted to any other Institute for any degree or diploma.
3. I have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.
4. Whenever I have used materials (data, theoretical analysis, figures, and text) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the references. Further, I have taken permission from the copyright owners of the sources, whenever necessary.

Ashrujit Ghoshal Date:
Department of Computer Science and Engineering
IIT Kharagpur

Certificate

This is to certify that the project report entitled "**Implementation Attacks on Block Ciphers: New Approaches and Countermeasures**" submitted by **Ashrujit Ghoshal** (Roll No. 14CS10060) to Department of Computer Science and Engineering, IIT Kharagpur towards partial fulfilment of requirements for the award of degree of Bachelor of Technology in Computer Science and Engineering is a record of bona fide work carried out by him under my supervision and guidance.

Dr. Debdeep Mukhopadhyay
Associate Professor
Department of Computer Science and Engineering
IIT Kharagpur

Date:

Acknowledgements

This work encapsulates my nascent steps as a researcher. These steps have been made possible through help, encouragement and guidance of several people. First of all, I would like to express my heartfelt gratitude to my advisor Prof. Debdeep Mukhopadhyay for his motivation, guidance and tremendous belief in me. His rich ideas form the basis of the work done in this thesis. His infectious enthusiasm for solving open-ended problems has inculcated in me the passion of pursuing research as a career. His advice on how to strike a balance of research work with undergraduate coursework has proven invaluable for me.

I would like to thank Prof. Mridul Nandi for giving me an opportunity to explore the field of cryptography during an internship at ISI Kolkata in 2016. That experience led me to choose cryptography as the topic for my thesis. I am hugely indebted to Prof. Vincent Rijmen for providing me the opportunity to work under him at Computer Security and Industrial Cryptography(COSIC) group at KU Leuven, Belgium. A major part of the thesis deals with Threshold Implementations, first introduced in a paper co-authored by Vincent. I learnt all about Threshold Implementations while working at COSIC and this thesis would not have been possible without that knowledge.

I am extremely grateful to Sikhar Da for his patience and diligence in answering my questions. The long discussions and debates with him showed me the way forward whenever I got stuck on a problem. I am also thankful to him for sharing his experiences as an undergraduate researcher of this same institute. I would like to thank Thomas De Cnudde for helping me out with the experiments at COSIC and taking time out of his busy schedule to help me write the draft of my first research paper. I also learnt a great deal in how to present technical papers from Thomas. I am grateful to my other co-authors: Rajat Da, Nilanjan Da, Dr. Vishal Saraswat, Dr. Santosh Ghosh and Dr. Stjepan Picek. I would express my gratitude to Dr. Tomer Ashur for making sure I felt part of the family at COSIC. I am extremely thankful to my friends and wingmates for their support during my stay at this institute. My friends Sayan and Meghna deserve a special mention for providing me with constant motivation and help even during lows.

Most importantly, none of this would have been possible without the love, support and sacrifices of my parents- Maa and Baba. Words are not enough to express my gratitude towards them for always believing in me and making sure that I had their unwavering support even when I faltered. I dedicate this thesis to them.

Last but not the least, I thank my institute IIT Kharagpur for providing me several opportunities and making my B.Tech life enjoyable and fulfilling.

Ashrujit Ghoshal

Abstract

In today's world implementation attacks like active fault attacks and side-channel power analysis attacks are potent attacks on standard cryptosystems. These attacks can be carried out in spite of mathematically provable security of cryptosystems if they are naively implemented. Devising new implementation attacks is important from a security point of view since countermeasures can be designed only after the attacks are known. This thesis introduces a new category of active fault attacks called Template based fault injection analysis attacks which uses pre-built templates to retrieve the secret key. These attacks involve a profiling phase followed by a matching phase. Countermeasures against very common and easy to carry out implementation attacks are necessary to be deployed in hardware implementations of encryption algorithms. First order differential power analysis is one such common side-channel analysis attack. This thesis focuses on threshold implementation (TI) design of block ciphers, an effective countermeasure against differential power attacks. We focus on reducing the randomness used in TI designs and present the first threshold implementation designs of the Boyar Peralta AES S-Box which uses significantly lesser randomness than existing TI designs of the AES S-Box. Designing TI involves an increase in area, hence lightweight TI designs are of special interest. This thesis presents the first TI design of the block cipher KHUDRA and devises a general strategy of designing 4×4 S-Boxes with optimal cryptographic properties which have low area and power footprints. We use cellular automata and use a time-area trade-off for these designs. Finally, we present a S-Box whose TI design has area smaller than TI designs of S-Boxes of existing lightweight block cipher implementations. All claims and propositions in the thesis have been substantiated by simulation studies and real life experiments.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Recent times have seen an exponential increase in financial and other transactions over the internet which necessarily need to be carried out in an encrypted manner. Devices responsible for carrying out the encryptions span a wide spectrum ranging from smartcards to high-end servers. Due to the varied range of devices on which these computationally intensive encryption algorithms are executed, there is ample scope for an adversary to exploit flaws in the implementation of these algorithms on these devices. Mathematically provable security of the underlying cryptosystem is not a sufficient condition to prevent an adversary with access to the encryption hardware from compromising security. For example, commonly used symmetric key algorithm like AES and public key algorithm like RSA are provably secure mathematically. However, if naively implemented, the purpose of these schemes are defeated.

The structure of inputs and outputs are mathematically analysed to find potential weaknesses in classical cryptanalytic attacks. Such a weakness can be exploited to recover plaintexts with a certain number of ciphertexts or in the best case reconstruct the entire secret key. Implementation attacks differ fundamentally from classical cryptanalytic attacks since they do not solely target the abstract cryptographic algorithm. With the increase in the number and variety of devices that perform cryptographic operations, implementation attacks have gained popularity. Side-channel analysis attacks and active fault attacks are two of the most common implementation based attacks. Side-channel analysis of ciphers attempts to exploit the correlation between physical measurements such as power dissipation obtained at various time instances, and the internal state of the processing device at those time instances, which is a function of the secret key. Active fault analysis, on the other hand, involves the injection of faults into cryptographic systems and analysis under different fault models to retrieve the secret key. In this thesis, we present a new category of active fault attacks. We also present new countermeasures against side-channel power analysis attacks.

This thesis also specifically designs countermeasures against side-channel power analysis attacks for lightweight block ciphers. Exploration of lightweight designs has become a topic of great interest since NIST (National Institute of Standards and Technology) has announced to create a portfolio of lightweight algorithms through an open process. The report emphasizes that with emerging applications like automotive systems, sensor networks, healthcare, distributed control systems, the Internet of Things (IoT), cyber-physical systems, and the smart grid, a detailed evaluation of the so-called light-weight ciphers helps to recommend algorithms in the context of profiles, which describe physical, performance, and security characteristics. This thesis provides countermeasures for some existing lightweight designs. We also provide design strategies for lightweight designs whose countermeasures have less resource requirements in terms of power and area.

1.1 Motivation and Objectives of the Work

The study of implementation attacks can be broadly divided into two aspects:

1. **Devising new strategies for implementation based attacks**: This involves exploiting existing side-channels or injecting faults in implementations in order to recover the secret key of a cipher with complexity better than the brute force attack.

2. **Designing countermeasures against known implementation attacks**: In order to thwart implementation attacks, countermeasures are designed under suitable assumptions. These countermeasures must provide some form of guarantee against implementation attacks like formal proofs of security.

Devising new implementation attacks is important from a security point of view since countermeasures can be designed only after the attacks are known. Ever since the seminal work of Boneh *et al.* demonstrated a successful a fault attack on the RSA cryptosystem in [BDL97], there has been a development of a wide array of new fault analysis attacks in the cryptographic community. Several types of fault attacks have been proposed in the recent literature like DFA which exploit the relation between faulty and fault-free ciphertext pairs requiring pairs of correct and faulty ciphertexts [PQ03, TMA11, Muk09], DFIA which exploits the bias in fault distribution and requires only faulty ciphertexts [FJLT13, GYTS14], Safe error attacks (SEA) [RM07] which do not require any faulty ciphertexts and exploit the behavior of the cipher under fault attacks, etc.

Template attacks (TA) have been a popular form of side-channel analysis in the cryptographic literature, the use of templates in the context of fault attacks has not been widely

explored. TA models precisely the noise pattern of the target device and extracts the maximum possible information from any available leakage sample. This makes TA a threat to implementations otherwise secure based on the assumption that an adversary has access to only a limited number of side-channel samples. In this thesis, we intend to explore whether the concept of templates can be extended to fault attacks. We aim to design a generalized Fault Injection Analysis (FIA) strategy that removes the dependency of existing techniques on specific fault models. Our approach would learn the behavior of the device-under-test (DUT) under an unrestricted set of fault injection parameters, irrespective of the fault nature instead of analyzing the behavior of the target implementation under a given set of faults. Such an attack strategy would allow a larger exploitable fault space.

Preventing easily exploitable implementation attacks is of prime importance since a wide array of devices ranging from smartcards to IoT devices run encryption algorithms. Differential Power Attacks (DPA) are extremely potent for hardware implementations as they require a few power traces to recover the secret key in naively implemented algorithms. First order DPA is very easy to carry out due to its low complexity and hence very common in practice. The efficacy of DPA makes mandatory to thwart first-order DPA in the hardware implementation of any cryptographic algorithm. The countermeasures against DPA try to remove the correlation between intermediate state value of the algorithms and the power traces. Several techniques have been proposed over the years as countermeasures against DPA. Leakage resilience [DP08] is one such countermeasure which limits the number of iterations of an algorithm using the same key. This approach drastically affects the performance of the system and hence is not practical for real-world implementations. A general approach focuses on decreasing the information gathered from traces. Some ad-hoc techniques like increasing noise, introducing dummy operations thus decreasing SNR are used to decrease information gathered from traces. These countermeasures become insecure with increasing attack time [DRS+12]. Some special constant power implementation like WDDL [TV04] can also be used to decrease information gathered from traces at huge overhead costs. One of the most efficient approaches to thwart DPA involves breaking the correlation between the power traces and the intermediate values of the computations. This method achieves security by randomizing intermediate values using secret sharing and carrying out computation on the shared values. This technique is referred to as masking [CJRR99, GP99].

Threshold Implementation (TI) is one such masking scheme that provides provable security even in the presence of glitches. It is one of the most used masing schemes due to its security guarantees and practical and minimal underlying assumptions. However, one disadvantage of TI is that it is resource hungry. Additional clock cycles, area, and randomness are required to design TI of cryptographic algorithms. Non-linear functions especially

tend to blow up in terms of area and randomness in TI designs. It is extremely important to reduce clock cycles, area and randomness as much as possible in TI designs of non-linear function. In this thesis, we aim to design the TI of AES S-Box. AES is one of the most commonly used block ciphers and the S-Box is the only non-linear function in AES. We target a particular implementation of the AES S-Box, the Boyar-Peralta S-Box [BP12] which has no previous TI designs. We aim to minimize the randomness and the number of clock cycles required while keeping the area of the S-Box small.

TI of lightweight S-Boxes is of increasing interest in the age of IoT and ubiquitous computing. The advent of the era of Internet-of-Things (IoT) has given rise to a number of smart devices with the ability to communicate with each other across heterogeneous network interfaces. The main constituents of any IoT framework are the numerous end nodes/devices, that are often constrained in terms of their memory capacity, processing speed, and power consumption rates. At the same time, these nodes commonly process sensitive data that needs to cryptographically protected against possible leakages to malicious adversaries. Traditional encryption mechanisms in the public and private-key settings are mostly resource-hungry, which makes them unsuitable for deployment in IoT devices. This has motivated the development of a large number of symmetric-key block ciphers that are lightweight, in the sense that they are area-efficient and/or low power consuming. When deployed in hardware the lightweight designs must be protected against first order DPA at the very least. Designing DPA resistant countermeasures of lightweight block ciphers is especially challenging since countermeasures like TI are resource hungry. KHUDRA [KM14] is one such lightweight block cipher. KHUDRA had no previous TI design. In this thesis we aim to design the first TI of the block cipher KHUDRA ensuring its resource requirements to the existing DPA resistant lightweight ciphers like PRESENT.

As mentioned earlier, the main disadvantage TI suffers from is that the non linear functions require large area and randomness. In this thesis we aim to introduce a design paradigm of lightweight 4×4 S-Boxes which are amenable to small area TI designs. Moreover these S-Boxes need to have optimal cryptographic properties. We specifically target the area and power requirements of resultant S-Boxes and aim to produce a TI of an optimal 4×4 S-Box with one of the smallest area and power requirements. We use cellular automata rules to design a trade-off between area and clock cycles. Use of cellular automata helps us to reuse parts of the circuit leading to low area and low power designs.

1.2 Major contributions of this Thesis

This section summarizes the major contributions of the thesis.

1. **Development of template-based fault injection analysis of block ciphers:** This thesis presents a generic algorithm comprising of a template building and a template matching phase, which are easily instantiable for any target block cipher. The templates are built on pairs of internal state segment and key segment values at different fault intensities. This attack allows exploitation of low-granularity faults such as multi-byte faults, that do not require high precision fault injection equipment. The attack does not require the exact knowledge of the underlying fault model. In order to substantiate the effectiveness of our methodology, a case-study targeting a hardware implementation of AES-128 on a Spartan-6 FPGA is presented in this thesis.

2. **Design of the first Threshold Implementation of KHUDRA:** This thesis presents the design of the first threshold implementation of the block cipher KHUDRA. KHUDRA is a lightweight block cipher and hence it is necessary that its threshold implementation had resource requirements comparable to other lightweight block ciphers. Our design requires lesser area than protected implementations of other well-known block ciphers like PRESENT, SIMON, SPECK etc.

3. **Design of several masked implementations of the Boyar Peralta AES S-Box:** In this thesis, we present the first threshold implementations of the Boyar-Peralta AES S-Box. We carry out a full design space exploration, investigating various trade-offs between area, randomness and the number of clock cycles. The set of secure implementations we present gives the hardware designer more options for tailoring their implementations according to their specifications. Our implementations compare favourably with the existing implementations of the AES S-Box. The smallest implementation we design reduces the randomness by 63% and the number of clock cycles by 50% compared to the smallest known masked Canright's AES -Box.

4. **First Threshold Implementation design of the AES S-Box with zero randomness:** We present a 4-shared TI design of the Boyar Peralta AES S-Box that requires no randomness at all. This is the first TI design of an AES S-Box with no randomness. The design uses a novel 4-to-4 sharing of the 2-input AND gate to remove randomness.

5. **Formulation of design strategy of Lightweight and Side-channel Secure S-Boxes using Cellular Automata:** In this thesis, we explore the possibility of designing cryptographically optimal 4×4 S-Boxes from CA rules, while also ensuring that such S-Boxes give rise to side-channel secure TI circuits with low area footprint and power consumption. We show how cellular automata can be used in order to

design nonlinear functions with inherently lightweight implementations. We iterate over a single instance of the CA rule, while cyclically shifting the input bits, to obtain one output bit of an S-Box at a time. We demonstrate that a significant proportion of the resulting S-Boxes achieve cryptographically optimal properties, and give rise to distinct classes based on their implementation overheads and amenability to TI implementations.

6. **Design of area and power efficient Threshold Implementation of an optimal 4×4 S-Boxes:** Our implementation results on ASIC (180nm technology) show that the most lightweight TI circuit among all CA-based S-boxes has a 49.42% smaller area-footprint and consumes 52.3% less power as compared to the best-known TI of the PRESENT S-Box. The same TI circuit also consumes 35.36% smaller area-footprint and consumes 44.46% less power as compared to a highly optimized TI of the GIFT S-Box.

1.3 Thesis Organization and Overview

This thesis provides a new technique for fault attacks and designs countermeasures against differential power attacks, the most common form of side-channel analysis attacks. Chapter 2 provides the methodology of the new category of fault attacks called Template based fault injection analysis attacks on block ciphers. We use the concepts of templates similar to template attacks using power traces. Our attacks do not require exact knowledge of the underlying fault model. In order to validate the feasibility of our attack, we present real-life experimental results. In chapter 3, we introduce the theory behind threshold implementations. We then present the first threshold implementation of the block cipher KHUDRA. In chapter 4, we design several threshold implementations of the Boyar Peralta AES S-Box. The chapter provides justification for the design choices we make to come up with the set of secure implementations. In chapter 5, we design cryptographically optimal 4×4 S-Boxes using cellular automata with the objective that the TI designs of these S-Boxes be power and area efficient. We provide a general design strategy and divide S-Boxes into distinct classes based on implementation overheads. We provide a detailed case study of S-Boxes from two of the classes with the smallest area and provide detailed comparision with implementations of existing threshold implementations of lightweight block ciphers. Finally, in chapter 6, we summarize the contributions of the thesis and suggest directions of future works.

Chapter 2

Template-based Fault Injection Analysis of Block Ciphers

While template attacks have been a popular form of side-channel analysis in the cryptographic literature, the use of templates in the context of fault attacks has not yet been explored to the best of our knowledge. In this chapter, we present the first template-based fault injection analysis of FPGA-based block cipher implementations. Our approach involves two phases. The first phase is a profiling phase where we build templates of the fault behavior of a cryptographic device for different secret key segments under different fault injection intensities. This is followed by a matching phase where we match the observed fault behavior of an identical but black-box device with the pre-built templates to retrieve the secret key. We present a generic treatment of our template-based fault attack approach for SPN block ciphers, and illustrate the same with case studies on a Xilinx Spartan-6 FPGA-based implementation of AES-128.

2.1 Introduction

The advent of implementation-level attacks has challenged the security of a number of mathematically robust cryptosystems, including symmetric-key cryptographic primitives such as block ciphers and stream ciphers, as well as public-key encryption schemes. Implementation attacks come in two major flavors - side-channel analysis (SCA) and fault injection analysis (FIA). SCA techniques typically monitor the leakage of a cryptographic implementation from various channels, such as timing/power/EM radiations, and attempt to infer the secret-key from these leakages [KJJ99, MOP08]. FIA techniques, on the other hand, actively perturb the correct execution of a cryptographic implementation via voltage/clock glitches [SGD08, BBBP13, ADN+10], EM pulses [DDRT12] or precise laser beams [CCF+08, CML+11]. With the growing number of physically accessible embedded devices processing sensitive data in today's world, implementation level attacks assume

significance. In particular, a thorough exploration of the best possible attacks on any cryptographic implementation is the need of the hour.

2.1.1 Fault Models for Fault Injection Analysis

Nearly all FIA techniques in the existing literature assume a given *fault model* (such as random faults [DDRT12] and/or stuck-at-faults [RM07]) in a given location of the cipher state. Some of these techniques, such as differential fault analysis (DFA) [PQ03, TMA11, Muk09] and differential fault intensity analysis (DFIA) [FJLT13, GYTS14] are found to be more efficient in the presence of highly localized faults, such as single bit flips, or faults restricted to a given byte of the cipher state. While DFA attacks are possible using multiple byte faults, e.g. diagonal faults [SMC09], the fault pattern impacts the complexity of key-recovery. In particular, with respect to AES-128, faults restricted to a single diagonal allow more efficient key-recovery as compared to faults spread across multiple diagonals. Similarly, DFIA typically exploits the bias of fault distribution at various fault intensities, under the assumption that the fault is restricted to a single byte/nibble of the cipher state [GYTS14]. Other techniques such as fault sensitivity analysis (FSA) [LSG$^+$10, MMG14] require the knowledge of the critical fault intensity at which the onset of faulty behavior is observed. This critical value is then correlated with the secret-key dependent cipher state value. Finally, FIA techniques such as safe-error analysis (SEA) [BS03] and differential behavioral analysis (DBA) [RM07] require highly restrictive fault models such as stuck-at faults, where a specific target bit of the cipher state is set to either 0 or 1. In recent literature, microcontroller-based implementation of cryptographic algorithms have been subjected to instruction-skip attacks [CT05, HMER], where the adversary uses precise injection techniques to transform the opcode for specific instructions into that for NOP (no-operation).

Similarity between FIA and SCA. The above discussion clearly reveals that existing FIA techniques are *inherently dependent* on the ability of an adversary to replicate a *specific fault model on an actual target device*. Fault precision and fault localization contribute to the efficiency of the attack, while the occurrence of random faults outside the target model generate *noisy ciphertexts*, thereby degrading the attack efficiency. Observe that this is conceptually similar to the effect of noise on the efficiency of traditional SCA techniques such as simple power analysis (SPA) and differential power analysis (DPA). In particular, the success rate for these techniques is directly proportional to the signal-to-noise ratio (SNR) of an implementation.

Our Motivation. In this work, we aim to devise a generalized FIA strategy that overcomes the dependency of existing techniques on specific fault models. Rather than analyzing the behavior of the target implementation under a given set of faults, our approach would *learn*

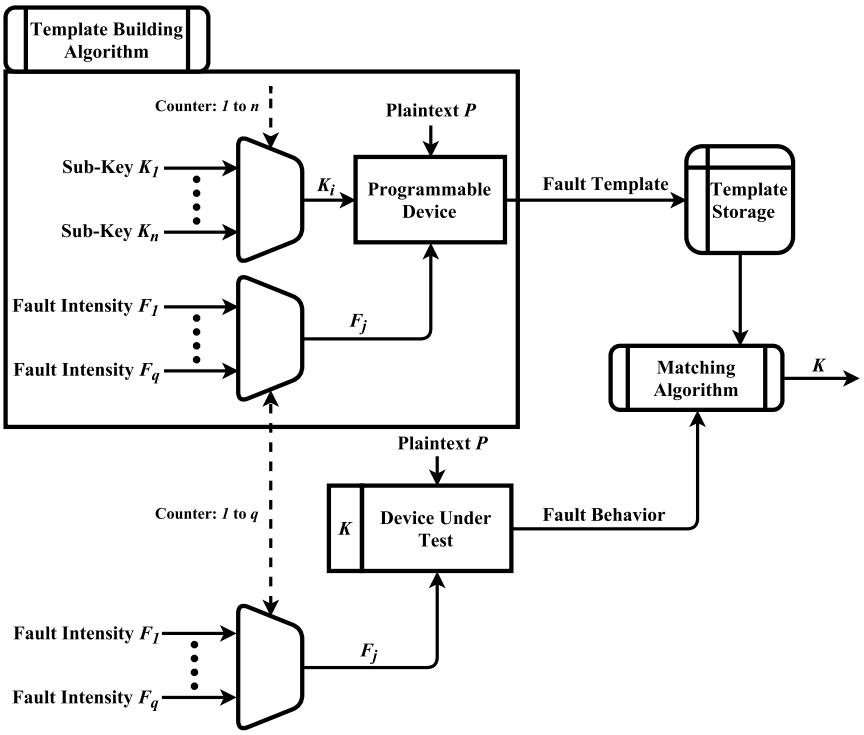


Figure 2.1: Template-based Fault Injection Analysis: An Overview

the behavior of the device-under-test (DUT) under an unrestricted set of fault injection parameters, irrespective of the fault nature. Such an attack strategy would allow a larger exploitable fault space, making it more powerful than all reported FIA techniques. As discussed next, an equivalent of the same approach in the context of SCA is well-studied in the literature.

2.1.2 Template Attacks: Maximizing the Power of SCA

Template attacks (TA) were proposed in [CRR02] as the strongest form of SCA in an information-theoretic setting. Unlike other popular SCA techniques such as DPA, TA does not view the noise inherent to any cryptographic implementation as a hindrance to the success rate of the attack. Rather, it models precisely the noise pattern of the target device, and extracts the maximum possible information from any available leakage sample. This makes TA a threat to implementations otherwise secure based on the assumption that an adversary has access to only a limited number of side-channel samples. On the flip side, TA assumes that the adversary has full programming capability on a cryptographic device identical to the target black-box device.

2.1.3 Our Contribution: Templates for Fault Injection Analysis

The existing literature on TA is limited principally to SCA, exploiting passive leakages from a target cryptographic device for key recovery. In this work, we aim to extend the scope of TA to active FIA attacks. Figure 2.1 summarizes our template-based FIA technique. Our approach is broadly divided into two main phases:

- The first phase of the attack is a *profiling phase*, where the adversary is assumed to have programming access to a device identical to the black-box target device. The adversary uses this phase to characterize the *fault behavior of the device* under varying fault injection intensities. We refer to such characterizations as the *fault template* for the device. We choose the *statistical distribution of faulty ciphertext values* under different fault injection intensities as the basis of our characterization. The templates are typically built on small-segments of the overall secret-key, which makes a divide-and-conquer key recovery strategy practically achievable. Note that the matching phase does not require the correct ciphertext value corresponding to a given encryption operation.
- The second phase of the attack is the *matching phase*, where the adversary obtains the fault behavior of the target black-box device (with an embedded non-programmable secret-key K) under a set of fault injection intensities, and matches them with the templates obtained in the profiling phase to try and recover K. The idea is to use a maximum likelihood estimator-like distinguisher to identify the key hypothesis for which the template exhibits the maximum similarity with the experimentally obtained fault behavior of the target device.

2.1.4 Comparison with Existing FIA Techniques

In this section, we briefly recall existing FIA techniques, and explain their differences with our proposed template-based FIA approach. As already mentioned, our technique has two phases, and assumes that the adversary has programmable access to a device identical to the device under test. At the same time, it allows modeling the behavior of the device independent of specific fault models, as is done in most state-of-the-art FIA techniques. We explicitly enumerate these differences below.

Differential Fault Analysis (DFA): In DFA [DLV03, PQ03, Kim10, TMA11], the adversary injects a fault under a specific fault model in target location of the cipher state, and analyzes the fault propagation characteristics using the knowledge of the fault-free and faulty ciphertexts. Our template-based FIA does not trace the propagation of the fault;

rather it simply creates a template of the faulty ciphertext distribution under different fault injection intensities. This makes our approach independent of any specific fault model.

Differential Fault Intensity Analysis (DFIA): DFIA [GYTS14, PCNM15] exploits the underlying bias of any practically achieved fault distribution on the target device, once again under a chosen fault model. It is similar in principle to DPA in the sense that it chooses the most likely secret-key value based upon a statistical analysis of the faulty intermediate state of the block cipher, derived from the faulty ciphertext values only. Our template-based FIA can be viewed as a generalization of DFIA with less stringent fault model requirements. Similar to DFIA, our approach also does not require the correct ciphertext values. However, our approach does not statistically analyze the faulty intermediate state based upon several key hypotheses. Rather, it pre-constructs separate templates of the *faulty ciphertext distribution* for each possible key value, and matches them with the experimentally obtained faulty ciphertext distribution from the black-box target device. Rather than focusing on specific fault models, the templates are built for varying fault injection intensities.

Fault Sensitivity Analysis (FSA): FSA [LSG+10, MMG14] exploits the knowledge of the critical fault intensity under which a device under test starts exhibiting faulty output behavior. The critical intensity is typically data-dependent, which allows secret-key recovery. FSA does not use the values of either the correct or the faulty ciphertexts. However, it requires a precise modeling of the onset of faults on the target device. Our methodology, on the other hand, uses the faulty ciphertext values, and is free of such precise critical fault intensity modeling requirements.

