SECURITY ANALYSIS OF LIGHTWEIGHT CRYPTOGRAPHIC PRIMITIVES

by
William Unger

A dissertation

submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy in Computing, Cyber Security

Boise State University

May 2022

BOISE STATE UNIVERSITY GRADUATE COLLEGE

DEFENSE COMMITTEE AND FINAL READING APPROVALS

of the dissertation submitted by

William Unger

Dissertation Title: Security Analysis of Lightweight Cryptographic Primitives

Date of Final Oral Examination: 1 February 2022

The following individuals read and discussed the dissertation submitted by student William Unger, and they evaluated the student's presentation and response to questions during the final oral examination. They found that the student passed the final oral examination.

Liljana Babinkostova Ph.D.	Chair, Supervisory Committee
Robert Erbes M.S.	Member, Supervisory Committee
Sin Ming Loo Ph.D.	Member, Supervisory Committee
Marion Scheepers Ph.D.	Member, Supervisory Committee

The final reading approval of the dissertation was granted by Liljana Babinkostova Ph.D., Chair of the Supervisory Committee. The dissertation was approved by the Graduate College.

DEDICATION

Dedicated to My Family.

ACKNOWLEDGMENT

There are no words to describe by deep gratefulness for my advisor Dr. Liljana Babinkostova. During my PhD journey she saw greatness in me even when I did not and molded me into the a PhD level researcher. I will take the lessons she taught me and use that in my future career.

I express my thankfulness to Robert Erbes, Marion Scheepers and Mike Borowczak for their collaboration and advising on some of the dissertation research as well as Riley Myers and Erik Corrington. I would also like to thank the organizations Boise State University, Idaho National Laboratory, Battelle Energy Alliance, and Sigma Xi for their support of the dissertation research.

I would not have been successful in this journey if it was not for the constant support of my family. Lastly I would like to thank my dog Nikki in which she was always able to cheer me up even in the darkest of times.

ABSTRACT

Symmetric key cryptographic primitives are essential to encrypt data and protect communication between parties. Due to resource constraints, some modern devices are not capable of executing traditional cryptographic algorithms. This fact necessitates new lightweight cryptographic algorithms. Current research into lightweight cryptology is vast, in part due to the National Institute of Standards and Technology's (NIST) lightweight cryptographic standardization process.

There is not much research into the vulnerability to a power analysis attack created by the choice of parameters of lightweight symmetric ciphers. This dissertation develops and demonstrates white box and black box cryptanalysis models for power analysis attacks on lightweight cryptographic primitives.

The white box cryptanalysis targets the GIFT-COFB family of lightweight ciphers that include NIST lightweight standard finalists, and examines the security of their substitution layers in the power analysis setting. Findings include: When deployed over fields of characteristic 2, the most used platform, the non-linearity metric provides the best prediction of susceptibility to power analysis attacks. When deployed over fields of characteristic 3, substitution boxes display a wide range of vulnerability to power analysis attacks, leading to a classification of substitution boxes into weak and strong categories.

The black box cryptanalysis focuses on a proprietary cryptosystem acting between

two embedded systems which require a lightweight cipher. The results of the black box cryptanalysis include a model for the decryption process of the proprietary system, and a software implementation of a prediction algorithm that predicts the plaintext giving rise to given ciphertext values.

These research results shed a new light on the resilience of lightweight cryptographic protocols against side-channel and black-box attacks and help in bridging the gap between theory and practice.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AES Advanced Encryption Standard

CPA Correlation Power Analysis

DPA Differential Power Analysis

DPA-SNR Differential Power Analysis Signal to Noise Ratio

DUT Device Under Target

HD Hamming Distance

HW Hamming Weight

NL Non Linearity

POI Point of Interest

RTO Revisited Transparency Order

SCA Side Channel Attack

SNR Signal to Noise Ratio

SPN Substitution Permutation Network

SR Success Rate

TO Transparency Order

CHAPTER 1:

INTRODUCTION

With the rise of Internet of Things (IoT) devices and cyber-physical systems (CPS), there is now a need to construct resilient countermeasures to physical attacks on the cryptographic implementations such as general defences (Bayrak et al., 2011; Chari et al., 1999) and multiplicative masking (Golić and Tymen, 2002).