Safe Error Analysis (SEA): In SEA [BS03, RM07], the adversary injects a fault into a precise location of the cipher state, and observes the corresponding effect on the cipher behavior. A popular fault model used in such attacks is the stuck-at fault model. The adversary injects a fault to set/reset a bit of the cipher state, and infers from the nature of the output if the corresponding bit was flipped as a result of the fault injection. Quite clearly, this fault model is highly restrictive. Our approach, on the other hand, allows random fault injections under varying fault intensities, which makes easier to reproduce in practice on real-world target devices.

2.2 Template-Based FIA: Detailed Approach

In this section, we present the details of our proposed template-based FIA. Given a target device containing a block cipher implementation, let $\mathcal{F}$ be the space of all possible fault

intensities under which an adversary can inject a fault on this device. Now, assume that a random fault is injected in a given-segment S_k of the cipher state under a fault intensity $F_j \in \mathcal{F}$. Also assume that this state segment has value $P_{i'} \in \mathcal{P}$, and subsequently combines with a key segment $K_i \in \mathcal{K}$, where $\mathcal{P}$ and $\mathcal{K}$ are the space of all possible intermediate state values and key segment values respectively, resulting in a faulty ciphertext segment $C_{i,i',j,k}$. The granularity of fault intensity values depends on the injection equipment used - precise injection techniques such as laser pulses are expected to offer higher granularity levels than simpler injection techniques such as clock/voltage glitches. Note that we do not restrict the nature of the faults resulting from such injections to any specific model, such as single bit/single byte/stuck-at faults. With these assumptions in place, we now describe the two phases - the template building phase and the template matching phase - of our approach.

2.2.1 Template Building Phase

In this phase, the adversary has programmable access to a device identical to the device under test. By programmable access, we mean the following:

- The adversary can feed a plaintext P and master secret-key K of his choice to the device.
- Upon fault injection under a fault intensity $F_j \in \mathcal{F}$, the adversary can detect the target location S_k in the cipher state where the fault is induced
- The adversary has the knowledge of the corresponding key segment $K_i \in \mathcal{K}$ and the intermediate state segment $P_{i'} \in \mathcal{P}$. The key segment combines with the faulty state segment to produce the faulty ciphertext segment $C_{i,i',j,k}$.

Let $C^1_{i,i',j,k}, \cdots, C^N_{i,i',j,k}$ be the faulty ciphertext outputs upon N independent fault injections in the target location S_k under fault injection intensity F_j, corresponding to the intermediate state segment $P_{i'}$ and key segment K_i. We refer to the tuple $T_{i,i',j,k} = \left(C^1_{i,i',j,k}, \cdots, C^N_{i,i',j,k}\right)$ as a *fault template instance*. This template instance is prepared and stored for possible tuples $(K_i, P_{i'}, F_j, S_k) \in \mathcal{K} \times \mathcal{P} \times \mathcal{F} \times \mathcal{S}$, where $\mathcal{S}$ is the set of all fault locations in the cipher state that need to be covered for full key-recovery. The set of all such template instances constitutes the *fault template* for the target device. Algorithm 1 summarizes the main steps of the template building phase as described above.

Note: The number of fault injections N required per fault intensity during the template building phase may be determined empirically, based upon the desired success rate of key recovery in the subsequent template matching phase. Quite evidently, increasing N improves the success rate of key recovery.

Algorithm 1 Template Building Phase

Require: Programmable target device
Require: Target block cipher description
Ensure: Fault template T for the target device

Fix the set $\mathcal{S}$ of fault locations to be covered for successful key recovery depending on the block cipher description
Fix the space $\mathcal{F}$ of fault injection intensities depending on the device characteristics
Fix the number of fault injections N for each fault intensity
$T \leftarrow \phi$
for each fault location $S_k \in \mathcal{S}$ **do**
    for each corresponding intermediate state segment and key segment $(P_{i'}, K_i) \in \mathcal{P} \times \mathcal{K}$ **do**
        for each fault injection intensity $F_j \in \mathcal{F}$ **do**
            for each $l \in [1, N]$ **do**
                Run an encryption of $P_{i'}$ such that the target key segment has value K_i
                Inject a fault under intensity F_j in the target location S_k
                Let $C^l_{i,i',j,k}$ be the faulty ciphertext segment
            end for
            $T_{i,i',j,k} \leftarrow \left(C^1_{i,i',j,k}, \cdots, C^N_{i,i',j,k}\right)$
            $T \leftarrow T \cup T_{i,i',j,k}$
        end for
    end for
end for
return T

2.2.2 Template Matching Phase

In this phase, the adversary has black-box access to the target device. Under the purview of black-box access, we assume the following:

- The adversary can feed a plaintext P of his choice to the device and run the encryption algorithm multiple times on this plaintext.
- Upon fault injection under a fault intensity $F_j \in \mathcal{F}$, the adversary can induce the target location S_k in the cipher state where the fault is induced, by observing the corresponding faulty ciphertext $C'_{j,k}$.
- The adversary has no idea about the intermediate state segment $P_{i'}$ where the fault is injected, or the key segment K_i that subsequently combines with the faulty state segment to produce the ciphertext.

The adversary again performs N independent fault injections under each fault injection intensity F_j in a target location S_k, and obtains the corresponding faulty ciphertexts ${C'}^1_{j,k}, \cdots, {C'}^N_{j,k}$. All fault injections are performed during encryption operations using the same plaintext P as in the template building phase. These faulty ciphertexts are then given as input to a distinguisher $\mathcal{D}$. The distinguisher ranks the key-hypotheses $K_1, \cdots, K_n \in \mathcal{K}$, where the rank of K_i is estimated based upon the closeness of the experimentally obtained ciphertext distribution with the template instance $T_{i,i',j,k}$, for all possible intermediate state segments $P_{i'}$. The closeness is estimated using a statistical measure $\mathcal{M}$. The distinguisher finally outputs the key hypothesis K_i that is ranked consistently highly across all rank-lists corresponding to different fault injection intensities. Algorithm 2 summarizes our proposed template matching phase.

2.2.3 The Statistical measure M

An important aspect of the template matching phase is choosing the statistical measure M to measure the closeness of the experimentally observed faulty ciphertext segment distribution, with that corresponding to each template instance. We propose using a correlation-based matching approach for this purpose. The first step in this approach is to build a frequency-distribution table of each possible ciphertext segment value in each of the two distributions. Let the possible ciphertext segment values be in the range $[0, 2^{x-1}]$ (for example, $[0, 255]$ for a byte, or $[0, 15]$ in case of a nibble). Also, let $f(y)$ and $f'(y)$ denote the frequency with which a given ciphertext segment value $y \in [0, 2^{x-1}]$ occurs in the template and the experimentally obtained distribution, respectively. Since there are exactly N sample points in each distribution, we have $\sum_{y \in [0,2^{x-1}]} f(y) = \sum_{y \in [0,2^{x-1}]} f'(y) = N$.

Algorithm 2 Template Matching Phase

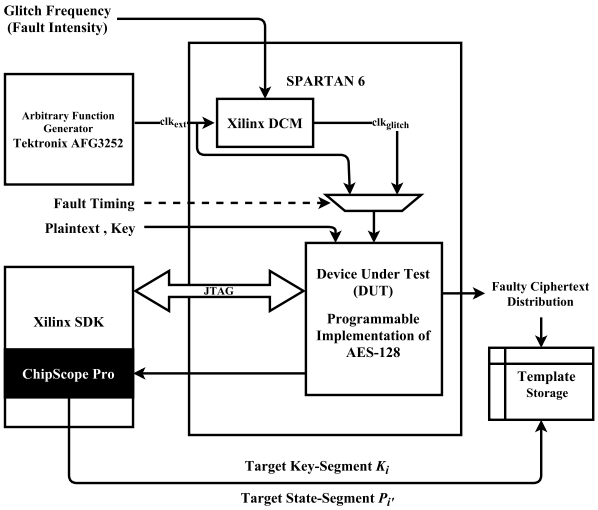
Require: Fault template T corresponding to plaintext P
Ensure: The secret-key
1: **for each** fault location $S_k \in \mathcal{S}$ **do**
2: **for each** fault injection intensity $F_j \in \mathcal{F}$ **do**
3: **for each** $l \in [1, N]$ **do**
4: Inject a fault under intensity F_j in location S_k
5: Let ${C'}^{l}_{j,k}$ be the faulty ciphertext segment
6: **end for**
7: $E_{j,k} \leftarrow \left({C'}^{1}_{j,k}, \cdots, {C'}^{N}_{j,k}\right)$
8: **end for**
9: **end for**
10: **for each** fault location $S_k \in \mathcal{S}$ **do**
11: **for each** fault injection intensity $F_j \in \mathcal{F}$ **do**
12: **for each** possible key hypothesis $K_i \in \mathcal{K}$ and intermediate state segment $P_{i'} \in \mathcal{P}$ **do**
13: $\rho_{i,i',j,k} \leftarrow \mathcal{M}\left(E_{j,k}, T_{i,i',j,k}\right)$
14: **end for**
15: **end for**
16: Store the pair $(K_i, P_{i'})$ pair such that $\sum_{F_j \in \mathcal{F}} \rho_{i,i'j,k}$ is maximum for the given fault location S_k.
17: **end for**
18: return the stored key hypothesis corresponding to each unique key segment location.

The next step is to compute the Pearson's correlation coefficient between the two distributions as:

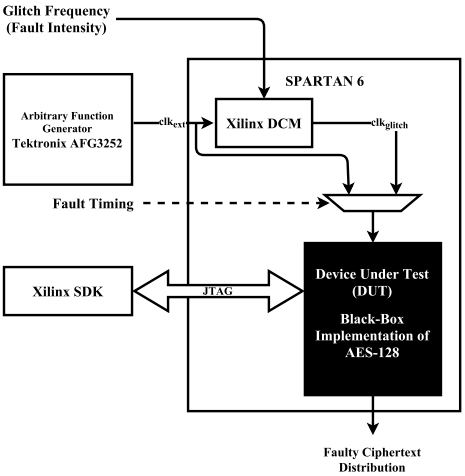
$$\rho = \frac{\sum\limits_{y \in [0, 2^x - 1]} \left(f(y) - \frac{N}{2^x}\right) \cdot \left(f'(y) - \frac{N}{2^x}\right)}{\sqrt{\sum\limits_{y \in [0, 2^x - 1]} \left(f(y) - \frac{N}{2^x}\right)^2} \sqrt{\sum\limits_{y \in [0, 2^x - 1]} \left(f'(y) - \frac{N}{2^x}\right)^2}}$$

The Pearson's correlation coefficient is used as the measure M. The choice of statistic is based on the rationale that, for the correct key segment hypothesis, the template would have a similar frequency distribution of ciphertext segment values as the experimentally obtained set of faulty ciphertexts, while for a wrong key segment hypothesis, the distribution of ciphertext segment values in the template and the experimentally obtained ciphertexts would be uncorrelated.

An advantage of the aforementioned statistical approach is that it can be extended to relaxed fault models such as multi-byte faults, that are typically not exploited in traditional FIA techniques. In general, if a given fault injection affects multiple locations in the block cipher state, the correlation analysis is simply repeated separately for each fault location. This is similar to the divide-and-conquer approach used in SCA-based key-recovery techniques.



(a) Template Building Phase



(b) Template Matching Phase

Figure 2.2: Experimental Set-Up

2.3 Case Study: Template-Based FIA on AES-128

In this section, we present a concrete case study of the proposed template-based FIA strategy on AES-128. As is well-known, AES has a plaintext and key size of 128 bits each, and a total of 10 rounds. Each round except the last one comprises of a non-linear S-Box layer (16 S-Boxes in parallel), a linear byte-wise ShiftRow operation, and a linear MixColumn operation, followed by XOR-ing with the round key. The last round does not have a MixColumn operation. This in turn implies that if a fault were injected in one or more bytes of the cipher state after the 9^{th} round MixColumn operation, the faulty state byte (or bytes) combines with only a specific byte (or bytes) of the 10^{th} round key. For example,

Table 2.1: Glitch Frequencies for Different Fault Models

Glitch Frequency (MHz)	Faulty Bytes	Bit Flips per Byte
125.3-125.5	1	1
125.6-125.7	1	2
125.8-126.0	1	3
126.1-126.2	2-3	1-3
> 126.2	> 3	> 5

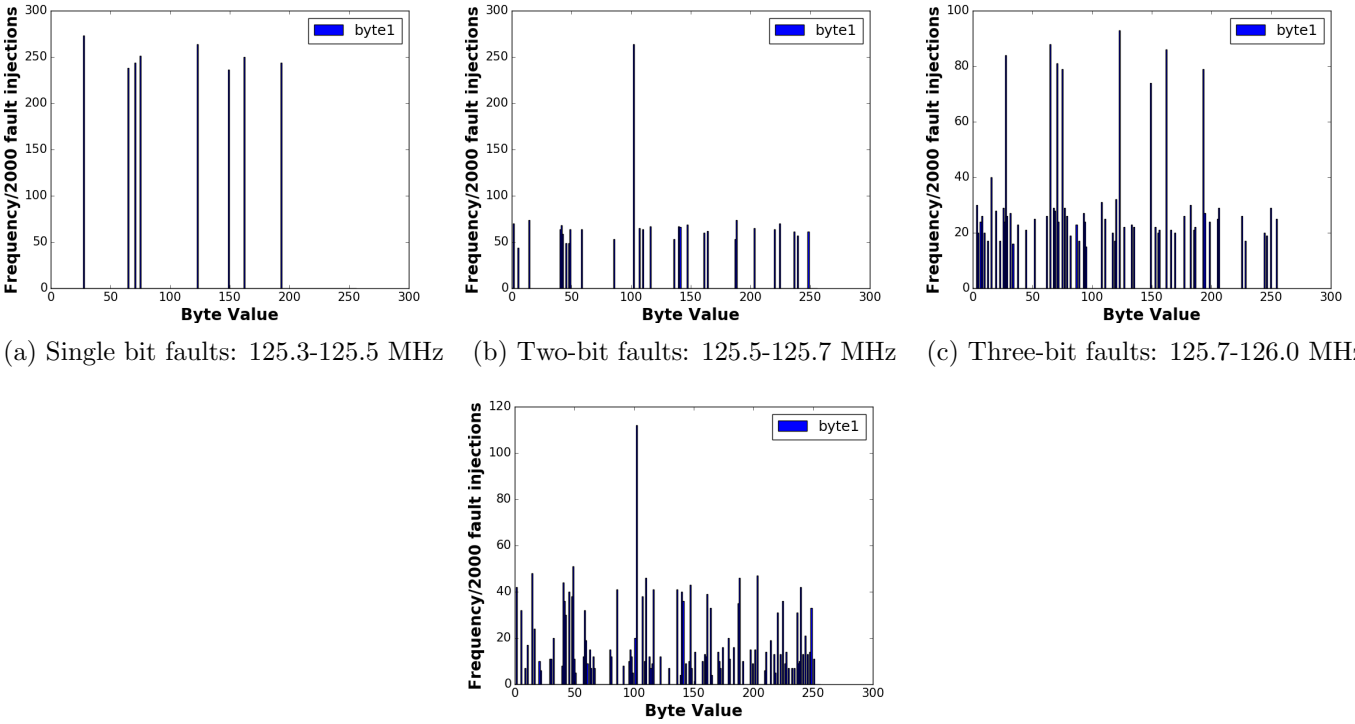
if a fault were injected in the first byte of the cipher state, the faulty byte would pass through the S-Box and ShiftRow operation, and combine with the first byte of the 10^{th} round key to produce the first byte of the faulty ciphertext. The exact relation between the fault injection location and the corresponding key segment depends solely on the ShiftRow operation, and is hence deterministic. This matches precisely the assumptions made in our attack description in the previous section. Consequently, this case study assumes that all faults are injected in the cipher state between the 9^{th} round MixColumn operation and the 10^{th} round S-Box operations. The aim of the fault attack is to recover byte-wise the whole 10^{th} round key of AES-128, which in turn deterministically reveals the entire secret-key.

2.3.1 The Fault Injection Setup

The fault injection setup (described in Figure 2.2) uses a Spartan 6 FPGA mounted on a Sakura-G evaluation board, a PC and an external arbitrary function generator (Tektronix AFG3252). The FPGA has a Device Under Test (DUT) block, which is an implementation of the block cipher AES-128. Faults are injected using clock glitches. The device operates normally under the external clock signal clk_{ext}. The glitch signal, referred to as clk_{fast}, is derived from the clk_{ext} via a Xilinx Digital Clock Manager (DCM) module. The fault injection intensity in our experiments is essentially the glitch frequency, and is varied using a combination of the DCM configuration, and the external function generator settings. In the template building phase, the intermediate cipher state $P_{i'}$ and the intermediate round key K_i are monitored using a ChipScope Pro analyzer, while in the template matching phase, the DUT is a black box with no input handles or internal monitoring capabilities. Table 2.1 summarizes the glitch frequency ranges at which these fault models were observed on the target device.

2.3.2 Templates for Single Byte Faults

In this section, we present examples of fault templates obtained from the device under test, for glitch frequencies that result in single byte fault injections in the AES-128 module. Since only a single byte is affected between the 9^{th} round MixColumn operation and the 10^{th} round



(a) Single bit faults: 125.3-125.5 MHz (b) Two-bit faults: 125.5-125.7 MHz (c) Three-bit faults: 125.7-126.0 MHz

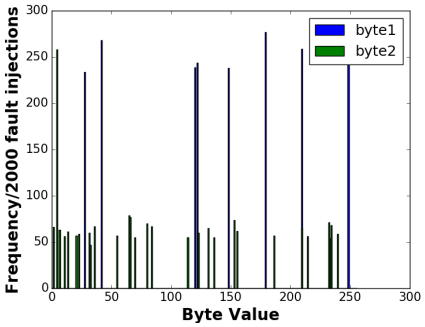
(d) 4-bit faults

Figure 2.3: Templates for Single Byte Faults: Distribution of Faulty Ciphertext Byte for Different Fault Injection Intensities

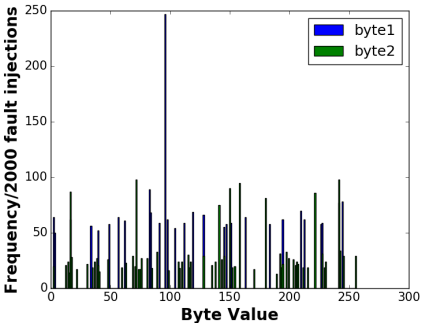
S-Box operations, we are interested in the distribution of the corresponding faulty byte in the ciphertext. Figure 2.3 presents fault templates containing ciphertext byte distributions for three categories of faults - single bit faults, two-bit faults, and three-bit faults. The templates correspond to the same pair of intermediate state byte and last round key byte for an AES-128 encryption. Quite evidently, the ciphertext distribution for each template reflects the granularity of the corresponding fault model. In particular, for a single bit fault, most of the faulty ciphertext bytes assume one of 8 possible values, while for three-bit faults, the ciphertext bytes assume more than 50 different values across all fault injections. In all cases, however, the distribution of ciphertext values is non-uniform, which provides good scope for characterizing the fault behavior of the device in the template building phase.

2.3.3 Templates for Multi-Byte Faults

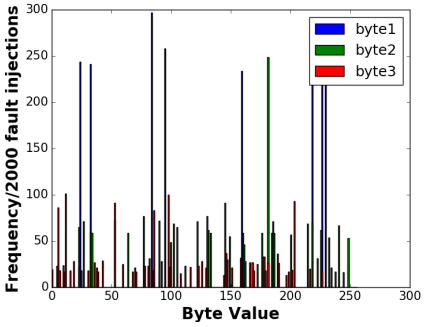
In this section, we present examples of fault templates constructed for glitch frequencies that result in multi-byte fault injections. Figure 2.4 shows the distribution of different bytes injected with different faults. It is interesting to observe that at the onset of multi-byte faults, the distribution of faulty ciphertext bytes is *not uniformly random*; indeed, it is



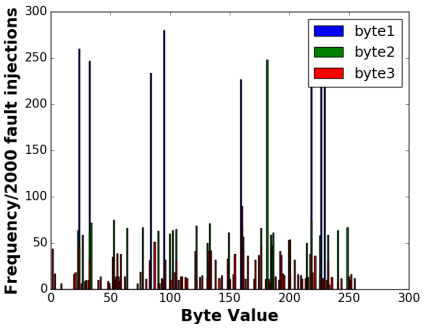
(a) 1-bit, 2-bit fault in 2 bytes: 126.1 MHz



(b) 2-bit, 3-bit fault in 2 bytes

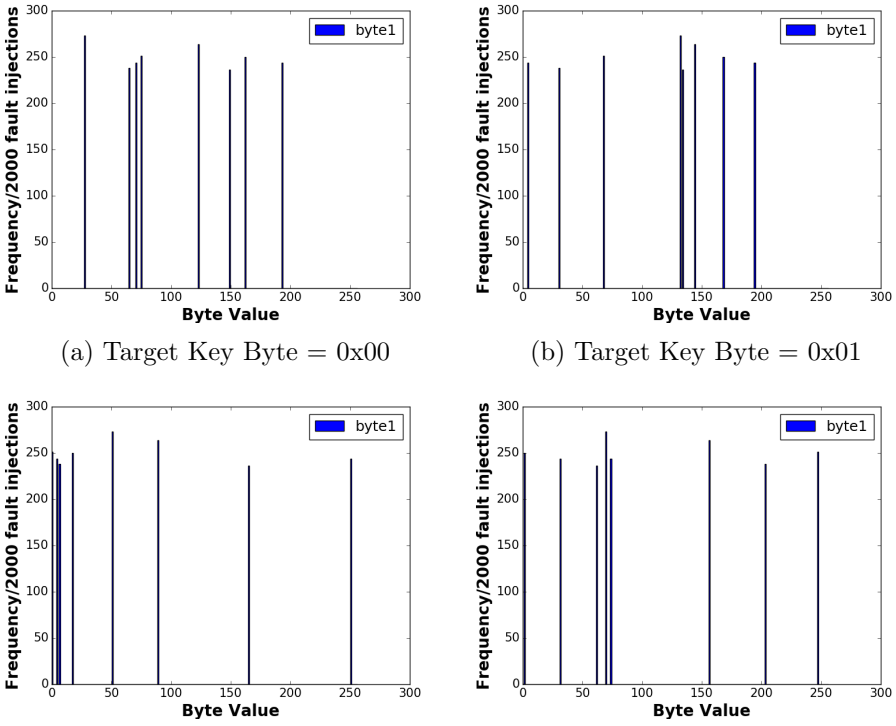


(c) 1-bit, 2-bit, 3-bit fault across 3 bytes: 126.2 MHz



(d) 1-bit, 2-bit, 4-bit fault in 3 bytes

Figure 2.4: Templates for Multi-Byte Faults: Distribution of Multiple Faulty Ciphertext Byte Values



(a) Target Key Byte = 0x00

(b) Target Key Byte = 0x01

(c) Target Key Byte = 0x02

(d) Target Key Byte = 0x03

Figure 2.5: Frequency Distributions for Faulty Ciphertext Byte : Same Intermediate State Byte but Different Key Byte Values

possible to characterize the fault behavior of the device in terms of templates under such fault models. Given the absence of MixColumn operation in the last round of AES, each faulty intermediate state byte combines independently with a random last round key byte. This allows a divide-and-conquer template matching approach, where the statistical analysis may be applied to each faulty ciphertext byte independently. This is a particularly useful mode of attack, since it can be launched even without precise fault injection techniques that allow targeting a single byte of the cipher state.

2.3.4 Variation with Key Byte Values

The success of our template matching procedure with respect to AES-128 relies on the hypothesis that for different key byte values, the ciphertext distribution corresponding to the same fault location is different. Otherwise, the key recovery would be ambiguous. We validated this hypothesis by examining the ciphertext distribution upon injecting a single bit fault in the first byte of the cipher state, corresponding to different key byte values. We illustrate this with a small example in Figure 2.5. Figures 2.5a, 2.5b, 2.5c and 2.5d represent the frequency distributions for faulty ciphertext byte corresponding to the same intermediate

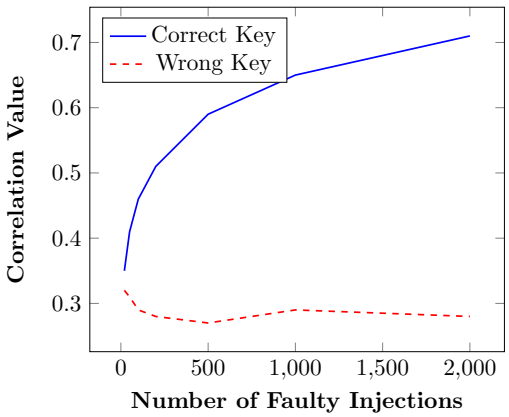
byte value of `0x00`, and key byte values `0x00`, `0x01`, `0x02` and `0x03`, respectively. Quite evidently, the three frequency distributions are unique and mutually non-overlapping. The same trend is observed across all 256 possible key byte values; exhaustive results for the same could not be provided due to space constraints.

2.3.5 Template matching for Key-Recovery

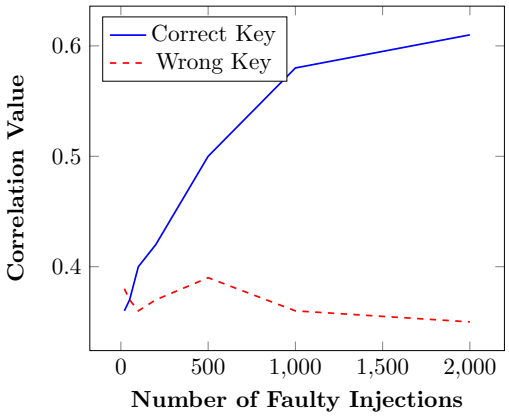
In this section, we present results for recovering a single key-byte for AES-128 under various fault granularities. As demonstrated in Figure 2.6, the correlation for the correct key hypothesis exceeds the average correlation over all wrong key hypotheses, across the three fault models - single bit faults, two-bit faults and three-bit faults. As is expected, precise single-bits faults within a given byte enable distinguishing the correct key hypothesis using very few number of fault injections (50-100); for less granular faults such as three-bit faults, more number of fault injections (200-500) are necessary. Finally, the same results also hold for multi-byte fault models, where each affected byte encounters a certain number of bit-flips. Since the key-recovery is performed byte-wise, the adversary can use the same fault instances to recover multiple key bytes in parallel.

2.4 Conclusion

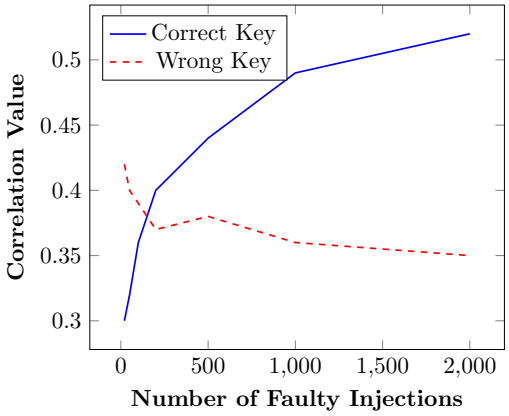
We presented the first template based fault injection analysis of block ciphers. We presented a generic algorithm comprising of a template building and a template matching phase, that can be easily instantiated for any target block cipher. The templates are built on pairs of internal state segment and key segment values at different fault intensities, while the number of fault instances per template depends on the statistical methodology used in the matching phase. In this work, we advocated the use of the Pearson correlation coefficient in the matching phase; exploring alternative techniques in this regard is an interesting future work. In order to substantiate the effectiveness of our methodology, we presented a case-study targeting a hardware implementation of AES-128 on a Spartan-6 FPGA. Interestingly, our attack allowed exploiting even low-granularity faults such as multi-byte faults, that do not require high precision fault injection equipment. It may be emphasized that the attack is devoid of the exact knowledge of the underlying fault model. Such fault models also allowed parallel recovery of multiple key-bytes, thus providing a trade-off between the number of fault injections, and the number of recovered key-bytes. An interesting extension of this work would be apply template-based analysis against implementations with fault attack countermeasures such as spatial/temporal/information redundancy.



(a) Single Bit Faults



(b) Two-Bit Faults



(c) Three-Bit Faults

Figure 2.6: Correlation between template and Observed Ciphertext Distribution: Correct Key Hypothesis v/s Wrong Key Hypothesis

Chapter 3

Threshold Implementation of KHUDRA

The block cipher KHUDRA [KM14] was designed keeping in mind the requirement of designing block ciphers which are also lightweight on FPGAs that are often preferred where reconfigurability and low development cost is a mandate. It has been shown that some algorithm choices like PRESENT are more apt for ASIC libraries with specialized library cells, while on FPGAs their compactness diminishes [KM14].

3.1 Preliminaries

3.1.1 Description of the KHUDRA Block Cipher

The encryption algorithm of KHUDRA operates on 64-bits plaintext block, along with 80-bits key length to produce 64-bits ciphertext. The structure of KHUDRA belongs to the category of Feistel ciphers (generalized type-2 [HR10]) and consist of 18 rounds. The Feistel structure of KHUDRA has two parts: a permutation based on Feistel and F-function, where the F-function, in turn, contains substitution-permutation-substitution layer. The block cipher use S-Box layer of PRESENT as it has "High Algebraic Degree" and "Low Differential and Linear Probability" [BKL$^+$07a]. The number of rounds inside the F-function is 6. The key scheduling algorithm and encryption algorithm are shown in Algorithm 4 and Algorithm 3 respectively. The block diagram of the encryption process is given in Figure 3.1.

3.1.2 Threshold Implementations

The threshold implementations (TI) masking technique was proposed by Nikova et al. [NRR06] as a countermeasure against Differential Power Analysis (DPA) attacks. It is secure even in non-ideal circuits where glitches have shown to result in leakage in more conventional masking schemes [MPO05]. The original proposal, which only dealt with first-order DPA

Algorithm 3 KHUDRA Encryption Algorithm

Require: $Plaintext[64], Key[80]$
Ensure: $Ciphertext[64]$
$R \leftarrow 0$
while $R \neq 17$ **do**
$branch_3[0:15] \leftarrow Plaintext[48-63]$
$branch_1[0:15] \leftarrow Plaintext[16-31]$
$internalRound \leftarrow 0$
while $internalRound \neq 5$ **do**
$internalbranch_3[0:3] \leftarrow Plaintext[60-63]$
$internalbranch_1[0:3] \leftarrow Plaintext[52-55]$
$Plaintext[60-63] \leftarrow SBoxlayer(Plaintext[60-63]) \oplus Plaintext[56-59]$
$Plaintext[52-55] \leftarrow SBoxlayer(Plaintext[52-55]) \oplus Plaintext[48-51]$
$Plaintext[56-59] \leftarrow internalbranch_1[0:3]$
$Plaintext[48-51] \leftarrow internalbranch_3[0:3]$
$internalbranch_3[0:3] \leftarrow Plaintext[28-31]$
$internalbranch_1[0:3] \leftarrow Plaintext[20-23]$
$Plaintext[28-31] \leftarrow SBoxlayer(Plaintext[28-31]) \oplus Plaintext[27-24]$
$Plaintext[20-23] \leftarrow SBoxlayer(Plaintext[20-23]) \oplus Plaintext[16-19]$
$Plaintext[24-27] \leftarrow internalbranch_1[0:3]$
$Plaintext[16-19] \leftarrow internalbranch_3[0:3]$
$internalRound \leftarrow internalRound + 1$
end while
$Plaintext[48-63] \leftarrow Plaintext[48-63] \oplus Plaintext[32-47] \oplus RoundKey[2 \times R + 1][0:15]$
$Plaintext[16-31] \leftarrow Plaintext[16-31] \oplus Plaintext[0-15] \oplus RoundKey[2 \times R][0:15]$
$Plaintext[32-47] \leftarrow branch_1[0:15]$
$Plaintext[0-15] \leftarrow branch_3[0:15]$
$R \leftarrow R + 1$
end while

Algorithm 4 KHUDRAgenerateKey

Require: $Key[80]$
Ensure: $roundKey[16]$
$whiteningKey_0[0:15] \leftarrow k_0 \leftarrow Key[0:15]$
$whiteningKey_1[0:15] \leftarrow k_1 \leftarrow Key[16:31]$
$whiteningKey_3[0:15] \leftarrow k_3 \leftarrow Key[49:63]$
$whiteningKey_4[0:15] \leftarrow k_4 \leftarrow Key[64:79]$
$k_2 \leftarrow Key[32:48]$
$i \leftarrow 0$
while $i \neq 35$ **do**
$Constant \leftarrow 0||i[0:5]||00||i[0:5]||0$
$roundKey_i \leftarrow k_{imod5} \oplus Constant$
$i \leftarrow i + 1$
end while

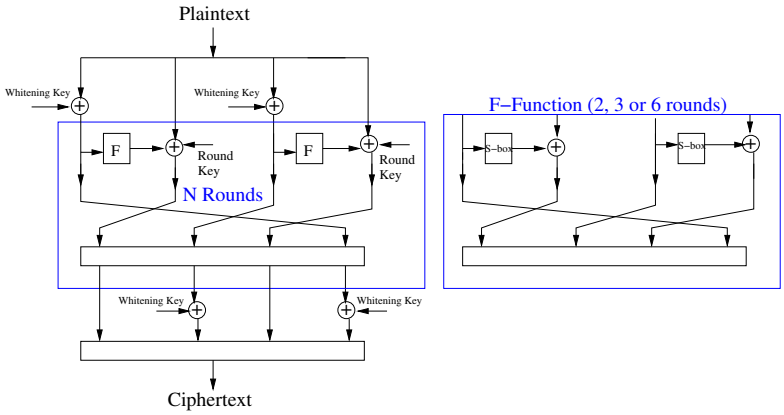


Figure 3.1: KHUDRA Block cipher operation

security, was later extended to protect against higher-order DPA attacks as well [BGN+14a, RBN+15]. The security of masking schemes is inherently tied to an adversary model. An attacker who observes the d^{th}-order statistical moment of e.g. a power trace or combines observations from d points in time nonlinearly in that power trace is said to be an attacker mounting a d^{th}-order attack. To prevent a d^{th}-order attack, a masking scheme of order $(d+1)$ is required. Fortunately, the number of readings needed for a higher-order attack to become successful grows exponentially with the noise standard deviation and therefore it is reasonable to guarantee practical security up to a certain order.

Implementing masking in hardware in a secure manner is not trivial. It is a delicate job since all the assumptions made on the leakage behavior of the underlying platform do not always hold in practice. For example, glitches are a known predominant threat [MPO05] to the security of masked implementations in CMOS technologies. Some masking schemes like Threshold Implementations (TI) work under assumptions which are more achievable in practical scenarios. In addition to these relaxed assumptions on the underlying leakage, TI offers provable security and allows to construct secure circuits which are realistic in size, all without requiring much intervention from a designer or many design iterations. For this reason, TI has been applied to many well-known cryptographic algorithms like KECCAK, AES, and PRESENT [BDN+13, DCBR+15, MPL+11, PMK+11].

TI is based on multi-party computation and secret sharing, and must satisfy the following properties in order to achieve the mentioned security:

1. **Uniformity.** Uniformity requires all intermediate shares to be uniformly distributed. It ensures state-independence from the mean of the leakages, which is a requirement to thwart first-order DPA. As mentioned in [Bil15] it suffices to check uniformity at the inputs and the outputs of each of the functions. Uniformity can be either achieved

through correction terms by using more input shares, or by adding randomness after the non-uniform computation.

2. **Non-completeness.** To achieve d^{th}-order non-completeness, any combination of d or less component functions f_i of $\mathbf{f}$ must be independent of at least one input share x_i. For protection against first-order DPA, 1^{st}-order non-completeness is required, i.e. every function must be independent of at least one input share. Non-completeness ensures that the side-channel security of the final circuit is not affected by glitches.

3. **Correctness.** This property simply states that applying the sub-functions to a valid shared input must always yield a valid sharing of the correct output.

In addition to TI's algorithmic properties, the physical leakage of each share or sub-function should be independent of all other shares or sub-functions, i.e. no coupling is present between the shares or sub-functions. Violating this assumption has shown to induce leakage in masked implementations [DCBG+17].

3.2 3-shared Threshold Implementation of Khudra

we designed a 3 shared Threshold Implementation of KHUDRA, which was the first such implementation of KHUDRA. The architecture of a 4-bit serialized implementation of KHUDRA is shown in Figure 3.3. The only non-linearity in the circuit for KHUDRA is in the S-Box which is present in the F-function. All other operations are linear. Designing threshold implementations of linear circuits is a fairly straightforward task. The operations performed on the shared and unshared input gives same output due to linearity. Hence for linear operations, we just need to replicate parts of the circuit. The challenge lies in designing threshold implementation of non-linear circuits. The S-box (non-linear) function $S(x)$ $(S : GF(2^4) \mapsto GF(2^4))$ can be decomposed into $G(x)$ $(G : GF(2^4) \mapsto GF(2^4))$ and $F(x)(F : GF(2^4) \mapsto GF(2^4))$, where $S(x) = F(G(x))$. Each of $F(x)$ and $G(x)$ is split into three shares as shown in Figure 3.2. The equations for uniform sharing of F and G are given below. It is similar to the shares in the TI of the PRESENT S-Box as shown in [PMK+11] Note that inputs and outputs are all 4-bit values.

$$
\begin{aligned}
G_1[3] =& m_2[2] \oplus m_2[1] \oplus m_2[0] \\
G_1[2] =& 1 \oplus m_2[2] \oplus m_2[1] \\
G_1[1] =& 1 \oplus m_2[3] \oplus m_2[1] \oplus (m_2[2] \times m_2[0]) \oplus (m_2[2] \times m_3[0]) \oplus (m_3[2] \times m_2[0]) \oplus (m_2[1] \times m_2[0]) \\
& \oplus (m_2[1] \times m_3[0]) \oplus (m_3[1] \times m_2[0]) \\
G_1[0] =& 1 \oplus m_2[0] \oplus (m_2[3] \times m_2[2]) \oplus (m_2[3] \times m_3[2]) \oplus (m_3[3] \times m_2[2]) \oplus (m_2[3] \times m_2[1]) \\
& \oplus (m_2[3] \times m_3[1]) \oplus (m_3[3] \times m_2[1]) \oplus (m_2[2] \times m_2[1]) \oplus (m_2[2] \times m_3[1]) \oplus (m_3[2] \times m_2[1])
\end{aligned}
$$

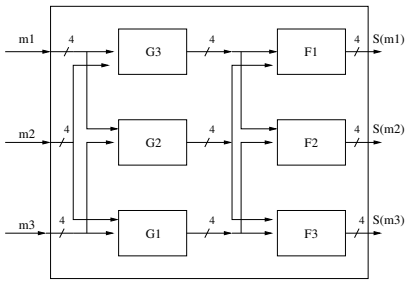


Figure 3.2: Decomposed S-box with three shares

$$
\begin{aligned}
G_2[3] =& m_3[2] \oplus m_3[1] \oplus m_3[0] \\
G_2[2] =& m_3[2] \oplus m_3[1] \\
G_2[1] =& m_3[3] \oplus m_3[1] \oplus (m_3[2] \times m_3[0]) \oplus (m_1[2] \times m_3[0]) \oplus (m_3[2] \times m_1[0]) \oplus (m_3[1] \times m_3[0]) \\
& \oplus (m_1[1] \times m_3[0]) \oplus (m_3[1] \times m_1[0]) \\
G_2[0] =& m_3[0] \oplus (m_3[3] \times m_3[2]) \oplus (m_1[3] \times m_3[2]) \oplus (m_3[3] \times m_1[2]) \oplus (m_3[3] \times m_3[1]) \\
& \oplus (m_1[3] \times m_3[1]) \oplus (m_3[3] \times m_1[1]) \oplus (m_3[2] \times m_3[1]) \oplus (m_1[2] \times m_3[1]) \oplus (m_3[2] \times m_1[1]) \\
G_3[3] =& m_1[2] \oplus m_1[1] \oplus m_1[0] \\
G_3[2] =& m_1[2] \oplus m_1[1] \\
G_3[1] =& m_1[3] \oplus m_1[1] \oplus (m_1[2] \times m_1[0]) \oplus (m_1[2] \times m_2[0]) \oplus (m_2[2] \times m_1[0]) \oplus (m_1[1] \times m_1[0]) \\
& \oplus (m_1[1] \times m_2[0]) \oplus (m_2[1] \times m_1[0]) \\
G_3[0] =& m_1[0] \oplus (m_1[3] \times m_1[2]) \oplus (m_1[3] \times m_2[2]) \oplus (m_2[3] \times m_1[2]) \oplus (m_1[3] \times m_1[1]) \\
& \oplus (m_1[3] \times m_2[1]) \oplus (m_2[3] \times m_1[1]) \oplus (m_1[2] \times m_1[1]) \oplus (m_1[2] \times m_2[1]) \oplus (m_2[2] \times m_1[1]) \\
F_1[3] =& G_2[2] \oplus G_2[1] \oplus G_2[0] \oplus (G_2[3] \times G_2[0]) \oplus (G_2[3] \times G_3[0]) \oplus (G_3[3] \times G_2[0]) \\
F_1[2] =& G_2[3] \oplus (G_2[1] \times G_2[0]) \oplus (G_2[1] \times G_3[0]) \oplus (G_3[1] \times G_2[0]) \\
F_1[1] =& G_2[2] \oplus G_2[1] \oplus (G_2[3] \times G_2[0]) \oplus (G_2[3] \times G_3[0]) \oplus (G_3[3] \times G_2[0]) \\
F_1[0] =& G_2[1] \oplus (G_2[2] \times G_2[0]) \oplus (G_2[2] \times G_3[0]) \oplus (G_3[2] \times G_2[0]) \\
F_2[3] =& G_3[2] \oplus G_3[1] \oplus G_3[0] \oplus (G_3[3] \times G_3[0]) \oplus (G_1[3] \times G_3[0]) \oplus (G_3[3] \times G_1[0]) \\
F_2[2] =& G_3[3] \oplus (G_3[1] \times G_3[0]) \oplus (G_1[1] \times G_3[0]) \oplus (G_3[1] \times G_1[0]) \\
F_2[1] =& G_3[2] \oplus G_3[1] \oplus (G_3[3] \times G_3[0]) \oplus (G_1[3] \times G_3[0]) \oplus (G_3[3] \times G_1[0]) \\
F_2[0] =& G_3[1] \oplus (G_3[2] \times G_3[0]) \oplus (G_1[2] \times G_3[0]) \oplus (G_3[2] \times G_1[0]) \\
F_3[3] =& G_1[2] \oplus G_1[1] \oplus G_1[0] \oplus (G_1[3] \times G_1[0]) \oplus (G_1[3] \times G_2[0]) \oplus (G_2[3] \times G_1[0]) \\
F_3[2] =& G_1[3] \oplus (G_1[1] \times G_1[0]) \oplus (G_1[1] \times G_2[0]) \oplus (G_2[1] \times G_1[0]) \\
F_3[1] =& G_1[2] \oplus G_1[1] \oplus (G_1[3] \times G_1[0]) \oplus (G_1[3] \times G_2[0]) \oplus (G_2[3] \times G_1[0]) \\
F_3[0] =& G_1[1] \oplus (G_1[2] \times G_1[0]) \oplus (G_1[2] \times G_2[0]) \oplus (G_2[2] \times G_1[0])
\end{aligned}
$$

The F-function block for KHUDRA, as shown in Figure 3.3, has 6 rounds, and iterates over 24 clock cycles, where in each 4 clock cycles one round of F-function gets executed. So, the left and right branch of F-function that pass through S-box takes 48 clock cycles to complete the whole F-function round and in total 864 (48×64) clock cycles for 18 rounds. The *TI* shared S-box will have one instance while other components will have three instances, one for each share.

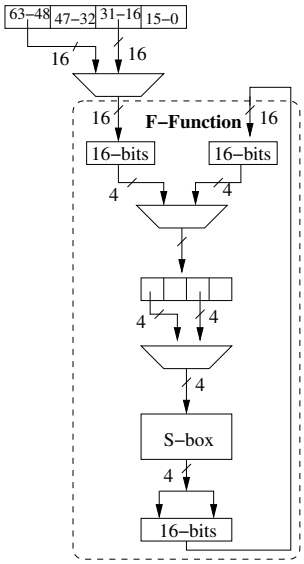


Figure 3.3: KHUDRA Serial Implementation

3.2.1 Test Vector Leakage Assessment (TVLA): T-Test Methodology

The TVLA test is a conformance test which attempts to detect the presence of any leakage in a cryptographic core. The block cipher hardware is made to operate on a constant selected plaintext, and the power consumption is compared with when the cipher operates on randomly chosen inputs. The existence of any differentiability denotes the presence of leakage of information which can be potentially exploited by an adversary for key recovery. The basis of the test is statistical hypothesis testing using Welch's T-test.

Suppose, the number of power traces collected for a fixed plaintext denoted by $\mid \mathscr{A} \mid$ for set $\mathscr{A}$, and number of power traces processing random inputs denoted by $\mid \mathscr{B} \mid$ for set $\mathscr{B}$. The sample mean, the variance for $\mathscr{A}$ is denoted by $\mu_{\mathscr{A}}$ and $\sigma^2_{\mathscr{A}}$ respectively and similar for $\mathscr{B}$ as well. Then a null hypothesis is made with $\mu_{\mathscr{A}}{=}\mu_{\mathscr{B}}$, after which Welch's T-test is used to accept or reject the null hypothesis with a confidence of 99.9%. The formula for T-test is shown below:

$$t = \frac{\mu_{\mathscr{A}} - \mu_{\mathscr{B}}}{\sqrt{\frac{\sigma^2_{\mathscr{A}}}{|\mathscr{A}|} + \frac{\sigma^2_{\mathscr{B}}}{|\mathscr{B}|}}}.$$

If the value of $t >\mid 4.5 \mid$, then the null hypothesis is rejected and the cryptographic algorithm is said to fail first-order leakage.

Our design had t values within the permitted range for 10000 traces thus experimentally validating the security of our design.

Table 3.1: Hardware Overhead Comparison With and Without Threshold Implementation in Terms of Gate Equivalents

Block Cipher	**Unprotected(GE)**	**Protected(GE)**
KHUDRA-64/80 [KM14]	1090	3738
PRESENT-64/80 [CN17]	1619.23	5236.49
SIMON-128/128 [STE17]	1234	5686
SPECK-128/128 [YZS$^+$15]	2018	5940.6

3.2.2 Area comparison with other lightweight protected and unprotected block ciphers

Table 3.1 shows that the threshold implementation of KHUDRA is one of the smallest protected implementations among all block ciphers.

3.3 Conclusion

In the present age of ubiquitous computing, extending classical attacker models is the need of the time. Physical attacks need to be taken into account simultaneously dealing with fierce area and power constraints. This scenario calls for lightweight solutions. Almost every side-channel countermeasures introduce power and area overhead which are proportional to the values of the unprotected implementation for most of the commonly used block ciphers. This fact prohibits the implementation of a wide range of proposed countermeasures and also narrows down possible cipher candidates. In this work, we have designed a 3-shared threshold implementation of the lightweight block cipher KHUDRA. We see that it has very small area requirement compared to other protected implementations of block ciphers. This makes the threshold implementation of KHUDRA an ideal choice for lightweight applications. A future direction of research would be higher order threshold implementations of KHUDRA against adversaries who are equipped to surmount higher order differential fault attacks.

Chapter 4

Several Masked Implementations of the Boyar Peralta AES S-Box

Threshold implementation is a masking technique that provides provable security for implementations of cryptographic algorithms against power analysis attacks. In recent publications, several different threshold implementations of AES have been designed. However in most of the threshold implementations of AES, the Canright S-Box has been used. The Boyar-Peralta S-Box is an alternative implementation of the AES S-Box with a minimal circuit depth and is comparable in size to the frequently used Canright AES S-Box. In this chapter, we present several versions of first-order threshold implementations of the Boyar-Peralta AES S-Box with different number of shares and several trade-offs in area, randomness and speed. To the best of our knowledge these are the first threshold implementations of the Boyar-Peralta S-Box. Our implementations compare favourably with some of the existing threshold implementations of Canright S-Box along the design trade-offs, e.g. while one of our S-Boxes is 49% larger in area than the smallest known threshold implementation of the Canright AES S-Box, it uses 63% less randomness and requires only 50% of the clock cycles. We provide results of a practical security evaluation based on real power traces to confirm the first-order attack resistance of our implementations.

4.1 Introduction

In a black box model, embedded devices have been shown to be secure using modern ciphers. However, when naively implemented, side-channel information like power consumption, electromagnetic radiations or timing of the device's computations can leak secret information unintentionally. Attacks based on various side channels were presented in [GMO01, Koc96, KJJ99] and their mitigation has been the subject of a great deal of research ever since.

Masking is an efficient way to strengthen cryptographic implementations against such physical side-channel attacks [CJRR99, GP99]. Masking detaches leaked side-channel information from secret dependent intermediate values by carrying out computations on randomized values. It offers provable security [PR13] and can be implemented on the algorithmic level, making it a flexible Side-Channel Analysis (SCA) countermeasure. The underlying principle of masking relies on splitting each variable into a set of random values using secret sharing techniques and using a certain multi-party computation protocol on the resulting random values for secure computations. Once the secret values are masked, they are in no way combined until the end of the algorithm, i.e. the sensitive values are not leaked at any point during the execution of the cryptographic algorithm. Only at the end of the computation, when the cipher's outputs are valid, the output masks are combined to reconstruct the unmasked output.

The security of masking schemes is inherently tied to an adversary model. An attacker who observes the d^{th}-order statistical moment of e.g. a power trace or combines observations from d points in time nonlinearly in that power trace is said to be an attacker mounting a d^{th}-order attack. To prevent a d^{th}-order attack, a masking scheme of order $(d+1)$ is required. Fortunately, the number of readings needed for a higher-order attack to become successful grows exponentially with the noise standard deviation and therefore it is reasonable to guarantee practical security up to a certain order.

Implementing masking in hardware in a secure manner is not trivial. It is a delicate job since all the assumptions made on the leakage behavior of the underlying platform do not always hold in practice. For example, glitches are a known predominant threat [MPO05] to the security of masked implementations in CMOS technologies. Some masking schemes like Threshold Implementations (TI) work under assumptions which are more achievable in a practical scenarios. In addition to these relaxed assumptions on the underlying leakage, TI offers provable security and allows to construct secure circuits which are realistic in size, all without requiring much intervention from a designer or many design iterations. For this reason, TI has been applied to many well-known cryptographic algorithms like KECCAK, AES and PRESENT [BDN+13, DCBR+15, MPL+11, PMK+11].

The Canright S-Box [Can05] and Boyar-Peralta S-Box [BP12] are two of the smallest implementations of the AES S-Box. As a starting point for threshold implementations and Side-Channel Analysis (SCA) secure designs, the Canright S-box has been used predominantly [MPL+11, BGN+14b, GMK16], whereas the Boyar-Peralta S-box has received little to no attention. The S-box introduced by Boyar and Peralta [BP12] is based on a novel logic minimization technique, which can be applied to any arbitrary combinational logic

problems and even circuits that have been optimized by standard methodologies. The authors described their techniques as a two-step process: a reduction of nonlinear gates and a reduction of linear gates. Using their method they came up with an S-Box for AES which has the smallest combinational circuit depth known till date.

The aim of this chapter is to develop secure masked implementations of the Boyar-Peralta AES S-Box using TI. The Boyar-Peralta S-Box is one of the smallest circuits implementing the AES S-box in unmasked form. We explore whether it is also one of the smallest masked S-Box of AES. For this purpose we explore several different masking styles of the Boyar-Peralta S-Box, focusing on various trade-offs between area, randomness and the number of clock cycles.

4.1.1 Contributions.

We present the first threshold implementations of the Boyar-Peralta AES S-Box. More precisely, we show TIs of the Boyar-Peralta AES S-Box with 3 and 4 shares, both with various trade-offs related to the circuit area, the consumed randomness and the required clock cycles. We consider two approaches to mask the S-Box. The first approach involves masking the AND gates alone using uniform sharing of the individual AND gates. The second approach is based on sharing a larger algebraic function, the $\mathbb{GF}(2^4)$ inverter as a whole.

Our smallest implementation is 2.75% larger in area than the smallest Canright S-Box presented in [BGN+15] but reduces randomness required by 37.5% and takes the same number of clock cycles. This implementation of ours which is the smallest in area takes as many clock cycles as the fastest known Threshold Implementation of the Canright S-Box. The Canright S-Box in [DCRB+16] is the smallest known TI of the AES S-Box so far. Our smallest implementation is 47% larger in area but reduces randomness by 63% and increases speed by 50%. One of our implementations uses no randomness at all while all known threshold implementations of the Canright S-Box need randomness. We show the results of leakage detection tests of our implementations on a low noise FPGA platform to back up the theoretical security.

4.1.2 Organization.

In Section 4.2, we provide the notation and the theory behind the threshold implementations masking scheme and the Boyar-Peralta AES S-Box. In Section 4.3, we develop the various secure implementations of the Boyar-Peralta S-Box by successively reducing either the number of shares, or the required randomness when the number of shares is kept constant. We present the results of the side-channel analysis in Section 4.4. In Section 4.5,

we discuss the implementation cost of our resulting designs and compare them with costs of related previously published threshold implementations. We conclude the chapter and propose directions for future work in Section 4.6.