Lightweight cryptology finds a balance in size, speed, and security for devices with resource constrains. The first noted lightweight cryptographic scheme was the work of (Bogdanov et al., 2007) in their creation of the cryptogprahic primitive PRESENT, but since then more lightweight ciphers have been introduced and will be noted later in this chapter.

1.1 Introduction Symmetric Key Cryptology

A symmetric key cipher operates over three sets being: $\mathcal{K}$ (Key Space), $\mathcal{M}$ (Message Space), and $\mathcal{C}$ (Ciphertext Space). Symmetric key primitives also make use of three algorithms that operate on the sets. These algorithms are: KeyGen - An algorithm that randomly produces a key $k \in \mathcal{K}$, Enc - An algorithm that takes as input a key $k \in \mathcal{K}$ and message $m \in \mathcal{M}$ that produces as an output a ciphertext $c \in \mathcal{C}$, and Dec - a deterministic algorithm that takes as input a key $k \in \mathcal{K}$ and ciphertext $c \in \mathcal{C}$ and outputs a plaintext $m \in \mathcal{M}$. A symmetric encryption primitive as considered as a

tuple (KeyGen,Enc,Dec,$\mathcal{K}, \mathcal{M}, \mathcal{C}$). In order for the primitive to be considered correct the decryption of the encryption of any message given a constant key must result in the initial message. This correctness propriety is more formally stated as:

$$Pr\big[Dec(k, Enc(k, m)) = m\big] = 1$$

Is is important to be able to formally describe what security is in the setting of symmetric key cryptography. While no practical cipher qualifies as being perfectly secure, we will use the notion of describing if a symmetric key cipher is computationally secure. Consider an instance in which a challenger C randomly selects two message m_0, m_1 with fixed size n from the message space $\mathcal{M}$ and sends the pair of messages to an encrypter E. The encrypter then randomly computes a bit $b \in \{0, 1\}$ and encrypts the message using a fixed key k as follows:

$$c = Enc(k, m_b)$$

The encrypter then sends c to the challenger. With knowledge of c, the challenger produces their guess of the b value being b'. If for all polynomial time challenger strategies there exists a negligible function $\epsilon(x)$ such that

$$Pr[b' = b] \leq \frac{1}{2} + \epsilon(n)$$

then the scheme is computationally secure.

Because of the needed security for computational devices, the U.S. Department of Commerce's National Institute of Standards and Technology (NIST) makes standards

for commercial, industry, and government use. There have been two standards for symmetric key primitives known as the Data Encryption Standard (DES) based upon the Lucifer cipher from the work of (Feistel, 1974) which was the industry standard until 1999, and the Advanced Encryption Standard (AES) which is still the standard at the time of writing this dissertation based on the work of (Daemen and Rijmen, 1998, 2001). While DES and AES are the NIST standards, there exist many other symmetric primitives such as RC5 (Rivest, 1994), Blowfish (Schneier, 1993), and Serpent (Anderson et al., 1998).

Block Ciphers

A block cipher is a special type of symmetric key cryptosystem. Similar to the symmetric key definition we have a key space $\mathcal{K}$, but differing from the symmetric key definition we have a message space $\mathcal{M}$ acting as both the plaintext and ciphertext space. Given a fixed key k we denote the encryption function as $Enc_k(p)$ where p as a plaintext. Similarly the decryption function is $Dec_k(c)$ where c is a ciphertext. Both the encryption and decryption are of the form $\mathcal{K} \times \mathcal{M} \to \mathcal{M}$ where $k \in \mathcal{K}$ and $p, c \in \mathcal{M}$

In order to perform the encryption process, a block cipher computes many sub-functions known as rounds. In each round, the output of one round is used as the input of another round. The incorporation of these rounds is used to provide diffusion (a small bit change in the input will lead to a large change overall of approximately half of the output bits), confusion (meaning that each output bit is dependent on several bits of the key) (Shannon, 1949).