4.2 Preliminaries

4.2.1 Notation

We use lowercase regular and bold letters to describe elements of $\mathbb{GF}(2^n)$ and their sharing respectively. Any sensitive variable $x \in \mathbb{GF}(2^n)$ is split into s shares $(x_1, ..., x_s) = \mathbf{x}$, where $x_i \in \mathbb{GF}(2^n)$, in the initialization phase of the cryptographic algorithm. A possible manner of performing this initialization, which we employ, is as follows: the shares $x_1, x_2, ..., x_{s-1}$ are selected randomly from an uniform distribution and x_s is calculated such that $x = \bigoplus_{i \in \{1,2,....,s\}} x_i$. We refer to the j^{th} bit of x as x^j unless $x \in \mathbb{GF}(2)$. We use the same notation to share a function f into s shares $\mathbf{f} = (f_1, ..., f_s)$. The number of input and output shares of $\mathbf{f}$ are denoted by s_{in} and s_{out} respectively. We refer to field multiplication as $\times$, to addition as $\oplus$ and denote negation of all bits in a value x using $\overline{x}$.

4.2.2 The Boyar-Peralta Implementation of the AES S-Box

The Boyar-Peralta S-Box, is a circuit of depth 16 introduced by Boyar and Peralta [BP12]. It uses a total of 128 2-input gates to construct the S-Box: 94 gates are linear operations (XOR and XNOR gates) and 34 gates are nonlinear (AND gates or 1-bit multiplications).

The circuit is divided into 3 layers:

1. the top linear layer
2. the middle nonlinear layer
3. the bottom linear layer

The equations involved are listed below. The 8 input bits are given by u_0, u_1, u_2, u_3, u_4, u_5, u_6 and u_7 with u_0 being the most significant bit and u_7 being the least significant bit. Similarly, the 8 output bits are given by s_0, s_1, s_2, s_3, s_4, s_5, s_6 and s_7, with s_0 being the most significant bit and s_7 being the least significant bit.

The set of equations for the top linear layer are:

$$t_1 = u_0 \oplus u_3 \qquad t_4 = u_3 \oplus u_5 \qquad t_7 = u_1 \oplus u_2$$

$$t_2 = u_0 \oplus u_5 \qquad t_5 = u_4 \oplus u_6 \qquad t_8 = u_7 \oplus t_6$$

$$t_3 = u_0 \oplus u_6 \qquad t_6 = t_1 \oplus t_5 \qquad t_9 = u_7 \oplus t_7$$

$$t_{10} = t_6 \oplus t_7 \qquad t_{16} = t_5 \oplus t_{12} \qquad t_{22} = t_7 \oplus t_{21}$$

$$t_{11} = u_1 \oplus u_5 \qquad t_{17} = t_9 \oplus t_{16} \qquad t_{23} = t_2 \oplus t_{22}$$

$$t_{12} = u_2 \oplus u_5 \qquad t_{18} = u_3 \oplus u_7 \qquad t_{24} = t_2 \oplus t_{10}$$

$$t_{13} = t_3 \oplus t_4 \qquad t_{19} = t_7 \oplus t_{18} \qquad t_{25} = t_{20} \oplus t_{17}$$

$$t_{14} = t_6 \oplus t_{11} \qquad t_{20} = t_1 \oplus t_{19} \qquad t_{26} = t_3 \oplus t_{16}$$

$$t_{15} = t_5 \oplus t_{11} \qquad t_{21} = u_6 \oplus u_7 \qquad t_{27} = t_1 \oplus t_{12}$$

The set of equations for the middle nonlinear layer are given by:

$$m_1 = t_{13} \times t_6 \qquad m_{18} = m_8 \oplus m_7 \qquad m_{35} = m_{24} \times m_{34}$$

$$m_2 = t_{23} \times t_8 \qquad m_{19} = m_{10} \oplus m_{15} \qquad m_{36} = m_{24} \oplus m_{25}$$

$$m_3 = t_{14} \oplus m_1 \qquad m_{20} = m_{16} \oplus m_{13} \qquad m_{37} = m_{21} \oplus m_{29}$$

$$m_4 = t_{19} \times u_7 \qquad m_{21} = m_{17} \oplus m_{15} \qquad m_{38} = m_{32} \oplus m_{33}$$

$$m_5 = m_4 \oplus m_1 \qquad m_{22} = m_{18} \oplus m_{13} \qquad m_{39} = m_{23} \oplus m_{30}$$

$$m_6 = t_3 \times t_{16} \qquad m_{23} = m_{19} \oplus t_{25} \qquad m_{40} = m_{35} \oplus m_{36}$$

$$m_7 = t_{22} \times t_9 \qquad m_{24} = m_{22} \oplus m_{23} \qquad m_{41} = m_{38} \oplus m_{40}$$

$$m_8 = t_{26} \oplus m_6 \qquad m_{25} = m_{22} \times m_{20} \qquad m_{42} = m_{37} \oplus m_{39}$$

$$m_9 = t_{20} \times t_{17} \qquad m_{26} = m_{21} \oplus m_{25} \qquad m_{43} = m_{37} \oplus m_{38}$$

$$m_{10} = m_9 \oplus m_6 \qquad m_{27} = m_{20} \oplus m_{21} \qquad m_{44} = m_{39} \oplus m_{40}$$

$$m_{11} = t_1 \times t_{15} \qquad m_{28} = m_{23} \oplus m_{25} \qquad m_{45} = m_{42} \oplus m_{41}$$

$$m_{12} = t_4 \times t_{27} \qquad m_{29} = m_{28} \times m_{27} \qquad m_{46} = m_{44} \times t_6$$

$$m_{13} = m_{12} \oplus m_{11} \qquad m_{30} = m_{26} \times m_{24} \qquad m_{47} = m_{40} \times t_8$$

$$m_{14} = t_2 \times t_{10} \qquad m_{31} = m_{20} \times m_{23} \qquad m_{48} = m_{39} \times u_7$$

$$m_{15} = m_{14} \oplus m_{11} \qquad m_{32} = m_{27} \times m_{31} \qquad m_{49} = m_{43} \times t_{16}$$

$$m_{16} = m_3 \oplus m_2 \qquad m_{33} = m_{27} \oplus m_{25} \qquad m_{50} = m_{38} \times t_9$$

$$m_{17} = m_5 \oplus t_{24} \qquad m_{34} = m_{21} \times m_{22} \qquad m_{51} = m_{37} \times t_{17}$$

$$m_{52} = m_{42} \times t_{15} \qquad m_{56} = m_{40} \times t_{23} \qquad m_{60} = m_{37} \times t_{20}$$

$$m_{53} = m_{45} \times t_{27} \qquad m_{57} = m_{39} \times t_{19} \qquad m_{61} = m_{42} \times t_{1}$$

$$m_{54} = m_{41} \times t_{10} \qquad m_{58} = m_{43} \times t_{3} \qquad m_{62} = m_{45} \times t_{4}$$

$$m_{55} = m_{44} \times t_{13} \qquad m_{59} = m_{38} \times t_{22} \qquad m_{63} = m_{41} \times t_{2}$$

The set of equations for the bottom linear layer consist of:

$$l_0 = m_{61} \oplus m_{62} \qquad l_{13} = m_{50} \oplus l_0 \qquad l_{26} = l_7 \oplus l_9$$

$$l_1 = m_{50} \oplus m_{56} \qquad l_{14} = m_{52} \oplus m_{61} \qquad l_{27} = l_8 \oplus l_{10}$$

$$l_2 = m_{46} \oplus m_{48} \qquad l_{15} = m_{55} \oplus l_1 \qquad l_{28} = l_{11} \oplus l_{14}$$

$$l_3 = m_{47} \oplus m_{55} \qquad l_{16} = m_{56} \oplus l_0 \qquad l_{29} = l_{11} \oplus l_{17}$$

$$l_4 = m_{54} \oplus m_{58} \qquad l_{17} = m_{57} \oplus l_1 \qquad s_0 = l_6 \oplus l_{24}$$

$$l_5 = m_{49} \oplus m_{61} \qquad l_{18} = m_{58} \oplus l_8 \qquad s_1 = \overline{l_{16} \oplus l_{26}}$$

$$l_6 = m_{62} \oplus l_5 \qquad l_{19} = m_{63} \oplus l_4 \qquad s_2 = \overline{l_{19} \oplus l_{28}}$$

$$l_7 = m_{46} \oplus l_3 \qquad l_{20} = l_0 \oplus l_1 \qquad s_3 = l_6 \oplus l_{21}$$

$$l_8 = m_{51} \oplus m_{59} \qquad l_{21} = l_1 \oplus l_7 \qquad s_4 = l_{20} \oplus l_{22}$$

$$l_9 = m_{52} \oplus m_{53} \qquad l_{22} = l_3 \oplus l_{12} \qquad s_5 = l_{25} \oplus l_{29}$$

$$l_{10} = m_{53} \oplus l_4 \qquad l_{23} = l_{18} \oplus l_2 \qquad s_6 = \overline{l_{13} \oplus l_{27}}$$

$$l_{11} = m_{60} \oplus l_2 \qquad l_{24} = l_{15} \oplus l_9 \qquad s_7 = \overline{l_6 \oplus l_{23}}$$

$$l_{12} = m_{48} \oplus m_{51} \qquad l_{25} = l_6 \oplus l_{10}$$

Masked software implementations of the Boyar Peralta AES S-Box were proposed in [JS17, GR17]. A modified version of the Boyar Peralta S-Box has been masked using the ISW AND gate [ISW03] in [GR17].

4.2.3 Threshold Implementations

The preliminaries of Threshold Implementations has been covered in the previous chapter.

4.3 Several SCA Secure Implementations of the Boyar-Peralta AES S-Box

In this section we present several different threshold implementations of the Boyar-Peralta AES S-Box. Applying TI to linear functions is straightforward due to the linearity of the XOR and XNOR operations. Masking the nonlinear functions on the other hand is known to pose more of a challenge. As mentioned in the previous section the only nonlinear functions in the Boyar-Peralta AES S-Box are the AND gates. In order to apply TI to these AND gates we need to make sure the resulting sharings are non-complete and correct, and that their outputs are uniform. In our first approach, we therefore consider the uniform sharing of an AND gate and formulate several 1^{st}-order non-complete TI sharings for this S-box. We additionally investigate a second approach: instead of masking each AND gate individually, we combine several AND gates to form an inversion in $\mathbb{GF}(2^4)$. In both cases, to avoid first-order leakages from glitches and early propagation of signals, each masked nonlinear function must be followed by a set of registers.

The middle layer is the nonlinear layer in the Boyar-Peralta AES S-Box. The top and the bottom layer are composed of linear functions only. When we mask each gate individually, the outputs of every AND gate in the middle layer must be registered before the next operation starts. Hence, we divide the middle layer into stages such that at each stage, the outputs produced by the AND gates are put into registers before proceeding for the operation in the next stage.

On inspection of the set of equations, we divide the circuit into 4 stages where each stage ends with a set of AND operations. Note that there may be other ways to divide the nonlinear layer into stages. The top linear layer was combined with the first stage of the nonlinear layer and the outputs of the AND gates from the 4^{th} and final stage of the middle nonlinear layer are fed into the bottom layer directly, which causes no problem since this layer is linear. Therefore, we divide our circuit into 4 stages with a set of registers after the first three stages. A total of 4 clock cycles are required to complete the computation of the S-Box. The entire circuit of the nonlinear middle layer is shown in Figure 4.1. The set of equations after division into stages are given below.

Stage 1.

$$t_1 = u_0 \oplus u_3 \qquad t_4 = u_3 \oplus u_5 \qquad t_7 = u_1 \oplus u_2$$

$$t_2 = u_0 \oplus u_5 \qquad t_5 = u_4 \oplus u_6 \qquad t_8 = u_7 \oplus t_6$$

$$t_3 = u_0 \oplus u_6 \qquad t_6 = t_1 \oplus t_5 \qquad t_9 = u_7 \oplus t_7$$

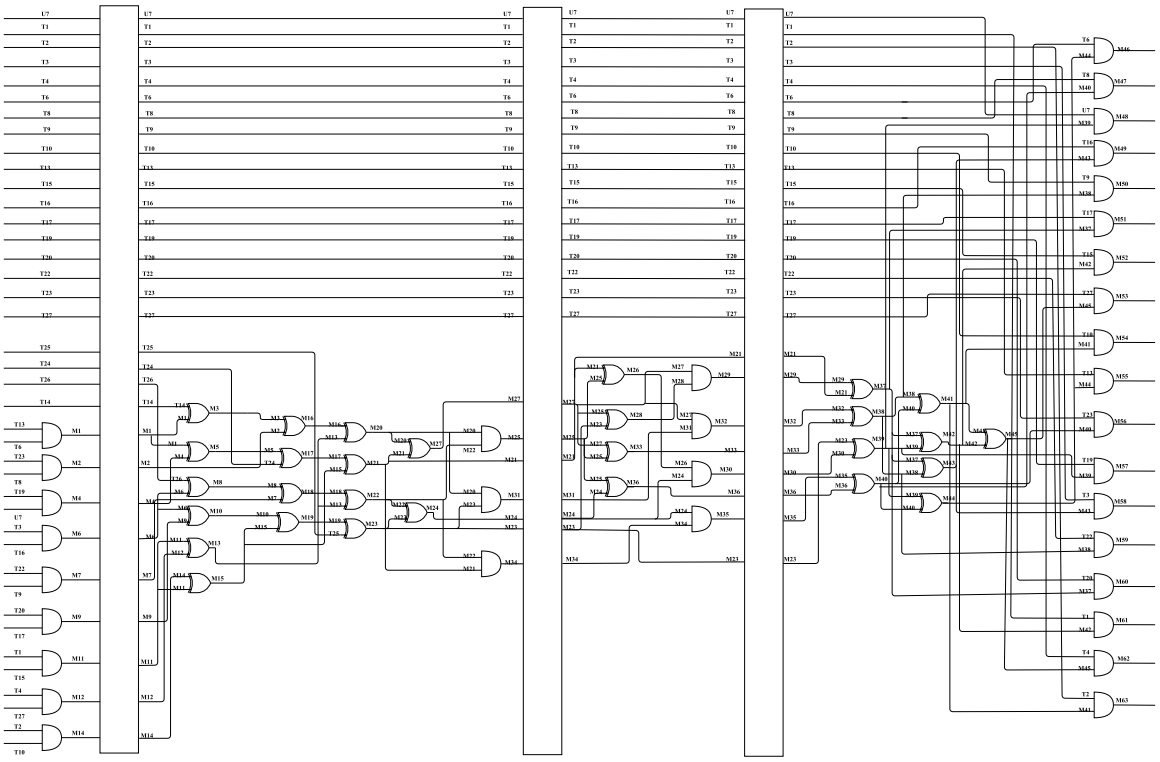


Figure 4.1: Division of the nonlinear layer into stages.

$$t_{10} = t_6 \oplus t_7 \qquad t_{19} = t_7 \oplus t_{18} \qquad m_1 = t_{13} \times t_6$$

$$t_{11} = u_1 \oplus u_5 \qquad t_{20} = t_1 \oplus t_{19} \qquad m_2 = t_{23} \times t_8$$

$$t_{12} = u_2 \oplus u_5 \qquad t_{21} = u_6 \oplus u_7 \qquad m_4 = t_{19} \times u_7$$

$$t_{13} = t_3 \oplus t_4 \qquad t_{22} = t_7 \oplus t_{21} \qquad m_6 = t_3 \times t_{16}$$

$$t_{14} = t_6 \oplus t_{11} \qquad t_{23} = t_2 \oplus t_{22} \qquad m_7 = t_{22} \times t_9$$

$$t_{15} = t_5 \oplus t_{11} \qquad t_{24} = t_2 \oplus t_{10} \qquad m_9 = t_{20} \times t_{17}$$

$$t_{16} = t_5 \oplus t_{12} \qquad t_{25} = t_{20} \oplus t_{17} \qquad m_{11} = t_1 \times t_{15}$$

$$t_{17} = t_9 \oplus t_{16} \qquad t_{26} = t_3 \oplus t_{16} \qquad m_{12} = t_4 \times t_{27}$$

$$t_{18} = u_3 \oplus u_7 \qquad t_{27} = t_1 \oplus t_{12} \qquad m_{14} = t_2 \times t_{10}$$

Stage 2.

$$m_3 = t_{14} \oplus m_1 \qquad m_8 = t_{26} \oplus m_6 \qquad m_{13} = m_{12} \oplus m_{11}$$

$$m_5 = m_4 \oplus m_1 \qquad m_{10} = m_9 \oplus m_6 \qquad m_{15} = m_{14} \oplus m_{11}$$

$m_{16} = m_3 \oplus m_2$ $m_{21} = m_{17} \oplus m_{15}$ $m_{27} = m_{20} \oplus m_{21}$

$m_{17} = m_5 \oplus t_{24}$ $m_{22} = m_{18} \oplus m_{13}$

$m_{18} = m_8 \oplus m_7$ $m_{23} = m_{19} \oplus t_{25}$ $m_{31} = m_{20} \times m_{23}$

$m_{19} = m_{10} \oplus m_{15}$ $m_{24} = m_{22} \oplus m_{23}$

$m_{20} = m_{16} \oplus m_{13}$ $m_{25} = m_{22} \times m_{20}$ $m_{34} = m_{21} \times m_{22}$

Stage 3.

$m_{26} = m_{21} \oplus m_{25}$ $m_{30} = m_{26} \times m_{24}$ $m_{35} = m_{24} \times m_{34}$

$m_{28} = m_{23} \oplus m_{25}$ $m_{32} = m_{27} \times m_{31}$

$m_{29} = m_{28} \times m_{27}$ $m_{33} = m_{27} \oplus m_{25}$ $m_{36} = m_{24} \oplus m_{25}$

Stage 4.

$m_{37} = m_{21} \oplus m_{29}$ $m_{51} = m_{37} \times t_{17}$ $l_1 = m_{50} \oplus m_{56}$

$m_{38} = m_{32} \oplus m_{33}$ $m_{52} = m_{42} \times t_{15}$ $l_2 = m_{46} \oplus m_{48}$

$m_{39} = m_{23} \oplus m_{30}$ $m_{53} = m_{45} \times t_{27}$ $l_3 = m_{47} \oplus m_{55}$

$m_{40} = m_{35} \oplus m_{36}$ $m_{54} = m_{41} \times t_{10}$ $l_4 = m_{54} \oplus m_{58}$

$m_{41} = m_{38} \oplus m_{40}$ $m_{55} = m_{44} \times t_{13}$ $l_5 = m_{49} \oplus m_{61}$

$m_{42} = m_{37} \oplus m_{39}$ $m_{56} = m_{40} \times t_{23}$ $l_6 = m_{62} \oplus l_5$

$m_{43} = m_{37} \oplus m_{38}$ $m_{57} = m_{39} \times t_{19}$ $l_7 = m_{46} \oplus l_3$

$m_{44} = m_{39} \oplus m_{40}$ $m_{58} = m_{43} \times t_3$ $l_8 = m_{51} \oplus m_{59}$

$m_{45} = m_{42} \oplus m_{41}$ $m_{59} = m_{38} \times t_{22}$ $l_9 = m_{52} \oplus m_{53}$

$m_{46} = m_{44} \times t_6$ $m_{60} = m_{37} \times t_{20}$ $l_{10} = m_{53} \oplus l_4$

$m_{47} = m_{40} \times t_8$ $m_{61} = m_{42} \times t_1$ $l_{11} = m_{60} \oplus l_2$

$m_{48} = m_{39} \times u_7$ $m_{62} = m_{45} \times t_4$ $l_{12} = m_{48} \oplus m_{51}$

$m_{49} = m_{43} \times t_{16}$ $m_{63} = m_{41} \times t_2$ $l_{13} = m_{50} \oplus l_0$

$m_{50} = m_{38} \times t_9$ $l_0 = m_{61} \oplus m_{62}$ $l_{14} = m_{52} \oplus m_{61}$

$l_{15} = m_{55} \oplus l_1$
$l_{16} = m_{56} \oplus l_0$
$l_{17} = m_{57} \oplus l_1$
$l_{18} = m_{58} \oplus l_8$
$l_{19} = m_{63} \oplus l_4$
$l_{20} = l_0 \oplus l_1$
$l_{21} = l_1 \oplus l_7$
$l_{22} = l_3 \oplus l_{12}$
$l_{23} = l_{18} \oplus l_2$
$l_{24} = l_{15} \oplus l_9$
$l_{25} = l_6 \oplus l_{10}$
$l_{26} = l_7 \oplus l_9$
$l_{27} = l_8 \oplus l_{10}$
$l_{28} = l_{11} \oplus l_{14}$
$l_{29} = l_{11} \oplus l_{17}$
$s_0 = l_6 \oplus l_{24}$
$s_1 = \overline{l_{16} \oplus l_{26}}$
$s_2 = \overline{l_{19} \oplus l_{28}}$
$s_3 = l_6 \oplus l_{21}$
$s_4 = l_{20} \oplus l_{22}$
$s_5 = l_{25} \oplus l_{29}$
$s_6 = \overline{l_{13} \oplus l_{27}}$
$s_7 = \overline{l_6 \oplus l_{23}}$

For the second approach, where we mask the circuit using the inversion in $\mathbb{GF}(2^4)$ instead of masking each individual AND gate. $m_{20}m_{21}m_{22}m_{23}$ are inputs to the $\mathbb{GF}(2^4)$ inverter and $m_{36}m_{32}m_{39}m_{28}$ being the output where m_{20} and m_{36} are the most significant bits of the input and output respectively. $m_{20}, m_{21}, m_{22}, m_{23}$ become available in Stage 2. The part of the circuit in Stage 2 to obtain $m_{20}, m_{21}, m_{22}, m_{23}$ is linear. Hence we can put the inverter right after $m_{20}, m_{21}, m_{22}, m_{23}$ become available without using a register. The outputs of the inverter $m_{36}, m_{32}, m_{39}, m_{28}$ were the outputs of Stage 3. Therefore, we combine Stage 2 and 3 to isolate the inverter. The modified set of equations are given below:

Stage 1.

$t_1{=}u_0 \oplus u_3$
$t_2{=}u_0 \oplus u_5$
$t_3{=}u_0 \oplus u_6$
$t_4{=}u_3 \oplus u_5$
$t_{13}{=}t_3 \oplus t_4$
$t_5{=}u_4 \oplus t_{13}$
$t_6{=}t_5 \oplus u_5$
$t_7{=}u_1 \oplus u_2$
$t_8{=}u_7 \oplus t_6$
$t_9{=}u_7 \oplus t_7$
$t_{10}{=}t_6 \oplus t_7$
$t_{14}{=}t_5 \oplus u_1$
$t_{15}{=}t_{14} \oplus t_1$
$t_{16}{=}t_7 \oplus t_{15}$
$t_{17}{=}t_9 \oplus t_{16}$
$t_{19}{=}t_9 \oplus u_3$
$t_{20}{=}t_1 \oplus t_{19}$
$t_{22}{=}t_9 \oplus u_6$
$t_{23}{=}t_2 \oplus t_{22}$
$t_{24}{=}t_2 \oplus t_{10}$
$t_{25}{=}t_{20} \oplus t_{17}$
$t_{26}{=}t_3 \oplus t_{16}$
$t_{27}{=}t_{10} \oplus t_{15}$
$m_1{=}t_{13} \times t_6$
$m_2{=}t_{23} \times t_8$
$m_3{=}m_2 \oplus m_1$
$m_4{=}t_{19} \times u_7$
$m_5{=}m_4 \oplus m_1$
$m_6{=}t_3 \times t_{16}$
$m_7{=}t_{22} \times t_9$

m_8=$m_7 \oplus m_6$ m_{11}=$t_1 \times t_{15}$ m_{14}=$t_2 \times t_{10}$

m_9=$t_{20} \times t_{17}$ m_{12}=$t_4 \times t_{27}$

m_{10}=$m_9 \oplus m_6$ m_{13}=$m_{12} \oplus m_{11}$

Stage 2.

m_{15}=$m_{14} \oplus m_{11}$ m_{18}=$m_8 \oplus m_{13}$ m_{21}=$m_{17} \oplus t_{24}$

m_{16}=$m_3 \oplus m_{13}$ m_{19}=$m_{10} \oplus m_{15}$ m_{22}=$m_{18} \oplus t_{26}$

m_{17}=$m_5 \oplus m_{15}$ m_{20}=$m_{16} \oplus t_{14}$ m_{23}=$m_{19} \oplus t_{25}$

$m_{20}m_{21}m_{22}m_{23}$ are inputs to the $\mathbb{GF}(2^4)$ inverter and $m_{36}m_{32}m_{39}m_{28}$ being the output where m_{20} and m_{36} are the most significant bits of the input and output respectively.

Stage 3.

m_{40}=$m_{39} \oplus m_{36}$ z_9=$m_{43} \times t_{13}$ l_6=$z_3 \oplus z_4$

m_{41}=$m_{28} \oplus m_{32}$ z_{10}=$m_{36} \times t_{23}$ l_7=$z_{12} \oplus l_4$

m_{42}=$m_{28} \oplus m_{39}$ z_{11}=$m_{32} \times t_{19}$ l_8=$z_7 \oplus l_6$

m_{43}=$m_{32} \oplus m_{36}$ z_{12}=$m_{42} \times t_3$ l_9=$z_8 \oplus l_7$

m_{44}=$m_{40} \oplus m_{41}$ z_{13}=$m_{39} \times t_{22}$ l_{10}=$l_8 \oplus l_9$

z_0=$m_{43} \times t_6$ z_{14}=$m_{28} \times t_{20}$ l_{11}=$l_6 \oplus l_5$

z_1=$m_{36} \times t_8$ z_{15}=$m_{41} \times t_1$ l_{12}=$z_3 \oplus z_5$

z_2=$m_{32} \times u_7$ z_{16}=$m_{44} \times t_4$ l_{13}=$z_{13} \oplus l_1$

z_3=$m_{42} \times t_{16}$ z_{17}=$m_{40} \times t_2$ l_{14}=$l_4 \oplus l_{12}$

z_4=$m_{39} \times t_9$ l_1=$z_{15} \oplus z_{16}$ s_3=$l_3 \oplus l_{11}$

z_5=$m_{28} \times t_{17}$ l_2=$z_{10} \oplus l_1$ l_{16}=$z_6 \oplus l_8$

z_6=$m_{41} \times t_{15}$ l_3=$z_9 \oplus l_2$ l_{17}=$z_{14} \oplus l_{10}$

z_7=$m_{44} \times t_{27}$ l_4=$z_0 \oplus z_2$ l_{18}=$l_{13} \oplus l_{14}$

z_8=$m_{40} \times t_{10}$ l_5=$z_1 \oplus z_0$ s_7=$\overline{z_{12} \oplus l_{18}}$

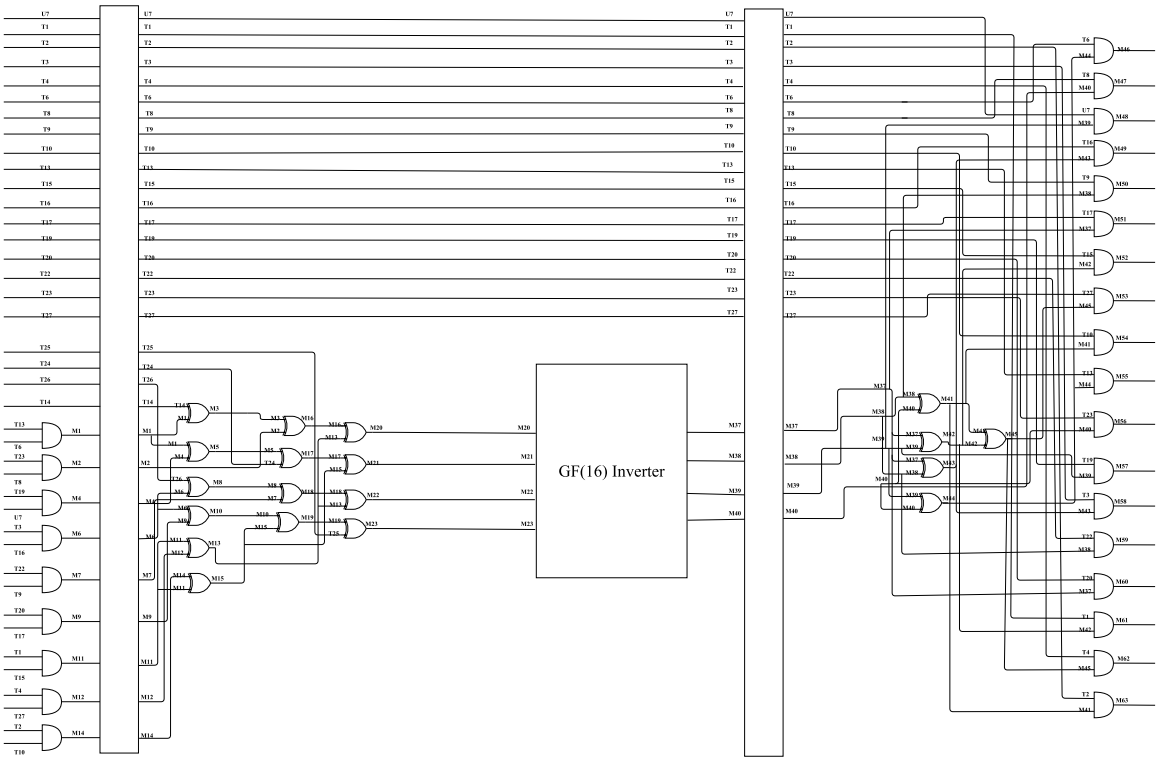


Figure 4.2: Division of the nonlinear layer into stages when centered around the inversion in $\mathbb{GF}(2^4)$.

l_{20}=$z_{15} \oplus l_{16}$ s_6=$\overline{l_{10} \oplus l_{18}}$ l_{26}=$l_{17} \oplus l_{20}$

l_{21}=$l_2 \oplus z_{11}$ s_4=$l_{14} \oplus s_3$ s_2=$\overline{l_{26} \oplus z_{17}}$

s_0=$l_3 \oplus l_{16}$ s_1=$\overline{s_3 \oplus l_{16}}$ s_5=$l_{21} \oplus l_{17}$

The circuit of the middle nonlinear layer using an inverter is shown in figure 4.2.

Using these two different approaches for division into stages of the circuit, we design the following secure implementations of the Boyar-Peralta S-Box:

1. Threshold implementation with 4 shares and no randomness in Section 3.1
2. Threshold implementation with 3 shares and 68 bits randomness in Section 3.2
3. Threshold implementation with 3 shares and 34 bits of randomness in Section 3.3
4. Threshold Implementation using 3 shares and using sharing with $s_{in} = 5$ and $s_{out} = 5$ for a $\mathbb{GF}(2^4)$ inverter in Section 3.4
5. Threshold Implementation using 3 shares and using sharing with $s_{in} = 4$ and $s_{out} = 4$ for a $\mathbb{GF}(2^4)$ inverter in Section 3.5

4.3.1 Threshold implementation with 4 shares and no randomness

As previously mentioned, the sharing for the linear operations is trivial. For the nonlinear AND gate we first use the following uniform 4-to-4 sharing. This is a novel modification of a 4-to-3 uniform sharing of the AND gate used in [BGN+14b].

$$\mathbf{a} = \mathbf{x} \times \mathbf{y}$$
$$\mathbf{x} = (x_1, x_2, x_3, x_4)$$
$$\mathbf{y} = (y_1, y_2, y_3, y_4)$$
$$\mathbf{a} = (a_1, a_2, a_3, a_4)$$
$$a_1 = (x_2 \oplus x_3 \oplus x_4) \times (y_2 \oplus y_3) \oplus y_3$$
$$a_2 = ((x_1 \oplus x_3) \times (y_1 \oplus y_4)) \oplus (x_1 \times y_3) \oplus x_4$$
$$a_3 = (x_2 \oplus x_4) \times (y_1 \oplus y_4) \oplus x_4 \oplus y_4$$
$$a_4 = (x_1 \times y_2) \oplus y_3$$

The complete computation of the S-Box will take 4 clock cycles and will not consume any randomness.

4.3.2 Threshold implementation with 3 shares and 68 bits randomness

Having designed a threshold implementation for the Boyar-Peralta AES S-Box which uses no randomness, we now aim to reduce the size of our circuit. This can be achieved by reducing the number of shares.

There is however no 3-to-3 uniform sharing for a 2-input AND gate. To keep the uniformity of sharing property intact, we introduce some randomness to remask the shares as shown in [MPL+11]. We use the following 3-to-3 sharing of the 2 input AND gate. r_1, r_2 are the 2 bits of randomness.

$$\mathbf{a} = \mathbf{x} \times \mathbf{y}$$
$$\mathbf{x} = (x_1, x_2, x_3)$$
$$\mathbf{y} = (y_1, y_2, y_3)$$
$$\mathbf{a} = (a_1, a_2, a_3)$$
$$a_1 = (x_2 \times y_2) \oplus (x_2 \times y_3) \oplus (x_3 \times y_2) \oplus r_1 \oplus r_2$$
$$a_2 = (x_3 \times y_3) \oplus (x_1 \times y_3) \oplus (x_3 \times y_1) \oplus r_2$$
$$a_3 = (x_1 \times y_1) \oplus (x_1 \times y_2) \oplus (x_2 \times y_1) \oplus r_1$$

One masked AND gate consumes 2-bits of randomness. The whole S-Box circuit requires 2 $\times$ 34 = 68 bits of randomness in total. The complete computation of the S-Box will take 4 clock cycles.

4.3.3 Threshold implementation with 3 shares and 34 bits randomness

We now reduce the amount of randomness required in our circuit by using the technique of virtual sharing as used in [BNN+12]. This sharing uses 1 bit of randomness per 2-input AND gate. The following is the resulting 3-to-3 sharing of the 2-input AND gate using 1 bit of randomness. r denotes a bit of randomness.

$$\mathbf{a} = \mathbf{x} \times \mathbf{y}$$
$$\mathbf{x} = (x_1, x_2, x_3)$$
$$\mathbf{y} = (y_1, y_2, y_3)$$
$$\mathbf{a} = (a_1, a_2, a_3)$$
$$a_1 = (x_2 \times y_2) \oplus (x_2 \times y_3) \oplus (x_3 \times y_2) \oplus r$$
$$a_2 = (x_3 \times y_3) \oplus (x_1 \times y_3) \oplus (x_3 \times y_1) \oplus (x_1 \times r) \oplus (y_1 \times r)$$
$$a_3 = (x_1 \times y_1) \oplus (x_1 \times y_2) \oplus (x_2 \times y_1) \oplus (x_1 \times r) \oplus (y_1 \times r) \oplus r$$

This S-Box circuit requires 34 bits of randomness. The complete computation of the S-Box will again take 4 clock cycles.

4.3.4 Threshold Implementation using 3 shares and using sharing with $s_{in} = 5$ and $s_{out} = 5$ for a $\mathbb{GF}(2^4)$ inverter

As stated earlier, we can isolate an inverter in $\mathbb{GF}(2^4)$ within the Boyar-Peralta S-Box. As shown in [BGN+14b] we can use a 5-to-5 uniform sharing for this $\mathbb{GF}(2^4)$ inverter. We use 3 shares for the linear and nonlinear gates that fall outside the inverter. In order to increase the number of shares from 3 to 5 at the input of the inverter we use 4 extra bits of randomness. To reduce the number of shares at the output from 5 to 3 we use 2 bits of randomness to combine the output shares just after the register. As mentioned in [Bil15] uniformity is necessary only for the input of nonlinear functions. The part of the circuit before the inverter in Stage 2 is linear. In order increase in the number of shares before input to the inverter, the shares are remasked using randomness. Therefore before input to the nonlinear part of Stage 2, the inverter, the shares are uniform due to remasking.

Hence inputs to stage 2 i.e. outputs of Stage 1 need not be uniform. Also all the outputs of the AND operations in stage 3 are inputs linear functions, hence they need not be uniform. This version has 27 2-input AND gates. All of them are in stages 1 and 3. Since the outputs of Stages 1 and 3 need not be uniform, none of the AND gates need to be uniform. So, we may use any non-complete and correct 3 sharing without using randomness for these AND gates. The total amount of randomness required is $4 \times 4 \times 4 + 2 \times 4 = 24$ bits.

4.3.5 Threshold Implementation using 3 shares and using sharing with $s_{in} = 4$ and $s_{out} = 4$ for a $\mathbb{GF}(2^4)$ inverter

Similar to the previous implementation, we again use the threshold implementation of the $\mathbb{GF}(2^4)$ inverter. There is 4-to-4 sharing of the $\mathbb{GF}(2^4)$ inverter which is not uniform. We observe that for decreasing the output shares from 4 to 3, we add randomness to the outputs, which essentially remasks the outputs and provides uniformity.

The circuit differs from the previous one only in the aspects that the shares are increased from 3 to 4 and decreased from 4 to 3, and that the sharing for the inverter itself is different. It takes the same number of clock cycles as the previous one, i.e. 3, but requires 3 bits of randomness for increasing the number of shares from 3 to 4 and then 2 bits of randomness for reducing the shares back from 4 to 3. The argument to not use a uniform sharing of AND gates used in the previous implementation is applicable here too. Hence a total of $3 \times 4 + 2 \times 4 = 20$ bits of randomness is required.

4.4 Side-Channel Analysis Evaluation

First, we describe the circuit that we used for the sequential evaluation of the S-Boxes. All the S-Boxes have separate input ports for the input shares and the randomness, and separate output ports for the output shares. Each S-Box has an enable signal and a reset signal as input. The execution of the S-Box begins when the enable signal is set to high. The values at the ports having the input shares and randomness for the corresponding S-Box, at the time enable goes high, are the ones used as the input to the S-Box. The reset signal is used to reset the S-Box to a known state. Each S-Box has an output done signal which goes high after the execution of the S-Box is complete and the outputs at the corresponding ports of output shares are the results of the execution of the S-Box.

There is an outer wrapper encapsulating the 5 S-Boxes. The wrapper has a control module. The control module of the wrapper has an enable as input signal and a complete signal as an output signal. The enable signal is needed to start the sequence of S-Boxes. The start signal of the first S-Box is set to high on the positive edge of clock following the enable signal going high. When the done signal of the S-Box goes high, the control waits

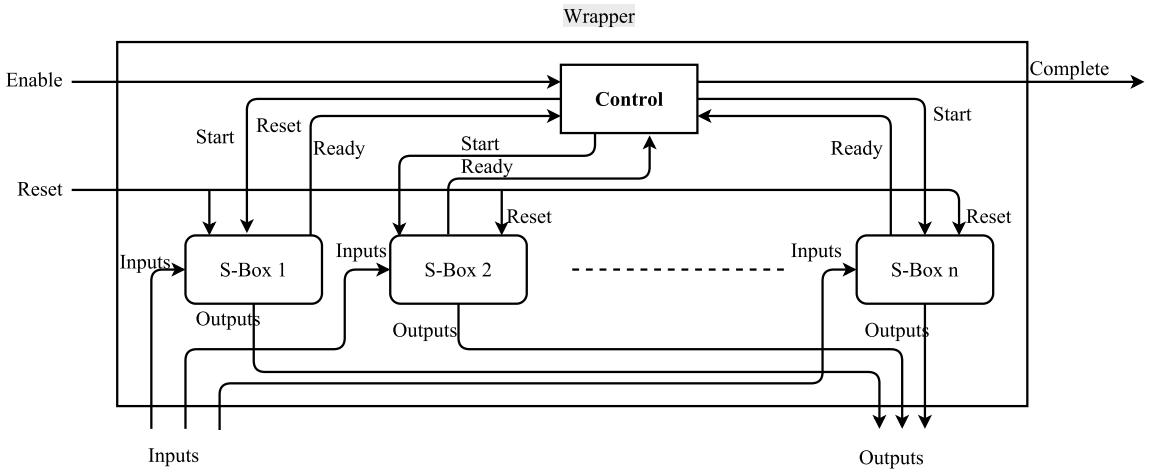


Figure 4.3: Structure of circuit for sequential evaluation of the S-Boxes

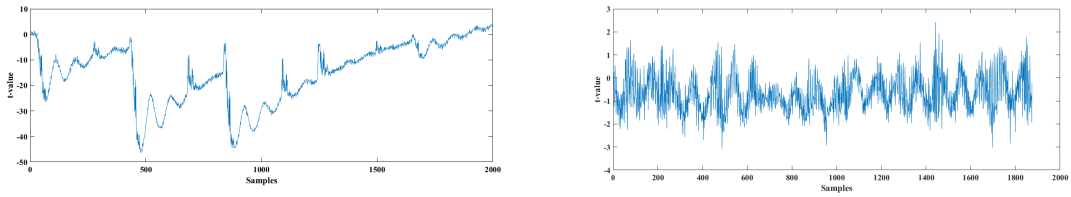
for a few clock cycles before setting the start signal of the next S-Box high. After the done signal of the the last S-Box goes high, the complete signal of the wrapper is set to high. The wrapper has a reset signal as an input which is sent as the reset signal of the S-Boxes when set to high resets all S-Boxes to known states. Figure 4.3 shows the structure of the wrapper.

The design was implemented on a SASEBO-G measurement board using Xilinx ISE 10.1 in order to analyze their leakage characteristics.

The SASEGO-G board has two Xilinx Virtex-II Pro FPGA devices. Our design, was implemented on the crypto FPGA(xc2vp7). In order to prevent optimizations over module boundaries, the "Keep Hierarchy" constraint was kept on while generating the programming file. The control FPGA (xc2vp30) is responsible for the I/O with the measurement PC and generation of random bits. The PRNG which the control FPGA uses to generate the input sharings and random masks for the S-boxes is an AES-128 in OFB mode.

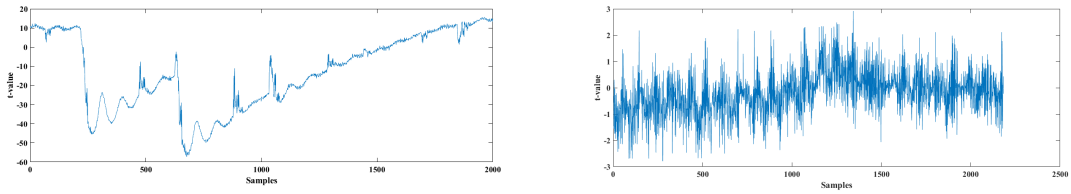
We evaluate the security of our first order secure implementations of the Boyar Peralta AES S-Boxes. We use leakage detection tests [GGJR$^+$11, Koc96, CKN00, CNK04, BCD$^+$13] to test for any power leakage of our masked implementations. The fix class of the leakage detection is chosen as the zero plaintext in all our evaluations.

We follow the standard practice when testing a masked design i.e. first turn off the PRNG to switch off the masking countermeasure. The design is expected to show leakage in this setting, and this serves to confirm that the experimental setup is sound (we can detect leakage). We then proceed by turning on the PRNG. If we do not detect leakage in this setting, the masking countermeasure is deemed to be effective. Figures 4.4, 4.5, 4.6, 4.7, 4.8 show the result of the first order leakage detection tests on the S-Boxes.



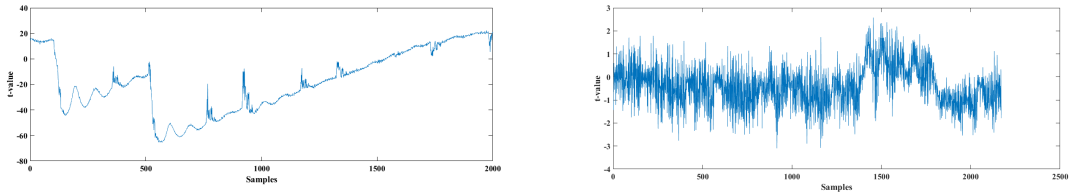
(a) 5K Traces, Masks Off (b) 10M Traces, Masks On

Figure 4.4: First Order leakage detection test for the S-Box with 4 shares.



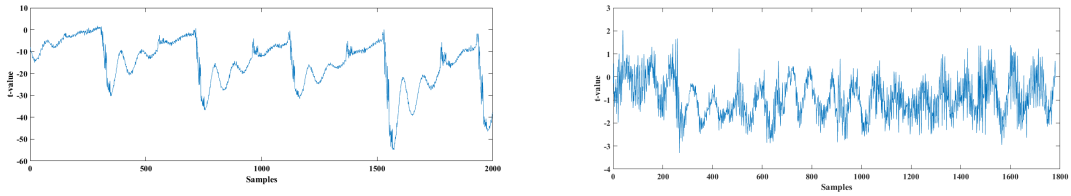
(a) 5K Traces, Masks Off (b) 10M Traces, Masks On

Figure 4.5: First Order leakage detection test for the S-Box with 3 shares, 68 bits of randomness



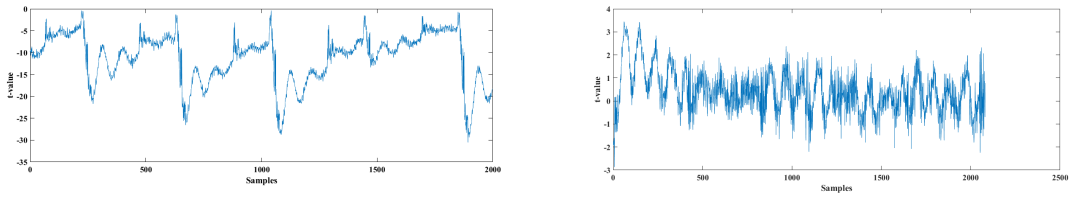
(a) 5K Traces, Masks Off (b) 10M Traces, Masks On

Figure 4.6: First Order leakage detection test for the S-Box with 3 shares,34 bits of randomness



(a) 5K Traces, Masks Off (b) 10M Traces, Masks On

Figure 4.7: First Order leakage detection test for the S-Box with 3 shares and using sharing with $s_{in} = 5$ and $s_{out} = 5$ for a $\mathbb{GF}(2^4)$ inverter



(a) 5K Traces, Masks Off (b) 10M Traces, Masks On

Figure 4.8: First Order leakage detection test for the S-Box with 3 shares and using sharing with $s_{in} = 4$ and $s_{out} = 4$ for a $\mathbb{GF}(2^4)$ inverter

Table 4.1: Area, Randomness and Clock Cycles required per S-box Implementation.

	Area [GEs]	Randomness [bits]	Clock Cycles
Unprotected	269	0	1
$s_{in} = 4$, $s_{out} = 4$	4609	0	4
$s_{in} = 3$, $s_{out} = 3$, 68 random bits, individual AND gates masked	3630	68	4
$s_{in} = 3$, $s_{out} = 3$, 34 random bits, individual AND gates masked	3798	34	4
$s_{in} = 3$, $s_{out} = 3$, inverter masked with $s_{in} = 5$, $s_{out} = 5$	3344	24	3
$s_{in} = 3$, $s_{out} = 3$, inverter masked with $s_{in} = 4$, $s_{out} = 4$	2913	20	3

4.5 Implementation Cost

Here we give a comparison of the area, the required randomness and the number of clock cycles for our implementations. The results of area have been obtained using Synopsys 2013.12 and NanGate 45nm Open Cell Library.

In Table 1, we observe a trade-off between randomness, area and clock cycles. As we reduce the area, the randomness per S-box lookup increases or the number of clock cycles required increase. In our implementation with smallest area, where we share a large algebraic function, the $\mathbb{GF}(2^4)$ inverter as a whole, both the number of clock cycles and the area are reduced.

In Table 2 we compare our implementation with smallest area to some masked implementations based on Canright S-Box.

We can summarize the comparison of our implementations with related implementations as follows:

Table 4.2: Area, Randomness and Clock Cycles required per S-box for related Implementations.

	Area [GEs]	Randomness [bits]	Clock Cycles
$s_{in} = 3$, $s_{out} = 3$, inverter masked with $s_{in} = 4$, $s_{out} = 4$,Boyar Peralta	2914	20	3
$s_{in} = 3$, $s_{out} = 3$,Canright S-Box in [BGN$^+$14b]	3708	44	3
$s_{in} = 3$, $s_{out} = 3$,Canright S-Box in [MPL$^+$11]	4244	48	4
$s_{in} = 3$, $s_{out} = 3$,Canright S-Box in [BGN$^+$15]	2835	32	3
$s_{in} = 2$, $s_{out} = 2$,Canright S-Box in [DCRB$^+$16]	1977	54	6

- We achieve an implementation that consumes no randomness.
- Two of our implementations, which use the sharing for inversion in $\mathbb{GF}(2^4)$, take 3 clock cycles, which is faster than implementations in [MPL$^+$11, DCRB$^+$16]
- Our implementation that uses the 4-sharing of an inverter needs the same number of clock cycles as the smallest one in [BGN$^+$15], while consuming less randomness for an increase in area of only 2.75%.
- The S-Box in [DCRB$^+$16] is the smallest known TI of the AES S-Box. Our implementation is 47% larger in comparison but we obtain a 63% reduction in randomness of and 50% reduction in number of clock cycles required.

4.6 Conclusion

In this chapter, we present the first threshold implementations of the Boyar-Peralta AES S-Box. Since this AES S-Box is of minimum known depth, the critical path might be smaller which would allow clocking the core at higher frequencies making it highly important for secure high-speed and high-throughput applications. We go through an iterated design process, starting from a straightforward approach where we mask each gate individually to arrive at a more efficient implementation by masking the larger algebraic structure of the inversion in $\mathbb{GF}(2^4)$.

Our smallest implementation is 49% larger in area compared to the smallest known threshold implementation of the Canright AES S-box but reduces the randomness by 63%

and number of clock cycles by 50%. Moreover, we achieve a secure implementation of the AES S-Box that requires no randomness at all. The set of secure implementations we present gives the hardware designer more options for tailoring their implementations according to their specifications.

A future direction of research can investigate the result of starting from a masked Canright AES S-Box and using the optimizations mentioned in [BP12] to arrive at a small and secure implementation of the Boyar-Peralta S-Box. Masking the Boyar Peralta S-Box with $d + 1$ shares as shown in [RBN+15] is a possible direction for future work. Another future work would be designing circuits for this S-Box with higher-order security levels, as a determined adversary can still break the first-order masking scheme with a second order attack.

Chapter 5

Lightweight and Side-channel Secure 4×4 S-Boxes from Cellular Automata Rules

This work focuses on side-channel resilient design strategies for symmetric-key cryptographic primitives targeting lightweight applications. In light of NIST's lightweight cryptography project, design choices for block ciphers must consider not only security against traditional cryptanalysis but also side-channel security, while adhering to low area and power requirements. In this chapter, we explore design strategies for substitution-permutation network (SPN)-based block ciphers that make them amenable to low-cost threshold implementations (TI) - a provably secure strategy against side-channel attacks. The core building blocks of our strategy are cryptographically optimal 4×4 S-Boxes, implemented via repeated iterations of simple cellular automata (CA) rules. We present highly optimized TI circuits for such S-Boxes, that consume nearly 40% less area and power as compared to popular lightweight S-Boxes such as PRESENT and GIFT. We validate our claims via implementation results on ASIC using 180nm technology. We also present a comparison of TI circuits for two popular lightweight linear diffusion layer choices - bit permutations and Mix Columns using almost-maximum-distance-separable (MDS) matrices. We finally illustrate design paradigms that combine the aforementioned TI circuits for S-Boxes and diffusion layers to obtain fully side-channel secure SPN block cipher implementations with low area and power requirements.

5.1 Introduction

Lightweight cryptography has received great momentum with the proposal of a number of efficient symmetric-key cryptographic primitives in recent years. Design choices for

lightweight cryptography typically focus on optimizing one or more essential implementation-based criteria, including (but not limited to) area, power, and throughput. At the same time, these primitives must also satisfy the basic security requirements against well-known cryptanalytic attacks such as linear [MY93] and differential [BS91] cryptanalysis. Lightweight block ciphers follow various design principles, amongst which *substitution-permutation network* (SPN) is highly popular. An SPN structure typically comprises of several rounds, where each round has three operational layers - (a) a layer of nonlinear substitution-boxes (S-Boxes), (b) a linear permutation-layer, and (c) round-key-XOR. The impetus for lightweight cryptography has been further enhanced by NIST's recent announcement of a lightweight cryptography project [MBTM17], seeking design choices targeting a variety of devices and applications. In particular, the announcement lists resistance against *side-channel attacks* (SCA) as a principal design criterion. This opens up the need to explore new design strategies for lightweight block ciphers that focus not only on security against traditional cryptanalysis but also side-channel security while adhering to low area and power requirements. The aim of this chapter is to address this issue with respect to the SPN block ciphers. In particular, our proposed strategies focus on protecting the two main components of any SPN block cipher, namely the S-Box layer and the permutation layer. A common protection strategy applied to both layers is the use of TI [NRR06], a provably secure technique against side-channels that has its roots in multi-party computation.

S-Boxes are essential components for any SPN block cipher since they contribute to the protection against traditional cryptanalytic techniques. In order to do so, S-Boxes must fulfill certain cryptographic properties. The minimum set of criteria necessary to consider when designing S-Boxes for SPN designs includes bijectivity, high nonlinearity, and low differential uniformity. Naturally, in various ciphers, S-Boxes are of different sizes, which results in different values of cryptographic properties and can even lead to using S-Boxes with suboptimal properties (see e.g., the Keccak design where the S-Box (χ transformation) is suboptimal with respect to the nonlinearity and differential uniformity properties [BDPA11]).

When considering lightweight cryptography, the situation is simpler. The dominant S-Box size there is 4×4, which does not allow much difference in cryptographic properties, and in fact, ciphers commonly use S-Boxes that are *optimal*. Optimal S-Boxes are those that are bijective, with nonlinearity equal to 4, and differential uniformity equal to 4 [LP07]. Such optimal S-Boxes are found in numerous popular designs like PRESENT [BKL$^+$07b], Prince [BCG$^+$12], Rectangle [ZBL$^+$15], and Midori [BBI$^+$15]. Some recently proposed block ciphers such as GIFT [BPP$^+$17] use cryptographically non-optimal lightweight 4×4 S-Boxes with special properties that allow combining them with bit permutations to achieve optimal diffusion characteristics. The small size of 4×4 S-Boxes has also enabled researchers

to classify all optimal S-Boxes up to the affine equivalence where they show there are 16 optimal non-equivalent classes (commonly denoted G_0 to G_{15}) [LP07]. Existing works have also gone so far as to exhaustively enumerate all 4×4 bijective S-Boxes [Saa12].

Despite the existence of such classifications, it is largely an open problem to propose design strategies for S-Boxes that are low-area, low-power, and at the same-time, amenable to side-channel secure implementations (that is, the corresponding SCA-resistant implementations also optimize area and power as much as possible). One of the foremost techniques for securing S-Box implementations is the use of masking countermeasures [RP10, GPQ11, RBN+15] that are provably secure up to a pre-determined attack order. In more recent times, TI seems to be the preferred choice owing to their enhanced security coverage, particularly against glitch-based SCAs. Thus, our aim is to design cryptographically optimal 4×4 nonlinear functions that support low-area and low-power implementations, while having low-cost side-channel protections in the form of TI circuits.

5.1.1 Overview of Our Contributions and Techniques

The main contributions of this chapter are briefly summarized below:

- **Lightweight and Side-channel Secure Design Strategies for S-Boxes.**

  In this chapter, we use *cellular automata* in order to design such nonlinear functions with inherently lightweight implementations. A cellular automaton is a finite state machine whose state transitions are based on simple local rules. Prior studies have extensively analyzed the scope of realizing complex functions via repeated iterations of this simple rules [Wol83, Wol84b, Wol84a]. A recent work by Picek et al. [PMY+17] explores the possibility of designing cryptographically optimal 4×4 S-Boxes from such simple 4×1 CA-based rules. The idea is to iterate over a single instance of the CA rule, while cyclically shifting the input bits, to obtain one output bit of an S-Box at a time. In this chapter, we take a step further and explore the possibility of designing cryptographically optimal 4×4 S-Boxes from CA rules, while also ensuring that such S-Boxes give rise to side-channel secure TI circuits with low area footprint and power consumption. The main design principle for the TI circuit remains the same - we protect the core CA rule by decomposing the input and output bits into as few shares as possible, and then iterate over this core unit by cyclically permuting the input bits. We demonstrate that a significant proportion of the resulting S-Boxes achieve cryptographically optimal properties, and give rise to distinct classes based on their implementation overheads and amenability to TI implementations. We also demonstrate additional optimizations on the most lightweight of these S-Box classes by exploiting the decomposability of its CA rule into smaller Boolean

functions. Our implementation results on ASIC (180nm technology) show that the most lightweight TI circuit among all CA-based S-boxes has a 49.42% smaller area-footprint and consumes 52.3% less power as compared to the best-known TI of the PRESENT S-Box [PMK+11]. The same TI circuit also consumes 35.36% smaller area-footprint and consumes 44.46% less power as compared to a highly optimized TI of the GIFT S-Box.

- **Lightweight and Side-channel Secure Design Strategies for Permutation Layers.** Permutation layers provide the much-needed diffusion in any block cipher construction, and are hence important for side-channel security. Two main classes of permutation layers dominate nearly all lightweight SPN constructions - bit permutations and almost-maximum-distance-separable (almost-MDS) permutations. Examples of the former include PRESENT [BKL+07b] and GIFT [BPP+17], while an example of the latter strategy is Midori [BBI+15]. In this chapter, we present a comparative analysis of the area and power overheads corresponding to TI implementations for both these choices of permutations. Such a comparative analysis allows a designer to analyze the pros and cons of choosing either of these strategies with respect to a given application.

- **Putting it All Together.** Finally, we present a trade-off analysis between the design choices for the S-Box and permutation layers as components in an overall SPN structure. We first observe that our CA-based S-Boxes have a branch number of 2 (as opposed to 3 for the PRESENT S-Box), and also lacks the bad-output-good-input (BOGI) property exhibited by the GIFT S-Box [BPP+17]. This makes it practically infeasible to combine these S-Boxes with bit-permutation layers in a full SPN structure and necessitates almost-MDS permutation layers. Interestingly, it turns out that the area and power savings from our CA-based S-Boxes outweigh the additional area and power requirements for an almost-MDS permutation layer over a bit permutation layer, particularly when implemented for side-channel security via TI. With these observations, we propose using CA-based S-Boxes in conjunction with almost-MDS mappings as a new design-for-security strategy for designing lightweight block ciphers that are amenable to low-area and low-power TI implementations.

5.1.2 Organization

The rest of this chapter is organized as follows. In Section 5.2, we introduce the notation and present background material on cryptographic properties of S-Boxes, threshold implementations (TI), cellular automata (CA) and its properties, and relevant measurement

units for area footprint and power consumption of CMOS devices. Section 5.3 presents direct-shared TI circuits for cryptographically optimal 4×4 S-Boxes obtained via repeated iterations of local CA rules, along with area and power overheads for the same on ASIC platforms (180nm technology). Section 5.4 further refines these TI circuits by reducing the number of shares to achieve even lower area footprint and power consumption. Section 5.5 compares bit permutations and Mix Columns using almost-MDS matrices in terms of their amenability to low-cost TI implementations. This section also presents design paradigms for combining TI for S-Boxes and diffusion layers to achieve lightweight and fully side-channel secure block cipher implementations. Finally, Section 5.6 summarizes the major findings of the chapter and discusses possible future research directions.

5.2 Preliminaries

5.2.1 Cryptographic Optimality and Representation of S-Boxes

In the standard cryptographic nomenclature, a substitution box (abbreviated as S-Box), is a nonlinear $n \times m$ Boolean function f. Here, we briefly describe some important cryptographic properties of S-boxes.

- ALGEBRAIC DEGREE. To define the algebraic degree of an S-Box, we use the algebraic normal form (ANF) representation of a boolean function f represented by a polynomial in $\mathbb{F}_2[x_0, \ldots, x_{n-1}]/(x_0^2 - x_0, \ldots, x_{n-1}^2 - x_{n-1})$ [Car10a]. The algebraic degree deg_f of a Boolean function f is defined as the number of variables in the largest product term of the function's ANF having a non-zero coefficient [Car10a]. The algebraic degree deg_F of an S-Box F is the maximum algebraic degree of all non-zero linear combinations of the coordinate functions (i.e., component functions) or coordinate functions of F [Car10b]. Ideally, a cryptographically useful S-Box should have high algebraic degree to resist algebraic attacks [MPC]

- BALANCEDNESS. Let F be a function from $\mathbb{F}_2^n$ into $\mathbb{F}_2^m$. We call F to be balanced if it takes every value of $\mathbb{F}_2^m$ same number of times.

- NONLINEARITY. Nonlinearity of an (n, m)-function F equals the minimum nonlinearity of all its component functions $v \cdot F$, where $v \in \mathbb{F}_2^{m*}$ [Nyb93, Car10b]:

$$NL_F = 2^{n-1} - \frac{1}{2} \max_{\substack{a \in \mathbb{F}_2^n \\ v \in \mathbb{F}_2^{m*}}} |W_F(a, v)|,$$

where

$$W_F(a, v) = \sum_{x \in \mathbb{F}_2^m} (-1)^{v \cdot F(x) + a \cdot x}, \ a, v \in \mathbb{F}_2^m,$$

is the Walsh-Hadamard transform [Car10b] of the function F and $a \cdot b$ is the usual inner product of $a, b \in \mathbb{F}_2^n$ that equals $a \cdot b = \bigoplus_{i=1}^{n} a_i b_i$. The nonlinearity of any (n, m) function F is bounded above by the covering radius bound:

$$NL_F \leq 2^{n-1} - 2^{\frac{n}{2}-1}.$$

- DIFFERENTIAL UNIFORMITY. Let F be a function from $\mathbb{F}_2^n$ into $\mathbb{F}_2^m$ with $a \in \mathbb{F}_2^n$ and $b \in \mathbb{F}_2^m$. We define the *difference distribution table* of F with respect to a and b as:

$$D_F(a, b) = \{x \in \mathbb{F}_2^n : F(x) \oplus F(x \oplus a) = b\}.$$

The entry at position (a, b) corresponds to the cardinality of the difference distribution table $D_F(a, b)$ and is denoted as $\delta_F(a, b)$. The *differential uniformity* δ_F is then defined as [Nyb94]:

$$\delta_F = \max_{\substack{a \in \mathbb{F}_2^{n*} \\ b \in \mathbb{F}_2^m}} \delta_f(a, b).$$

- DIFFERENTIAL BRANCH NUMBER. Let F be a function from $\mathbb{F}_2^n$ into $\mathbb{F}_2^m$. We define the differential branch number of F as:

$$BN_F = \min_{x \neq y} \; wt(x \oplus y) + wt(F(x) \oplus F(y)),$$

where $wt(a)$ denotes the Hamming weight of a. Throughout this chapter we use the term branch number to denote differential branch number.

In order to resist linear and differential cryptanalysis attacks, a balanced S-Box should ideally have high nonlinearity and low differential uniformity. In particular, a 4×4 S-Box is said to be *cryptographically optimal* if it is balanced, has nonlinearity equal to 4, and differential uniformity equal to 4 [LP07].

5.2.2 Threshold Implementation: Countermeasure to SCA

The preliminaries of Threshold Implementations has been covered in chapter 3.

5.2.3 Cellular Automata

Cellular Automata (CA) are parallel computational models used in order to simulate and analyze various discrete complex systems. A cellular automaton consists of a regular grid (lattice) of cells. The grid may be in any finite number of dimensions. For each cell, a set of cells called its neighborhood is defined relative to the specified cell. Each cell is in one of a finite number of states. Typically, at every time step all the cells update their states synchronously. The state update is governed by a local rule which is applied to the

neighborhood of every cell.

CA as Vectorial Boolean Function. In this chapter, we restrict ourselves to periodic boundary one dimensional Boolean-cellular automata i.e., the case where every cell is in state 0 or 1 and the lattice is a linear array. A Periodic Boundary CA (PBCA) with n input cells $F : {\mathbb{F}_2}^n \to {\mathbb{F}_2}^n$ is defined for all $x \in {\mathbb{F}_2}^n$ as:

$$F(x_1, x_2, \cdots, x_n) = (f(x_1, \cdots, x_d), \cdots, f(x_{n-d}, \cdots, x_1), \cdots, f(x_n, \cdots, x_{d-1}))$$

where f is a Boolean function on d variables($d \leq n$) is called a local rule. Thus, a CA can be seen as a vectorial Boolean function where each coordinate function f_i corresponds to the local rule f applied to the neighborhood $(x_i, \cdots, x_{i+d-1})$. The vectorial Boolean function F of a CA is also called the CA global rule.

We note that cellular automata based S-Boxes are actually widely used today, since the nonlinear transformation χ in Keccak is actually a PBCA with $n = 5$ cells and local rule f defined as:

$$f(x_1, x_2, x_3) = x_1 \oplus x_2 x_3 \oplus x_3 \ . \tag{5.1}$$

Besides being used in Keccak, the same rule is also used in Panama [DC98], Radio-Gatún [BDPA06], Subterranean [CDGP93], and 3Way [DGV94] ciphers. Unfortunately, despite being a very small rule that can be efficiently implemented, it results in optimal S-Boxes only for dimension 3×3 and is bijective only for odd dimensions. Finally, Picek et al. recently showed that CA-based S-boxes can be very efficient when considering power and area [PMY+17].

5.2.4 Area Overhead and Power Consumption Results

The CMOS technology used for all ASIC implementation results reported in this chapter is 180nm. Each implemented circuit is taken through the RTL-to-GDS2 flow to estimate the area overhead and power consumption. We used Synopsys Design Compiler version I-2013.12-SP5-4 for synthesis and Synopsys IC-Compiler version J-2014.09-SP1 for placement and routing of the design. For simulation, we used Synopsys VCS version I-2014.03-SP1-1. Standard cell library TSL18FS120 from Tower Semiconductor Ltd. is used for physical design. The area overhead for all implemented circuits are measured in terms of gate equivalents (GE), where a GE in our case is equal to the lowest area occupied by a 2-input NAND gate of 1x drive of 180nm technology.

The total power consumption of a CMOS device is given by:

$$P_{total} = P_{static} + P_{dynamic},$$

where P_{static} and $P_{dynamic}$ denote the static and dynamic power consumption of the device. In this chapter, we concentrate on the dynamic power consumption that originates from the switching activity of the circuit:

$$P_{dynamic} = \alpha C V^2 f,$$

where α is the switching factor (the probability of a bit switching from 0 to 1), C is the switched capacitance, V is the voltage, and f is the clock frequency. In our approach, we aim to use a simple structure of CA-based elements, which reduces the area and consequently the capacitance (since capacitance depends on the area). As the capacitance reduces, $P_{dynamic}$ also reduces since the other factors do not increase.

5.3 Lightweight S-Boxes from Cellular Automata Rules

In this section, we illustrate our cellular automata (CA)-based design strategies for obtaining 4×4 S-Boxes that are area and power-efficient, and also amenable to low-cost TI. The idea is to choose a local CA rule, which is essentially a 4×1 Boolean function, such that it has a low-cost equivalent implementation in hardware. The 4×4 S-Box mapping is obtained by applying the same CA rule to four different (cyclic) permutations of the input bits. This allows for an iterative implementation in hardware, with the CA rule implemented once in the data-path, and the control unit applying a cyclically shifted variant of the input bits in each clock cycle to obtain the corresponding output bit. We first describe the De Bruijn graph-based technique to choose the local CA rule and subsequently enumerate certain cryptographically optimal S-Boxes obtained via the aforementioned procedure. We also classify these S-Boxes in terms of their amenability to low-area and low-power TI and present optimized TI implementations for representatives from each class.

5.3.1 Choosing the CA Rule

Given a 4×1 CA rule f, the corresponding 4×4 S-Box is given by:

$$S(X, Y, Z, W) = (f(X, Y, Z, W), f(Y, Z, W, X), f(Z, W, X, Y), f(W, X, Y, Z))$$

We focus on choosing such CA rules that ensure that the corresponding S-box is bijective. The test for injectivity of the global map of a one-dimensional CA was shown to be decidable in [AP72], while the test for surjectivity for the same was shown to have a quadratic-time algorithm in [Sut91], using De Bruijn graphs. These graphs provide a convenient way to describe configurations of linear CAs. We follow these principles to identify local 4×1 CA rules, which in turn guarantee that the resultant 4×4 S-Box is bijective. The detailed technique for choosing such a CA rule is as follows.

5.3.1.1 De Bruijn Graph Representation.

For any CA with an n-variable local rule $f : {\mathbb{F}_2}^n \to \mathbb{F}_2$, the associated De Bruijn graph is a directed graph $G = (V, E)$, where every vertex $v \in V$ is labeled with an $(n-1)$-bit string. There exists an edge e from vertex v_1 to vertex v_2 if the first $(n-2)$ bits of the label of v_2 are the same as the last $(n-2)$ bits of the label of v_1. For example, the De Bruijn graph with $n = 4$ has an edge from $v_1 = 010$ to $v_2 = 100$ as the first two bits of v_2 is 10 which is same as the last 2 bits of v_1. Quite evidently, $|V| = 2^{n-1}$, and $|E| = 2 \cdot 2^{n-1} = 2^n$ (observe that each vertex has exactly two incoming and two outgoing edges).

5.3.1.2 Generating Optimal 4×4 S-Boxes from De Bruijn Graphs.

Given a De-Bruijn graph $G = (V, E)$ with $|V| = 2^{n-1}$, a CA local rule may be derived by associating each edge of this graph with a bit $b \in \{0, 1\}$. Since there are 2^n edges, the total number of possible CA rules that can be associated with this graph is 2^{2^n}. In particular, for $n = 4$, the total number of such CA rules is $2^{2^4} = 2^{16}$. Each such rule gives rise to a unique surjective 4×4 function. An exhaustive search of these functions yields 1 536 bijective functions, which are our candidate S-Boxes. Finally, we test these functions for cryptographic optimality (in terms of their nonlinearity and differential uniformity), which narrows down our search space to 512 candidate S-Boxes, which may be further sub-classified into four affine-equivalent classes - namely, G_3, G_4, G_5, and G_6. Details of these S-Boxes have been reported previously in [MPLJ17].

5.3.2 Classification of Cryptographically Optimal CA-based 4×4 S-Boxes

Our next step is to classify the 512 cryptographically optimal CA-based 4×4 S-Boxes into certain classes, such that each category comprises of S-Boxes that are expected to have similar area and power overhead in hardware, as well as similar TI circuit representations. As it turns out, each of these quantities are closely related to the nature of the algebraic normal form (ANF) representation of the S-Boxes. Given that each S-Box under consideration has optimal algebraic degree 3, we make the following essential observations:

- S-Boxes with the same number of cubic, quadratic, and linear terms in their ANF form have similar area footprint and expected power consumption in hardware.
- S-Boxes with the same number of cubic, quadratic, and linear terms in their ANF form have nearly identical TI circuits owing to their nearly identical algebraic structure.

Based on this rationale, we classify the S-Boxes depending on the number of linear, quadratic, and cubic terms present in the ANF of the S-Box. According to this classification, we have

obtained 12 S-Box classes as shown in Table 5.1. We also list the CA rules corresponding to representative optimal S-Boxes for each class. Note that class (a, b, c) comprises of optimal S-Boxes with a cubic terms, b quadratic terms, and c linear terms, respectively. We also summarize the cryptographic properties of these representative S-Boxes in Table 5.2, and compare them with the cryptographic properties of the PRESENT and GIFT S-Boxes.

Table 5.1: Grouping S-Boxes into classes by ANF properties

S-Box Class	**Representative CA Rule**
(1,2,2)	$f(X,Y,Z,W) = XZW \oplus XY \oplus YW \oplus Y \oplus Z$
(1,3,1)	$f(X,Y,Z,W) = YZW \oplus XZ \oplus YZ \oplus YW \oplus X$
(1,3,3)	$f(X,Y,Z,W) = YZW \oplus XY \oplus XZ \oplus YW \oplus Y \oplus Z \oplus W$
(1,4,2)	$f(X,Y,Z,W) = YZW \oplus XY \oplus XZ \oplus XW \oplus ZW \oplus X \oplus W$
(1,5,1)	$f(X,Y,Z,W) = XYW \oplus XY \oplus XZ \oplus XW \oplus YW \oplus ZW \oplus Z$
(1,5,3)	$f(X,Y,Z,W) = XYW \oplus XY \oplus XZ \oplus XW \oplus YZ \oplus YW \oplus Y \oplus Z \oplus W$
(3,2,2)	$f(X,Y,Z,W) = XYZ \oplus XZW \oplus YZW \oplus XZ \oplus YZ \oplus X \oplus Y$
(3,3,1)	$f(X,Y,Z,W) = XYZ \oplus XZW \oplus YZW \oplus XZ \oplus XW \oplus YW \oplus Z$
(3,3,3)	$f(X,Y,Z,W) = XYW \oplus XZW \oplus YZW \oplus XY \oplus XZ \oplus YW \oplus X \oplus Z \oplus W$
(3,4,2)	$f(X,Y,Z,W) = XYZ \oplus XYW \oplus XZW \oplus XY \oplus XZ \oplus XW \oplus YZ \oplus Z \oplus W$
(3,5,1)	$f(X,Y,Z,W) = XYZ \oplus XYW \oplus YZW \oplus XZ \oplus XW \oplus YZ \oplus YW \oplus ZW \oplus Y$
(3,5,3)	$f(X,Y,Z,W) = XYZ \oplus XYW \oplus XZW \oplus XY \oplus XZ \oplus YZ \oplus YW \oplus ZW \oplus X \oplus Y \oplus W$

Table 5.2: Cryptographic properties of the considered S-boxes

S-Box	**Nonlinearity**	**Differential Uniformity**	**Balancedness**	**Algebraic Degree**	**Branch Number**
CA-based (Optimal)	4	4	Yes	3	2
PRESENT (Optimal)	4	4	Yes	3	3
GIFT (Non-Optimal)	4	6	Yes	3	2

5.3.3 Threshold Implementations of CA-based S-Boxes

We now describe direct sharing-based TI circuits for the aforementioned classes of CA-based S-boxes and compare their relative area overheads and power consumption results. Since each of the representative S-Boxes listed above has algebraic degree equal to 3, we adopt the direct 4-to-4 non-complete sharing method for cubic functions originally proposed in [Bil15] to obtain the corresponding TI circuits for each of the corresponding CA rules. We explicitly depict two of the most area-efficient and low-power TI circuits below. These correspond to the representative CA-rules for the S-Box classes $(1, 2, 2)$ and $(1, 3, 1)$ respectively. Note that $\{X_j, Y_j, Z_j, W_j\}_{j\in[1,4]}$ denote the shares for the input bits X, Y, Z and W, respectively, while $\{f_j\}_{j\in[1,4]}$ denotes the shares for the output f of the CA rule.

Class:(1,2,2) , CA-Rule: $f = XZW \oplus YW \oplus XY \oplus Y \oplus Z$

$$
\begin{aligned}
f_1 &= (X_1Z_2W_3) \oplus (X_1Z_3W_2) \oplus (X_2Z_1W_3) \oplus (X_2Z_3W_1) \oplus (X_3Z_1W_2) \oplus (X_3Z_2W_1) \oplus Y_1 \oplus Z_1 \\
f_2 &= ((X_2 \oplus X_3 \oplus X_4)(Z_2 \oplus Z_3 \oplus Z_4)(W_2 \oplus W_3 \oplus W_4)) \oplus ((X_2 \oplus X_3 \oplus X_4)(Y_2 \oplus Y_3 \oplus Y_4)) \\
&\quad \oplus ((Y_2 \oplus Y_3 \oplus Y_4)(W_2 \oplus W_3 \oplus W_4)) \oplus Y_2 \oplus Z_2 \\
f_3 &= (X_1(Z_3 \oplus Z_4)(W_3 \oplus W_4)) \oplus (Z_1(X_3 \oplus X_4)(W_3 \oplus W_4)) \oplus (W_1(X_3 \oplus X_4)(Z_3 \oplus Z_4)) \\
&\quad \oplus (X_1Z_1(W_3 \oplus W_4)) \oplus (X_1W_1(Z_3 \oplus Z_4)) \oplus (Z_1W_1(X_3 \oplus X_4)) \oplus (X_1Z_1W_1) \\
&\quad \oplus (X_1(Y_3 \oplus Y_4)) \oplus (Y_1(X_3 \oplus X_4)) \oplus (X_1Y_1) \oplus (Y_1(W_3 \oplus W_4)) \oplus (W_1(Y_3 \oplus Y_4)) \\
&\quad \oplus (Y_1W_1) \oplus Y_3 \oplus Z_3 \\
f_4 &= (X_1Z_1W_2) \oplus (X_1Z_2W_1) \oplus (X_2Z_1W_1) \oplus (X_1Z_2W_2) \oplus (X_2Z_1W_2) \oplus (X_2Z_2W_1) \\
&\quad \oplus (X_1Z_2W_4) \oplus (X_2Z_1W_4) \oplus (X_1Z_4W_2) \oplus (X_2Z_4W_1) \oplus (X_4Z_1W_2) \oplus (X_4Z_2W_1) \\
&\quad \oplus (X_1Y_2) \oplus (Y_1X_2) \oplus (Y_1W_2) \oplus (W_1Y_2) \oplus Y_4 \oplus Z_4
\end{aligned}
$$

Class:(1,3,1) , CA-Rule: $f = YZW \oplus YW \oplus YZ \oplus XZ \oplus X$

$$
\begin{aligned}
f_1 &= (Y_1Z_2W_3) \oplus (Y_1Z_3W_2) \oplus (Y_2Z_1W_3) \oplus (Y_2Z_3W_1) \oplus (Y_3Z_1W_2) \oplus (Y_3Z_2W_1) \oplus X_1 \\
f_2 &= ((Y_2 \oplus Y_3 \oplus Y_4)(Z_2 \oplus Z_3 \oplus Z_4)(W_2 \oplus W_3 \oplus W_4)) \oplus ((X_2 \oplus X_3 \oplus X_4)(Z_2 \oplus Z_3 \oplus Z_4)) \\
&\quad \oplus ((Y_2 \oplus Y_3 \oplus Y_4)(Z_2 \oplus Z_3 \oplus Z_4)) \oplus ((Y_2 \oplus Y_3 \oplus Y_4)(W_2 \oplus W_3 \oplus W_4)) \oplus X_2 \\
f_3 &= (Y_1(Z_3 \oplus Z_4)(W_3 \oplus W_4)) \oplus (Z_1(Y_3 \oplus Y_4)(W_3 \oplus W_4)) \oplus (W_1(Y_3 \oplus Y_4)(Z_3 \oplus Z_4)) \\
&\quad \oplus (Y_1Z_1(W_3 \oplus W_4)) \oplus (Y_1W_1(Z_3 \oplus Z_4)) \oplus (Z_1W_1(Y_3 \oplus Y_4)) \oplus (Y_1Z_1W_1) \\
&\quad \oplus (X_1(Z_3 \oplus Z_4)) \oplus (Z_1(X_3 \oplus X_4)) \oplus (X_1Z_1) \oplus (Y_1(Z_3 \oplus Z_4)) \oplus (Z_1(Y_3 \oplus Y_4)) \oplus \\
&\quad (Y_1Z_1) \oplus (Y_1(W_3 \oplus W_4)) \oplus (W_1(Y_3 \oplus Y_4)) \oplus (Y_1W_1) \oplus X_3 \\
f_4 &= (Y_1Z_1W_2) \oplus (Y_1Z_2W_1) \oplus (Y_2Z_1W_1) \oplus (Y_1Z_2W_2) \oplus (Y_2Z_1W_2) \oplus (Y_2Z_2W_1) \oplus (Y_1Z_2W_4) \\
&\quad \oplus (Y_2Z_1W_4) \oplus (Y_1Z_4W_2) \oplus (Y_2Z_4W_1) \oplus (Y_4Z_1W_2) \oplus (Y_4Z_2W_1) \oplus (X_1Z_2) \oplus (Z_1X_2) \\
&\quad \oplus (Y_1Z_2) \oplus (Z_1Y_2) \oplus (Y_1W_2) \oplus (W_1Y_2) \oplus X_4
\end{aligned}
$$

Figure 5.1 illustrates the hardware architecture for the direct-sharing based TI circuit corresponding to a given CA rule. The main components of the architecture are the shift registers (cyclic) for the shares corresponding to the input variables, the core block implementing the TI circuit for the CA rule, and the demultiplexer gates that are used to output one bit per clock cycle. Note that the counter bits are dependent only on the clock signal; in particular, they are independent of the other intermediate share values, and hence need not themselves

be shared. A comparison of the area and power consumption for the direct sharing-based TI circuits for all representative S-Boxes is depicted in Table 5.3. The following trend is evident from the hardware implementation results:

Observation 1. *TI of an S-Box of class* (a_1, b_1, c_1) *has lower area and power consumption than an S-Box of class* (a_2, b_2, c_2)*, if:*

- $a_1 < a_2$,
- $a_1 = a_2$ and $(b_1 + c_1) < (b_2 + c_2)$.

Contrary, in the case where $a_1 = a_2$ and $(b_1 + c_1) = (b_2 + c_2)$, there is no such obvious trend. This could be attributed to certain optimizations made by the design compiler during synthesis.

Table 5.3: TI of CA-based S-Box representatives: area and power consumption (ASIC Technology: 180nm)

S-Box		**Area (GE)**	**Dynamic Power (μW)**
	Class		
CA-Based	(1,2,2)	265.03	232.51
	(1,3,1)	259.23	222.36
	(1,3,3)	276.06	247.78
	(1,4,2)	288.35	254.89
	(1,5,1)	276.55	244.97
	(1,5,3)	298.7	284.19
	(3,2,2)	378.98	349.76
	(3,3,1)	393.83	357.6
	(3,3,3)	415.21	398.51
	(3,4,2)	405.57	381.00
	(3,5,1)	397.10	381.46
	(3,5,3)	418.16	413.14
PRESENT		450.54	490.18
GIFT		303.81	380.44

Comparison with Direct-Shared TI for PRESENT and GIFT S-Box. Note that the first six CA-based S-Box representatives (for classes $(1, 2, 2)$ through $(1, 5, 3)$) in Table 5.3 have TI circuits with lower area footprint as compared to both the PRESENT and GIFT S-Boxes. Additionally, the power consumption for nearly all CA-based TI circuits is significantly lower.

Note that in the direct-shared TI, each input and output variable is four-shared, which leads to a significant area overhead. It is possible to minimize the area overheads of these circuits even further by reducing the number of shares in each case. This is achieved by a technique referred to as *composite TI*, which we describe in the next section.

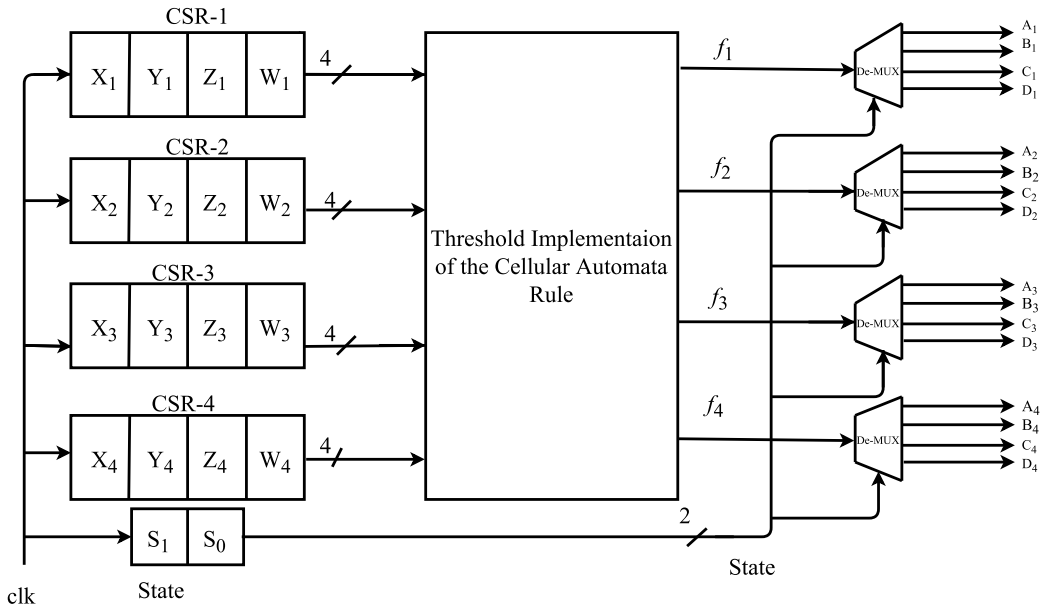


Figure 5.1: Architecture for TI circuits corresponding to CA-based S-Boxes

5.4 Composite TI: Optimizing TI Circuits for Low Area and Power

In this section, we present *composite TI* - a generic technique that allows for highly optimized TI implementations of CA rules, in comparison to direct sharing techniques. A similar technique has been used in [PMK+11] to obtain a highly optimized TI for the PRESENT S-Box. The idea is to express each 4×1 CA rule of algebraic degree 3 as a composition of Boolean sub-functions of degree 2 each. We then proceed by identifying uniform and non-complete sharing for these degree two sub-functions and subsequently cascading them. In order to maintain non-completeness, the cascading must ensure that the TI circuits for the two sub-functions are separated by using registers. This can be illustrated using the following instance. Suppose that a CA-rule $f(X)$ is expressed as a composition of two sub-rules $g(A)$ and $h(X)$, where A denotes the intermediate output of $h(X)$. Now, consider a uniform first-order 3-sharing of h, denoted as $A_1 = h_1(X_1, X_2)$ and $A_2 = h_2(X_2, X_3)$, that

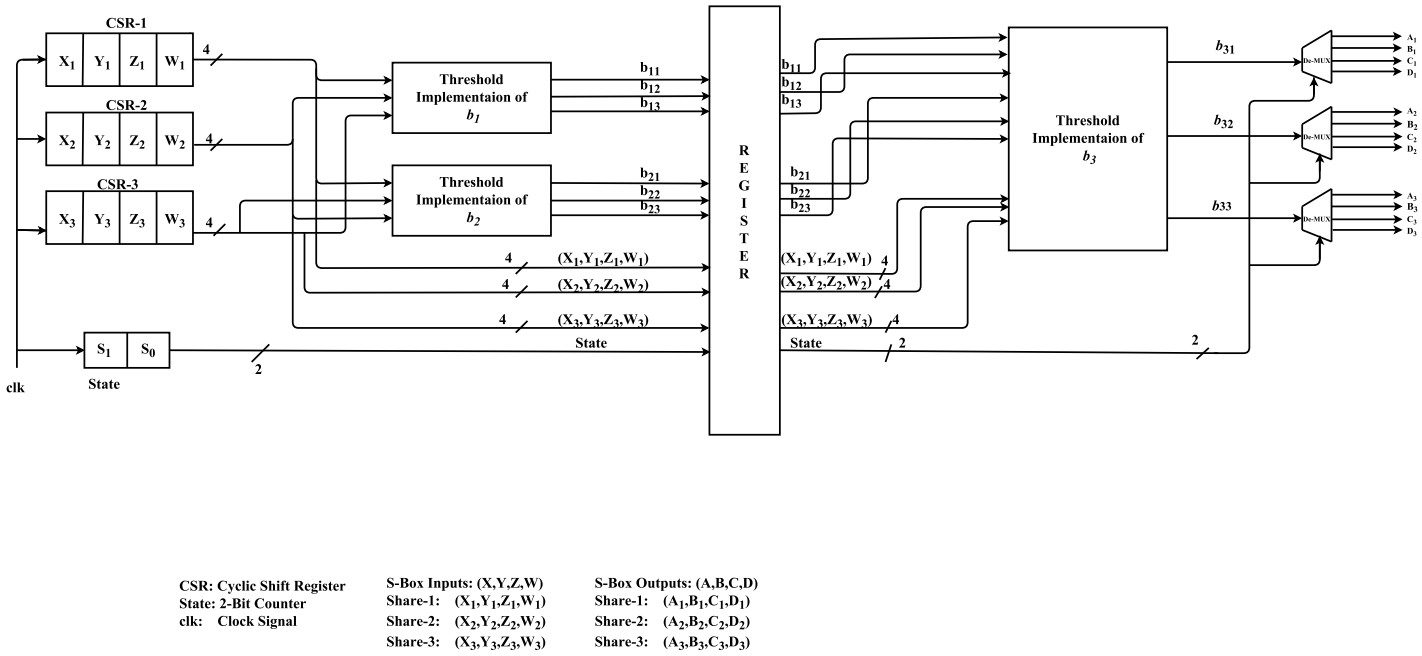


Figure 5.2: Architecture for TI circuits corresponding to CA-based S-Boxes

are fed subsequently to the sharing of g. Note that the share function $g_1(A_1, A_2)$ can also be written as $g_1(X_1, X_2, X_3)$, in which case, a glitch in this function produces a leakage dependent on all the shares of X. This is avoided by partitioning the nonlinear operations with a register that disallows the propagation of a glitch affecting all the shares of an unmasked value. We illustrate the decomposition strategy for the representative S-Boxes of the classes $(1, 2, 2)$ and $(1, 3, 1)$, which are the most area and power-efficient among all the S-Box classes (see Table 5.3).

5.4.1 Decomposition for CA-based S-Box Class $(1, 2, 2)$

We begin by illustrating a decomposition of the representative CA-rule for the S-Box class $(1, 2, 2)$. While the original rule f has algebraic degree 3, each of the decomposed functions b_1, b_2 and b_3 have degree 2.

$$\boldsymbol{f = XZW \oplus YW \oplus XY \oplus Y \oplus Z}$$

$$b_1(X, Y, W) = X \oplus Y \oplus XW \oplus YW$$
$$b_2(X, Y, Z) = Z \oplus XY \oplus XZ$$
$$b_3(X, Z, W) = X \oplus W \oplus XZ \oplus ZW$$
$$f(X, Y, Z, W) = b_1 \oplus b_2 \oplus b_1b_3 \oplus b_2b_3 = b_1(b_1, b_2, b_3)$$

The next step is to obtain a uniform three-sharing for the decomposed functions b_1, b_2 and b_3. We first present a nomenclature of the shares for the various input variables and decomposed functions.

$$\begin{aligned} b_1 &= b_{11} \oplus b_{12} \oplus b_{13} \\ b_2 &= b_{21} \oplus b_{22} \oplus b_{23} \\ b_3 &= b_{31} \oplus b_{32} \oplus b_{33} \\ X &= X_1 \oplus X_2 \oplus X_3 \\ Y &= Y_1 \oplus Y_2 \oplus Y_3 \\ Z &= Z_1 \oplus Z_2 \oplus Z_3 \\ W &= W_1 \oplus W_2 \oplus W_3 \end{aligned}$$

The three-shared TI circuit is now illustrated below:

$$\begin{aligned} b_{11} &= X_1 \oplus Y_2 \oplus (Y_1W_1) \oplus (Y_1W_2) \oplus (Y_2W_1) \oplus (X_1W_1) \oplus (X_1W_2) \oplus (X_2W_1) \\ b_{12} &= X_2 \oplus Y_3 \oplus (Y_2W_2) \oplus (Y_2W_3) \oplus (Y_3W_2) \oplus (X_2W_2) \oplus (X_2W_3) \oplus (X_3W_2) \\ b_{13} &= X_3 \oplus Y_1 \oplus (Y_3W_3) \oplus (Y_3W_1) \oplus (Y_1W_3) \oplus (X_3W_3) \oplus (X_3W_1) \oplus (X_1W_3) \\ b_{21} &= Z_1 \oplus (Z_1X_2) \oplus (Z_2X_1) \oplus (Y_1X_2) \oplus (Y_2X_1) \oplus (Z_1X_1) \oplus (Y_1X_1) \\ b_{22} &= Z_2 \oplus (Z_2X_3) \oplus (Z_3X_2) \oplus (Y_2X_3) \oplus (Y_3X_2) \oplus (Z_2X_2) \oplus (Y_2X_2) \\ b_{23} &= Z_3 \oplus (Z_1X_3) \oplus (Z_3X_1) \oplus (Y_1X_3) \oplus (Y_3X_1) \oplus (Y_3X_3) \oplus (Z_3X_3) \\ b_{31} &= X_1 \oplus W_2 \oplus (Z_1W_1) \oplus (Z_1W_2) \oplus (Z_2W_1) \oplus (X_1Z_1) \oplus (X_1Z_2) \oplus (X_2Z_1) \\ b_{32} &= X_2 \oplus W_3 \oplus (Z_2W_2) \oplus (Z_2W_3) \oplus (Z_3W_2) \oplus (X_2Z_2) \oplus (X_2Z_3) \oplus (X_3Z_2) \\ b_{33} &= X_3 \oplus W_1 \oplus (Z_3W_3) \oplus (Z_3W_1) \oplus (Z_1W_3) \oplus (X_3Z_3) \oplus (X_3Z_1) \oplus (X_1Z_3) \end{aligned}$$

5.4.2 Decomposition for CA-based S-Box Class $(1, 3, 1)$

We now illustrate a decomposition of the representative CA-rule for the S-Box class $(1, 3, 1)$. Once again, while the original rule f has algebraic degree 3, each of the decomposed functions b_1, b_2 and b_3 have degree 2.

$$\begin{aligned} \boldsymbol{f = YZW \oplus XZ \oplus YW \oplus YZ \oplus X} \\ \hline \\ b_1(X, Y, W) &= X \oplus YW \\ b_2(X, Y, Z) &= YZ \oplus YW \\ f(X, Y, Z, W) &= b_2 \oplus b_1W \oplus X = b_3(b_1, b_2, X, W) \end{aligned}$$

We now present a uniform three-sharing for the decomposed functions b_1, b_2 and b_3. The nomenclature of the shares for the various input variables and decomposed functions is the

Table 5.4: Hardware overhead of CA-based S-Box representatives(ASIC Technology: 180nm)

S-Box		**Area (GE)**	**Dynamic Power (μW)**
CA-Based	**Class**		
	(1,2,2)	212.61	170.2
	(1,3,1)	140.62	113.3
PRESENT		278.00	237.4
GIFT		217.57	207.75

same as described above.

$$
\begin{aligned}
b_{11} &= X_1 \oplus (Y_1W_2) \oplus (W_1Y_2) \\
b_{12} &= X_2 \oplus (W_2W_3) \oplus (Y_2Y_3) \\
b_{13} &= X_3 \oplus (Y_3W_1) \oplus (Y_3W_1) \oplus (Y_1W_3) \oplus (Y_3W_1) \\
b_{21} &= (Z_1Y_2) \oplus (Z_2Y_1) \oplus (W_1Y_2) \oplus (W_2Y_1) \oplus (Z_1Y_1) \oplus (W_1Y_1) \oplus (Z_1W_1) \oplus (Z_2W_2) \\
b_{22} &= (Z_2Y_3) \oplus (Z_3Y_2) \oplus (W_2Y_3) \oplus (W_3Y_2) \oplus (Z_2Y_2) \oplus (W_2Y_2) \oplus (Z_2W_2) \oplus (Z_3W_3) \\
b_{23} &= (Z_1Y_3) \oplus (Z_3Y_1) \oplus (W_1Y_3) \oplus (W_3Y_1) \oplus (W_3Y_3) \oplus (Z_3Y_3) \oplus (Z_3W_3) \oplus (Z_1W_1) \\
b_{31} &= (Z_1b_{12}) \oplus (b_{11}Z_2) \oplus (b_{21}) \oplus (X_1) \\
b_{32} &= ((Z_2 \oplus Z_3)(b_{12} \oplus (b_{13}))) \oplus (b_{22}) \oplus (X_2) \\
b_{33} &= (Z_1b_{13}) \oplus (b_{11}Z_3) \oplus (b_{13}b_{11}) \oplus (b_{23}) \oplus (X_3)
\end{aligned}
$$

5.4.3 Hardware Results for Composite TI of CA-based S-Boxes

In this section, we compare the area and power requirements of the composite TI circuits described above. We also compare these results with composite TI for the PRESENT and GIFT S-Boxes. The architecture for the composite TI circuit is illustrated in Figure 5.2. As mentioned before, the counter bits need not be shared as they are independent of all other intermediate share values. For the PRESENT S-Box, we implement the same composite TI circuit reported in [PMK$^+$11], while for the GIFT S-Box we present new results for composite TI that have not been reported in existing literature. The comparison reveals that the smallest composite TI circuit among CA-based S-boxes has a 49.42% smaller area-footprint and consumes 52.3% less power as compared to the best-known composite TI of the PRESENT S-Box. The same S-Box also consumes 35.36% smaller area-footprint and consumes 44.46% less power as compared to our highly optimized composite TI of the GIFT S-Box.

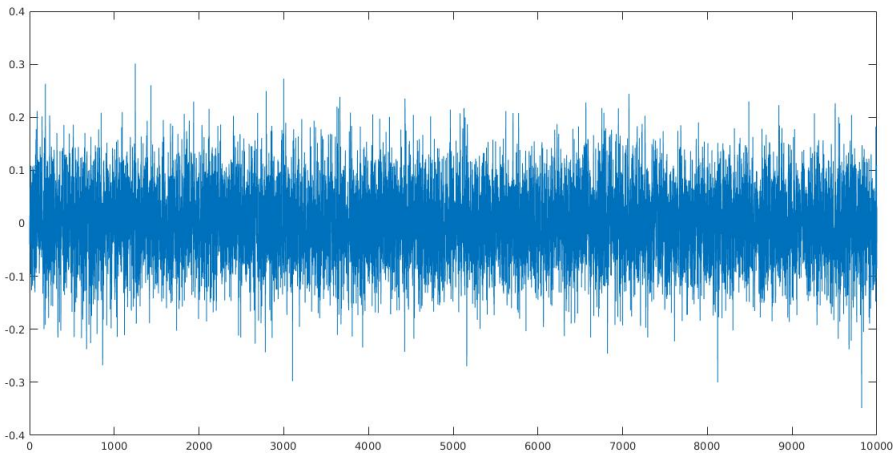


Figure 5.3: TVLA of Composite-TI circuit for CA-Based S-Box representing class $(1, 3, 3)$

5.4.4 Side-channel Leakage Resistance Evaluation using TVLA

We conclude this section by presenting a side-channel evaluation of the best TI circuit among all CA-based S-Boxes, corresponding to the representative CA rule for the class $(1, 3, 3)$. The evaluation was performed by implementing the TI circuit on a Virtex-5 FPGA on a SASEBO-GII board. The programming file for our design was generated using Xilinx ISE 14.7; the "Keep Hierarchy" constraint was kept on while generating the programming file in order to prevent optimizations over module boundaries. We collected 1 000 000 power trace samples from the target FPGA device, and performed a fixed-versus-random statistical test vector leakage assessment (TVLA) test on these collected traces. The fixed class for the test was chosen as the all-zero input in all our evaluations. Figure 5.3 demonstrates the result of the TVLA analysis on the power traces. The outcome of the statistical test consists of values in the range $(-0.4, 0.3)$, which is well within the permissible range of $(-4.5.4.5)$ [SM15].

5.5 Area and Power Efficient Threshold Implementations for SPN Block Ciphers

In this section, we provide a brief discussion on lightweight TI implementations for the other major component of an SPN block cipher, namely, the linear diffusion layer. We then discuss how our CA-based S-Boxes may be combined with such diffusion layers to achieve lightweight TI circuits for full block ciphers.

5.5.1 Lightweight TI circuits for Linear Diffusion Layers

Popular diffusion layer choices in SPN block ciphers include bit-permutation (as in PRESENT and GIFT), Mix Columns using MDS matrices (as in AES [DR00]), and Mix Columns using almost-MDS matrices (as in Midori [BBI+15]). Of these, MDS matrices are typically avoided in ciphers targeting lightweight applications owing to their high area footprints and power requirements. Bit permutations are obviously the most efficient choice for hardware implementations since they have minimal area footprint and power consumption. However, bit-permutation based block ciphers require a greater number of rounds to achieve security against standard cryptanalytic attacks. Almost-MDS matrices constitute a somewhat intermediate alternative, in the sense that they lead to slightly more expensive implementations, but provide better throughput by reducing the number of required rounds. In this section, we compare the area and power requirements for TI circuits of bit permutations and almost-MDS matrices:

TI for Bit Permutations. As a bit permutation is essentially simple wiring of bits, and doesn't require any mathematical operations, there is no extra overhead for TI of bit permutation. Note that, permutation layers like *Shift-Row* (used in AES) or *Shuffle-Cell* (used in Midori) are essentially bit permutations, and hence no additional overhead is required during TI design of these operations (see Table 5.5).

Mix Columns using Almost-MDS. Another lightweight choice for obtaining diffusion is mix column operation using almost-MDS matrices. Following is the most lightweight 4×4 almost-MDS matrix:

$$\begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

This matrix is used in popular block cipher Midori. We implemented a TI circuit for multiplying a 4×1 state vector with the aforementioned matrix. Note that a straightforward TI circuit must protect 8 XOR gates (two per row of the matrix). In our implementation, we reduce the overhead to 7 XOR gates as follows: we first compute the XOR of all input vector elements (this requires 3 XORs), and then XOR one element per row to obtain the desired output. The area and power requirements for the same are reported in Table 5.5.

Table 5.5: TI circuits for diffusion layer choices (ASIC Technology: 180nm)

Diffusion Layer	**Area (GE)**	**Dynamic Power (mW)**
Bit Permutation	3.15	0
Almost-MDS	213.47	1.47

5.5.2 Putting it all Together

In this section, we propose two design paradigms for combining the CA-based optimal S-Boxes with the aforementioned diffusion layer choices to achieve SPN block ciphers with low-area and low-power TI circuits. The first of these paradigms focuses only on optimizing area and power of the TI circuit, without caring for the throughput. In the second design paradigm, we also incorporate the throughput as an additional performance criterion for the TI circuit.

Design Paradigm-1: Focus on Area and Power Only. In this design paradigm, we adopt the SPN block cipher structure of GIFT (which is conceptually identical to that of PRESENT), in the sense that a layer of n 4×4 S-Boxes (typically, $n = 16$) is followed by a bit permutation layer. The S-Box is chosen to be one of the two CA-based S-Boxes (corresponding to classes $(1, 2, 2)$ and $(1, 3, 1)$), or is the original GIFT/PRESENT S-Box. Note that both the CA-based S-Boxes (i) have branch number equal to 2 and (ii) do not possess the BOGI (Bad Output Good Input) property[1] defined in [BPP+17]. This observation essentially tells us that, to sustain against linear and differential cryptanalysis, the number of rounds required for an SPN block cipher using our CA-based S-Box with bit permutations would be considerably higher than an equivalent cipher using the GIFT/PRESENT S-Box with bit permutations. However, since this design paradigm targets area and power efficiency, without any restrictions on throughput, our CA-based S-Boxes outperform the GIFT/PRESENT S-Boxes.

Design Paradigm-2: Focus on Area and Power with reasonable Throughput. In this design paradigm, we adopt an SPN block cipher structure with the following design choices:

- We use standard bit permutations in conjunction with the S-Boxes of PRESENT/GIFT.
- We use a standard bit permutation *followed by a Mix Columns operation using an almost-MDS matrix* in conjunction with our CA-based S-Boxes.

Note that use of Mix Columns operation with an almost-MDS matrix achieves significant diffusion in each round, ensuring a significant reduction in the number of rounds (and hence, an improved throughput) as compared to the previous design paradigm.

Table 5.6 summarizes the implementation overheads for TI circuits following both the design paradigms. We note that in both cases, the TI circuit using the CA-based S-Box representing class $(1, 3, 1)$ outperforms the TI circuits using the PRESENT and GIFT S-Boxes. Interestingly, even when the CA-based S-Box is used in conjunction with the almost-MDS

[1] In fact, all the 4×4 CA-based optimal S-Boxes have branch number 2 and do not possess BOGI property

matrix, the area and power savings from the choice of S-Box makes up for the additional overhead due to the linear layer.

Table 5.6: Lightweight TI for SPN block cipher: area and power (ASIC Technology: 180nm)

S-Box		**Diffusion Layer**	**Area (GE)**			**Power (mW)**
	Class		**16 S-Boxes**	**Diffusion Layer**	**Total**	
CA-Based	$(1,2,2)$	Bit permutation	3 401.76	3.15	3 404.91	2.72
		Almost-MDS		216.62	3 618.38	4.19
	$(1,3,1)$	Bit permutation	2 249.92	3.15	**2 253.07**	**1.81**
		Almost-MDS		216.62	**2 466.54**	**3.28**
PRESENT		Bit permutation	4 448.00	3.15	4 451.15	3.79
GIFT		Bit permutation	3 481.12	3.15	3 484.27	3.32

5.6 Conclusions and Discussions

In this chapter, we present highly optimized TI circuits for cryptographically optimal 4×4 S-Boxes, obtained from CA rules. We classify such CA-based S-Boxes into 12 categories based on their amenability to low-area and low-power TI, and present direct-sharings for representative S-Boxes from each class. The architecture for our implementation direct-shares the local CA rule, and iterates over the same to obtain SCA resistant S-Box implementations. Subsequently, we reduce the number of shares further via functional decomposition of CA-rules, to obtain composite TI-circuits with even lower area footprint and power consumption. Our implementation results on ASIC (180nm technology) show that the most lightweight TI circuit among all CA-based S-boxes has a 49.42% smaller area-footprint and consumes 52.3% less power as compared to the best-known TI of the PRESENT S-Box. The same TI circuit also consumes 35.36% smaller area-footprint and consumes 44.46% less power as compared to a highly optimized TI of the GIFT S-Box. Finally, this TI circuit also passes the TVLA test over 1 000 000 power traces.

Subsequently, we present TI circuits for bit permutations and Mix Columns using almost-MDS matrices, with hardware results naturally favoring the former for lightweight applications. We finally present design paradigms for SPN block ciphers that combine TI circuits for our CA-based S-Boxes with TI circuits for bit permutations (and optionally, for Mix Columns operations) for full-fledged side-channel resistance. In particular, the use of TI-protected Mix Columns operation offers a practical trade-off between area and power savings and reasonable throughput requirements.

An apparent disadvantage inherent to any CA-based S-Box design strategy is the reduction in throughput due to its iterative nature. One possible workaround is to operate the target device at higher clock frequencies, keeping in mind that local CA rules are usually simple combinatorial circuits, and hence afford designs with higher critical frequencies.

Additionally, with respect to TI circuits, iterative architectures seem to minimize the possibility of additional leakages resulting from correlations among the output bits, since they are processed in different clock cycles. A more thorough exploration of the pros and cons of such iterative S-Box design principles can be an interesting direction for future work. Extensions of our design principles to TI circuits for 5×5 and 8×8 S-Boxes seems to be an intriguing direction for future research.

Chapter 6

Conclusion and Future Work

The vulnerability of block ciphers to active fault analysis and side-channel analysis makes it imperative to study new attacks that can be exploited to compromise its security and design suitable countermeasures against these attacks. Template based fault injection analysis attacks is a new form of fault attacks that have been introduced in chapter 2. This attack does not the exact knowledge of the underlying fault model. This thesis also deals with designing countermeasures against side-channel power analysis attacks. Chapter 3 and 4 design side-channel resistant implementations of the KHUDRA and AES. Lightweight designs for S-Boxes with countermeasures are also explored. Chapter 5 deals with S-Box designs with optimal cryptographic properties amenable to small designs with side-channel resistant countermeasures. The next section summarizes the results obtained in this thesis.

6.1 Summary of Results

Introduction of a new domain of fault attacks: Template-based fault injection analysis of block ciphers

In Chapter 2 we develop a new form of fault attack based on templates akin to template attacks in side-channel power attacks. We formulate a generic algorithm which of a template building followed by a template matching phase. This generic algorithm is easily instantiable for any target block cipher. This attack does not require high precision fault injection equipment and allows exploitation of low-granularity faults such as multi-byte faults. Most importantly, the exact knowledge of the underlying fault model is not required. To round off, a case-study of this attack targeting a hardware implementation of AES-128 on a Spartan-6 FPGA is presented in order to substantiate our claims.

Side-channel Resistant implementation for lightweight block cipher KHUDRA

In the light of increased focus on lightweight cryptography, given the emergence of IoT devices, this thesis deals with side-channel resistant implementations of lightweight block ciphers. KHUDRA is one such lightweight block cipher which had no previous side-channel resistant implementation. This thesis presents the design of the first threshold implementation of the block cipher KHUDRA. The 3 shared threshold implementation presented in this thesis requires lesser area than protected implementations of other well-known block ciphers like PRESENT, SIMON, SPECK etc.

Several masked implementations of the Boyar Peralta AES S-Box

In this thesis, we present several threshold implementations of the Boyar-Peralta AES S-Box which has received little attention with respect to masked implementations. We explore several area, randomness and clock cycles trade-offs to come up with 5 separate designs. In terms of resource requirements, our implementations compare very favourably with the existing implementations of the AES S-Box. Our smallest implementation requires 63% less randomness and the 50% less number of clock cycles compared to the smallest known masked implementation of the AES S-Box. We also present a TI design of the Boyar Peralta AES S-Box using 4 shares that requires no randomness at all. This is the first TI design of an AES S-Box with no randomness.

Designing lightweight and Side-channel Secure cryptographically optimal 4×4 S-Boxes from Cellular Automata Rules

In order to design lightweight side-channel secure block ciphers, cryptographically optimal S-Boxes with low area and power requirements are necessary. We formulate a design strategy of optimal 4×4 S-Boxes from CA rules in this thesis. We demonstrate that cellular automata can be used in order to design nonlinear functions. The single instance of the CA rule can be iterated over while cyclically shifting the input bits, to obtain one output bit of an S-Box at a time. This greatly reduces the area requirement of the design. A large proportion of the resulting S-Boxes achieve cryptographically optimal properties. We segregate S-Boxes into distinct classes based on their implementation overheads and their amenability to TI designs. One of our implementations on ASIC has a 49.42% smaller area-footprint and consumes 52.3% less power as compared to the best-known TI of the PRESENT S-Box. The same TI circuit also consumes 35.36% smaller area-footprint and consumes 44.46% less power as compared to a highly optimized TI of the GIFT S-Box. Therefore, our exploration

leads to one of the smallest masked implementations of a 4×4 cryptographically optimal S-Box.

6.2 Directions for Future Research

We put forward a few research problems that can be explored as an extension of the work done in this thesis. Some possible problems that can be possibly addressed in future work are:

1. Testing the efficacy of template-based fault injection analysis attacks in presence of traditional time-redundancy, round-redundancy and infective countermeasures against fault attacks.

2. Higher order threshold implementations of KHUDRA and AES for protection against more resourceful adversaries with the ability to carry out higher order power attacks.

3. Design of a secure implementation of the Boyar-Peralta S-Box starting from a masked Canright AES S-Box and using the optimizations mentioned in [BP12].

4. Using cellular automata rules to find optimal 5×5 and 8×8 S-Boxes with small overheads for masked implementations.

Disseminations

- **Ghoshal A.**, De Cnudde T. (2017) Several Masked Implementations of the Boyar-Peralta AES S-Box. In: Patra A., Smart N. (eds) Progress in Cryptology INDOCRYPT 2017. INDOCRYPT 2017. Lecture Notes in Computer Science, vol 10698. Springer, Cham

- Sadhukhan, R., Patranabis, S., **Ghoshal, A.**, Mukhopadhyay, D., Saraswat, V. and Ghosh, S., 2017. An Evaluation of Lightweight Block Ciphers for Resource-Constrained Applications: Area, Performance, and Security. Journal of Hardware and Systems Security, pp.1-16.

- **Ghoshal, A.**, Sadhukhan, R., Patranabis, S., Datta, N., Picek, S., Mukhopadhyay, D. (2018). Lightweight and Side-channel Secure 4 4 S-Boxes from Cellular Automata Rules. IACR Transactions on Symmetric Cryptology, 2018(3), 311-334. https://doi.org/10.13154/tosc.v2018.i3.311-334

- **Ghoshal, A.**, Patranabis S., Mukhopadhyay D. (2018) Template-Based Fault Injection Analysis of Block Ciphers. In: Chattopadhyay A., Rebeiro C., Yarom Y. (eds) Security, Privacy, and Applied Cryptography Engineering. SPACE 2018. Lecture Notes in Computer Science, vol 11348. Springer, Cham

Bibliography

[ADN+10] Michel Agoyan, Jean-Max Dutertre, David Naccache, Bruno Robisson, and Assia Tria. When Clocks Fail: On Critical Paths and Clock Faults. *Smart Card Research and Advanced Application*, pages 182–193, 2010.

[AP72] Serafino Amoroso and Yale N. Patt. Decision procedures for surjectivity and injectivity of parallel maps for tessellation structures. *Journal of Computer and System Sciences*, 6(5):448–464, 1972.

[BBBP13] Alessandro Barenghi, Guido M Bertoni, Luca Breveglieri, and Gerardo Pelosi. A fault induction technique based on voltage underfeeding with application to attacks against aes and rsa. *Journal of Systems and Software*, 86(7):1864–1878, 2013.

[BBI+15] Subhadeep Banik, Andrey Bogdanov, Takanori Isobe, Kyoji Shibutani, Harunaga Hiwatari, Toru Akishita, and Francesco Regazzoni. Midori: A block cipher for low energy. In *Proceedings, Part II, of the 21st International Conference on Advances in Cryptology — ASIACRYPT 2015 - Volume 9453*, pages 411–436, New York, NY, USA, 2015. Springer-Verlag New York, Inc.

[BCD+13] GT Becker, J Cooper, E DeMulder, G Goodwill, J Jaffe, G Kenworthy, T Kouzminov, A Leiserson, M Marson, P Rohatgi, et al. Test vector leakage assessment (tvla) methodology in practice. In *International Cryptographic Module Conference*, volume 1001, page 13, 2013.

[BCG+12] Julia Borghoff, Anne Canteaut, Tim Güneysu, Elif Bilge Kavun, Miroslav Knezevic, Lars R. Knudsen, Gregor Leander, Ventzislav Nikov, Christof Paar, Christian Rechberger, Peter Rombouts, Søren S. Thomsen, and Tolga Yalçin. PRINCE - A low-latency block cipher for pervasive computing applications - extended abstract. In *Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and*

Information Security, Beijing, China, December 2-6, 2012. Proceedings, pages 208–225, 2012.

[BDL97] Dan Boneh, Richard DeMillo, and Richard Lipton. On the importance of checking cryptographic protocols for faults. In *Advances in Cryptology-EUROCRYPT97*, pages 37–51. Springer, 1997.

[BDN+13] Begül Bilgin, Joan Daemen, Ventzislav Nikov, Svetla Nikova, Vincent Rijmen, and Gilles Van Assche. Efficient and first-order dpa resistant implementations of keccak. In *International Conference on Smart Card Research and Advanced Applications*, pages 187–199. Springer, 2013.

[BDPA06] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Radiogatún, a belt-and-mill hash function. *IACR Cryptology ePrint Archive*, 2006:369, 2006.

[BDPA11] Guido Bertoni, Joan Daemen, Michäel Peeters, and Gilles Van Assche. The Keccak reference, January 2011. `http://keccak.noekeon.org/`.

[BGN+14a] Begül Bilgin, Benedikt Gierlichs, Svetla Nikova, Ventzislav Nikov, and Vincent Rijmen. Higher-order threshold implementations. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 326–343. Springer, 2014.

[BGN+14b] Begül Bilgin, Benedikt Gierlichs, Svetla Nikova, Ventzislav Nikov, and Vincent Rijmen. A more efficient aes threshold implementation. In *International Conference on Cryptology in Africa*, pages 267–284. Springer, 2014.

[BGN+15] Begül Bilgin, Benedikt Gierlichs, Svetla Nikova, Ventzislav Nikov, and Vincent Rijmen. Trade-offs for threshold implementations illustrated on aes. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 34(7):1188–1200, 2015.

[Bil15] Begül Bilgin. Threshold implementations: as countermeasure against higher-order differential power analysis. 2015.

[BKL+07a] Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, Yannick Seurin, and C. Vikkelsoe. PRESENT: an ultra-lightweight block cipher. In *CHES*, volume 4727 of *LNCS*, pages 450–466. Springer, 2007.

[BKL+07b] Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, Yannick Seurin, and C. Vikkelsoe. PRESENT: an ultra-lightweight block cipher. In *CHES 2007*, pages 450–466, 2007.

[BNN+12] Begül Bilgin, Svetla Nikova, Ventzislav Nikov, Vincent Rijmen, and Georg Stütz. Threshold implementations of all 3× 3 and 4× 4 s-boxes. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 76–91. Springer, 2012.

[BP12] Joan Boyar and René Peralta. A small depth-16 circuit for the aes s-box. In *IFIP International Information Security Conference*, pages 287–298. Springer, 2012.

[BPP+17] Subhadeep Banik, Sumit Kumar Pandey, Thomas Peyrin, Yu Sasaki, Siang Meng Sim, and Yosuke Todo. Gift: A small present. In Wieland Fischer and Naofumi Homma, editors, *Cryptographic Hardware and Embedded Systems – CHES 2017*, pages 321–345, Cham, 2017. Springer International Publishing.

[BS91] Eli Biham and Adi Shamir. Differential Cryptanalysis of DES-like Cryptosystems. In *Proceedings of the 10th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '90, pages 2–21, London, UK, UK, 1991. Springer-Verlag.

[BS03] Johannes Blömer and Jean-Pierre Seifert. Fault Based Cryptanalysis of the Advanced Encryption Standard (AES). In Rebecca N. Wright, editor, *Financial Cryptography*, volume 2742 of *Lecture Notes in Computer Science*, pages 162–181. Springer, 2003.

[Can05] D. Canright. *A Very Compact S-Box for AES*, pages 441–455. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

[Car10a] Claude Carlet. Boolean Functions for Cryptography and Error Correcting Codes. In Yves Crama and Peter L. Hammer, editors, *Boolean Models and Methods in Mathematics, Computer Science, and Engineering*, pages 257–397. Cambridge University Press, New York, NY, USA, 1st edition, 2010.

[Car10b] Claude Carlet. Vectorial Boolean Functions for Cryptography. In Yves Crama and Peter L. Hammer, editors, *Boolean Models and Methods in Mathematics, Computer Science, and Engineering*, pages 398–469. Cambridge University Press, New York, USA, 1st edition, 2010.

[CCF+08] Gaetan Canivet, Jessy Clédière, Jean Baptiste Ferron, Frédéric Valette, Marc Renaudin, and Régis Leveugle. Detailed analyses of single laser shot effects in the configuration of a virtex-ii fpga. In *On-Line Testing Symposium, 2008. IOLTS'08. 14th IEEE International*, pages 289–294. IEEE, 2008.

[CDGP93] L. Claesen, J. Daemen, M. Genoe, and G. Peeters. Subterranean: A 600 Mbit/sec cryptographic VLSI chip. In *Computer Design: VLSI in Computers and Processors, 1993. ICCD '93. Proceedings., 1993 IEEE International Conference on*, pages 610–613, Oct 1993.

[CJRR99] Suresh Chari, Charanjit Jutla, Josyula Rao, and Pankaj Rohatgi. Towards sound approaches to counteract power-analysis attacks. In *Advances in CryptologyCRYPTO99*, pages 791–791. Springer, 1999.

[CKN00] JS Coron, P Kocher, and D Naccache. Statistics and secret leackage, to appear in proceedings of financial cryptography, 2000.

[CML+11] Gaetan Canivet, Paolo Maistri, Régis Leveugle, Jessy Clédière, Florent Valette, and Marc Renaudin. Glitch and laser fault attacks onto a secure aes implementation on a sram-based fpga. *Journal of Cryptology*, 24(2):247–268, 2011.

[CN17] T. De Cnudde and S. Nikova. Securing the present block cipher against combined side-channel analysis and fault attacks. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, PP(99):1–11, 2017.

[CNK04] Jean-Sebastien Coron, David Naccache, and Paul Kocher. Statistics and secret leakage. *ACM Transactions on Embedded Computing Systems (TECS)*, 3(3):492–508, 2004.

[CRR02] Suresh Chari, Josyula R Rao, and Pankaj Rohatgi. Template attacks. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 13–28. Springer, 2002.

[CT05] Hamid Choukri and Michael Tunstall. Round reduction using faults. *FDTC*, 5:13–24, 2005.

[DC98] Joan Daemen and Craig S. K. Clapp. Fast Hashing and Stream Encryption with PANAMA. In *Fast Software Encryption, 5th International Workshop, FSE '98, Paris, France, March 23-25, 1998, Proceedings*, pages 60–74, 1998.

[DCBG+17] Thomas De Cnudde, Begül Bilgin, Benedikt Gierlichs, Ventzislav Nikov, Svetla Nikova, and Vincent Rijmen. Does coupling affect the security of masked implementations? In *International Workshop on Constructive Side-Channel Analysis and Secure Design*, pages 1–18. Springer, 2017.

[DCBR+15] Thomas De Cnudde, Begül Bilgin, Oscar Reparaz, Ventzislav Nikov, and Svetla Nikova. Higher-order threshold implementation of the aes s-box. In *International Conference on Smart Card Research and Advanced Applications*, pages 259–272. Springer, 2015.

[DCRB+16] Thomas De Cnudde, Oscar Reparaz, Begül Bilgin, Svetla Nikova, Ventzislav Nikov, and Vincent Rijmen. Masking aes with d+ 1 shares in hardware. In *International Conference on Cryptographic Hardware and Embedded Systems*, pages 194–212. Springer, 2016.

[DDRT12] Amine Dehbaoui, J-M Dutertre, Bruno Robisson, and Assia Tria. Electromagnetic transient faults injection on a hardware and a software implementations of aes. In *Fault Diagnosis and Tolerance in Cryptography (FDTC), 2012 Workshop on*, pages 7–15. IEEE, 2012.

[DGV94] Joan Daemen, René Govaerts, and Joos Vandewalle. A new approach to block cipher design. In Ross Anderson, editor, *Fast Software Encryption: Cambridge Security Workshop Cambridge, U. K.,1993 Proceedings*, pages 18–32, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg.

[DLV03] Pierre Dusart, Gilles Letourneux, and Olivier Vivolo. Differential fault analysis on aes. In *Applied Cryptography and Network Security*, pages 293–306. Springer, 2003.

[DP08] Stefan Dziembowski and Krzysztof Pietrzak. Leakage-resilient cryptography. In *Foundations of Computer Science, 2008. FOCS'08. IEEE 49th Annual IEEE Symposium on*, pages 293–302. IEEE, 2008.

[DR00] Joan Daemen and Vincent Rijmen. Rijndael for AES. In *AES Candidate Conference*, pages 343–348, 2000.

[DRS+12] François Durvaux, Mathieu Renauld, François-Xavier Standaert, Loic van Oldeneel tot Oldenzeel, and Nicolas Veyrat-Charvillon. Cryptanalysis of the ches 2009/2010 random delay countermeasure. *IACR Cryptology ePrint Archive*, 2012:38, 2012.

[FJLT13] Thomas Fuhr, Eliane Jaulmes, Victor Lomné, and Adrian Thillard. Fault attacks on aes with faulty ciphertexts only. In *Fault Diagnosis and Tolerance in Cryptography (FDTC), 2013 Workshop on*, pages 108–118. IEEE, 2013.

[GGJR+11] Benjamin Jun Gilbert Goodwill, Josh Jaffe, Pankaj Rohatgi, et al. A testing methodology for side-channel resistance validation. In *NIST Non-invasive attack testing workshop*, 2011.

[GMK16] Hannes Gross, Stefan Mangard, and Thomas Korak. Domain-oriented masking: Compact masked hardware implementations with arbitrary protection order. *IACR Cryptology ePrint Archive*, 2016:486, 2016.

[GMO01] Karine Gandolfi, Christophe Mourtel, and Francis Olivier. Electromagnetic analysis: Concrete results. In *Cryptographic Hardware and Embedded SystemsCHES 2001*, pages 251–261. Springer, 2001.

[GP99] Louis Goubin and Jacques Patarin. Des and differential power analysis the duplication method. In *Cryptographic Hardware and Embedded Systems*, pages 728–728. Springer, 1999.

[GPQ11] Laurie Genelle, Emmanuel Prouff, and Michaël Quisquater. Thwarting higher-order side channel analysis with additive and multiplicative maskings. In Bart Preneel and Tsuyoshi Takagi, editors, *Cryptographic Hardware and Embedded Systems – CHES 2011: 13th International Workshop, Nara, Japan, September 28 – October 1, 2011. Proceedings*, pages 240–255, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[GR17] Dahmun Goudarzi and Matthieu Rivain. *How Fast Can Higher-Order Masking Be in Software?*, pages 567–597. Springer International Publishing, Cham, 2017.

[GYTS14] Nahid Farhady Ghalaty, Bilgiday Yuce, Mostafa Taha, and Patrick Schaumont. Differential fault intensity analysis. In *Fault Diagnosis and Tolerance in Cryptography (FDTC), 2014 Workshop on*, pages 49–58. IEEE, 2014.

[HMER] Karine Heydemann, Nicolas Moro, Emmanuelle Encrenaz, and Bruno Robisson. Formal verification of a software countermeasure against instruction skip attacks. In *PROOFS 2013*.

[HR10] Viet Tung Hoang and Phillip Rogaway. On generalized feistel networks. In *CRYPTO*, volume 6223 of *LNCS*, pages 613–630. Springer, 2010.

[ISW03] Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. In *Annual International Cryptology Conference*, pages 463–481. Springer, 2003.

[JS17] Anthony Journault and François-Xavier Standaert. *Very High Order Masking: Efficient Implementation and Security Evaluation*, pages 623–643. Springer International Publishing, Cham, 2017.

[Kim10] Chong Hee Kim. Differential fault analysis against aes-192 and aes-256 with minimal faults. In *Fault Diagnosis and Tolerance in Cryptography (FDTC), 2010 Workshop on*, pages 3–9. IEEE, 2010.

[KJJ99] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *Advances in cryptologyCRYPTO99*, pages 789–789. Springer, 1999.

[KM14] Souvik Kolay and Debdeep Mukhopadhyay. Khudra: A new lightweight block cipher for fpgas. In *SPACE*, volume 8804 of *LNCS*, pages 126–145. Springer, 2014.

[Koc96] Paul C Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In *Annual International Cryptology Conference*, pages 104–113. Springer, 1996.

[LP07] G. Leander and A. Poschmann. On the Classification of 4 Bit S-Boxes. In Claude Carlet and Berk Sunar, editors, *Arithmetic of Finite Fields*, volume 4547 of *Lecture Notes in Computer Science*, pages 159–176. Springer Berlin Heidelberg, 2007.

[LSG+10] Yang Li, Kazuo Sakiyama, Shigeto Gomisawa, Toshinori Fukunaga, Junko Takahashi, and Kazuo Ohta. Fault sensitivity analysis. In *CHES*, volume 6225, pages 320–334. Springer, 2010.

[MBTM17] Kerry A. McKay, Larry Bassham, Meltem Snmez Turan, and Nicky Mouha. Report on Lightweight Cryptography. 2017. `http://nvlpubs.nist.gov/nistpubs/ir/2017/NIST.IR.8114.pdf`.

[MMG14] Oliver Mischke, Amir Moradi, and Tim Güneysu. Fault sensitivity analysis meets zero-value attack. In *2014 Workshop on Fault Diagnosis and Tolerance in Cryptography, FDTC 2014, Busan, South Korea, September 23, 2014*, pages 59–67, 2014.

[MOP08] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power analysis attacks: Revealing the secrets of smart cards*, volume 31. Springer Science & Business Media, 2008.

[MPC] Willi Meier, Enes Pasalic, and Claude Carlet. Algebraic attacks and decomposition of boolean functions. In Christian Cachin and Jan L. Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004*.

[MPL+11] Amir Moradi, Axel Poschmann, San Ling, Christof Paar, and Huaxiong Wang. Pushing the limits: a very compact and a threshold implementation of aes. In *Eurocrypt*, volume 6632, pages 69–88. Springer, 2011.

[MPLJ17] Luca Mariot, Stjepan Picek, Alberto Leporati, and Domagoj Jakobovic. Cellular automata based s-boxes. Cryptology ePrint Archive, Report 2017/1055, 2017. `https://eprint.iacr.org/2017/1055`.

[MPO05] Stefan Mangard, Norbert Pramstaller, and Elisabeth Oswald. Successfully attacking masked aes hardware implementations. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 157–171. Springer, 2005.

[Muk09] Debdeep Mukhopadhyay. An improved fault based attack of the advanced encryption standard. *Africacrypt*, 5580:421–434, 2009.

[MY93] Mitsuru Matsui and Atsuhiro Yamagishi. A new method for known plaintext attack of FEAL cipher. In *Proceedings of the 11th annual international conference on Theory and application of cryptographic techniques*, EUROCRYPT'92, pages 81–91, Berlin, Heidelberg, 1993. Springer-Verlag.

[NRR06] Svetla Nikova, Christian Rechberger, and Vincent Rijmen. Threshold implementations against side-channel attacks and glitches. In *International Conference on Information and Communications Security*, pages 529–545. Springer, 2006.

[Nyb93] Kaisa Nyberg. On the construction of highly nonlinear permutations. In RainerA. Rueppel, editor, *Advances in Cryptology - EUROCRYPT' 92*, volume 658 of *Lecture Notes in Computer Science*, pages 92–98. Springer Berlin Heidelberg, 1993.

[Nyb94] Kaisa Nyberg. S-boxes and round functions with controllable linearity and differential uniformity. In *Fast Software Encryption: Second International Workshop. Leuven, Belgium, 14-16 December 1994, Proceedings*, pages 111–130, 1994.

[PCNM15] Sikhar Patranabis, Abhishek Chakraborty, Phuong Ha Nguyen, and Debdeep Mukhopadhyay. A biased fault attack on the time redundancy countermeasure for aes. In *International Workshop on Constructive Side-Channel Analysis and Secure Design*, pages 189–203. Springer, 2015.

[PMK+11] Axel Poschmann, Amir Moradi, Khoongming Khoo, Chu-Wee Lim, Huaxiong Wang, and San Ling. Side-channel resistant crypto for less than 2,300 ge. *Journal of Cryptology*, 24(2):322–345, 2011.

[PMY+17] Stjepan Picek, Luca Mariot, Bohan Yang, Domagoj Jakobovic, and Nele Mentens. Design of s-boxes defined with cellular automata rules. In *Proceedings of the Computing Frontiers Conference, CF'17, Siena, Italy, May 15-17, 2017*, pages 409–414, 2017.

[PQ03] Gilles Piret and Jean-Jacques Quisquater. A differential fault attack technique against spn structures, with application to the aes and khazad. In *CHES*, volume 2779, pages 77–88. Springer, 2003.

[PR13] Emmanuel Prouff and Matthieu Rivain. Masking against side-channel attacks: A formal security proof. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 142–159. Springer, 2013.

[RBN+15] Oscar Reparaz, Begül Bilgin, Svetla Nikova, Benedikt Gierlichs, and Ingrid Verbauwhede. Consolidating masking schemes. In *Annual Cryptology Conference*, pages 764–783. Springer, 2015.

[RM07] Bruno Robisson and Pascal Manet. Differential behavioral analysis. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 413–426. Springer, 2007.

[RP10] Matthieu Rivain and Emmanuel Prouff. Provably secure higher-order masking of aes. In Stefan Mangard and François-Xavier Standaert, editors, *Cryptographic Hardware and Embedded Systems, CHES 2010: 12th International Workshop, Santa Barbara, USA, August 17-20, 2010. Proceedings*, pages 413–427, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[Saa12] Markku-Juhani O. Saarinen. Cryptographic analysis of all 4 x 4-bit s-boxes. In Ali Miri and Serge Vaudenay, editors, *Selected Areas in Cryptography*, pages 118–133, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[SGD08] Nidhal Selmane, Sylvain Guilley, and J-L Danger. Practical setup time violation attacks on aes. In *Dependable Computing Conference, 2008. EDCC 2008. Seventh European*, pages 91–96. IEEE, 2008.

[SM15] Tobias Schneider and Amir Moradi. Leakage assessment methodology. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 495–513. Springer, 2015.

[SMC09] Dhiman Saha, Debdeep Mukhopadhyay, and Dipanwita Roy Chowdhury. A diagonal fault attack on the advanced encryption standard. *IACR Cryptology ePrint Archive*, 2009:581, 2009.

[STE17] Aria Shahverdi, Mostafa Taha, and Thomas Eisenbarth. Lightweight side channel resistance: Threshold implementations of simon. *IEEE Transactions on Computers*, 66(4):661–671, 2017.

[Sut91] Klaus Sutner. De bruijn graphs and linear cellular automata. *Complex Systems*, 5(1):19–30, 1991.

[TMA11] Michael Tunstall, Debdeep Mukhopadhyay, and Subidh Ali. Differential fault analysis of the advanced encryption standard using a single fault. *WISTP*, 6633:224–233, 2011.

[TV04] Kris Tiri and Ingrid Verbauwhede. A logic level design methodology for a secure dpa resistant asic or fpga implementation. In *Proceedings of the conference on Design, automation and test in Europe-Volume 1*, page 10246. IEEE Computer Society, 2004.

[Wol83] Stephen Wolfram. Statistical mechanics of cellular automata. *Reviews of modern physics*, 55(3):601, 1983.

[Wol84a] Stephen Wolfram. Cellular automata as models of complexity. *Nature*, 311(5985):419, 1984.

[Wol84b] Stephen Wolfram. Universality and complexity in cellular automata. *Physica D: Nonlinear Phenomena*, 10(1-2):1–35, 1984.

[YZS+15] Gangqiang Yang, Bo Zhu, Valentin Suder, Mark D. Aagaard, and Guang Gong. The simeck family of lightweight block ciphers. In *CHES*, volume 9293 of *LNCS*, pages 307–329. Springer, 2015.

[ZBL+15] WenTao Zhang, ZhenZhen Bao, DongDai Lin, Vincent Rijmen, BoHan Yang, and Ingrid Verbauwhede. Rectangle: a bit-slice lightweight block cipher suitable for multiple platforms. *Science China Information Sciences*, 58(12):1–15, 2015.