A common form of block cipher is the Substitution Permutation Network (SPN)

in which the rounds consist of round functions that substitute data, compute permutations on the data, and add key content into the intermediary states. In SPN ciphers confusion is provided in the substitution box (S-Box) layer and diffusion is provided in the permutation layer.

1.1.1 Lightweight Cryptology

Lightweight cryptology attempts to find a balance in size, speed, and security for devices with resource constraints. The standards such as the Advanced Encryption Standard (AES) and Triple Data Encryption Standard (3DES) are made for 'regular' devices such as servers, laptops, desktops, and smart phones that do not have the constraint of size. In embedded systems, using small/cheap micro-controllers or FPGAs, RAM and gate equivalences (GE), are limited and need to perform what the device is intended to do as well as be secure. Lightweight cryptology minimizes the size of a cryptographic implementation and retains speed and security. Many lightweight cryptographic schemes are a block cipher construction such as PRESENT (Bogdanov et al., 2007), GIFT (Banik et al., 2017), SIMON/SPECK (Beaulieu et al., 2015), PRINCE (Borghoff et al., 2012), and PICCOLO (Shibutani et al., 2011). Lightweight cryptographic schemes have use in cyber physical systems as noted in the works of (Al Faruque et al., 2015; Kocabas et al., 2016).

When performing cryptanalysis on lightweight cryptographic implementations the knowledge of the attacker varies. In some cases we assume that the attacker has access to all information except for the private key which is noted as white box cryptanalysis. In other situations the attacker has no initial knowledge of the implementation and can only conduct attacks based upon observations of plaintext/ciphertext pairs which is noted as black box cryptanalysis. In this dissertation we consider both

these situations, white box cryptanalysis and black box cryptanalysis, as we perform cryptanalysis on lightweight cryptographic schemes.

In our research we will be comparing the security of the S-Box parameter in the power analysis setting. We will be using the GIFT64 cryptographic primitive (Banik et al., 2017) as a starting point and replace S-Boxes in our implementation with others and study the security implementations. While the GIFT cipher will be explained in detail in Chapter 2, we will state some of the design choices here. GIFT64 is a member of the GIFT cryptographic family. GIFT64 has 28 round in which each round is made up of S-Box, PermBits, and AddRoundKey layers. The S-Box is a 4-bit to 4-bit bijective function that acts as the non-linear layer of the cipher. The PermBits function is a invertible bit permutation that provides diffusion into the cipher. Lastly, the AddRoundKey function introduces key content into each round of the cipher.

Given these attack models there also exist knowledge assumptions on the attacker that classify the attacks into White and Black box attacks. White box attacks assume the attacker knows everything except for the cryptographic key and conversely in Black box the attacker has no knowledge of the underlying algorithm. Works comparing the attack assumptions are shown in (Biryukov et al., 2014).

Many lightweight cryptographic scheme were submitted for the NIST LWC standard and the NIST reports are noted in (Turan et al., 2019; Mohajerani et al., 2020; Mancillas-López et al., 2020), as well as in some papers (Bovy et al., 2020). Currently the LWC competition has 10 finalists being: ASCON (Dobraunig et al., 2016), Elephant (Mennink, 2021), GIFT-COFB (Banik et al., 2020), Grain128-AEAD (Hell et al., 2019), ISAP (Dobraunig et al., 2017), Photon-Beetle (Bao et al., 2019), Romulus (Iwata et al., 2020), Sparkle (Beierle et al., 2020), TinyJambu (Wu and Huang,

2019), and Xoodyak (Daemen et al., 2020).

The graphs below in Figure 1.1 and Figure 1.2 show FPGA and micro-controller comparison of the finalists. Figure 1.1 shows a comparison on size and throughput of all of the finalists on the Xilinx Artix-7 FPGA, except for GRAIN128-AEAD which was not part of the data-set gathered in (Mohajerani et al., 2020). As their were multiple variations of each FPGA implementation, we chose the versions that had the highest throughput. In this comparison we note that ideally the FPGA implementations should minimize size and maximize the throughput. Figure 1.2 shows a comparison of the finalists in the micro-controller setting using a ARM Cortex M3. This figure uses the speed up ratio between the LWC implementation and the cipher ChaChaPoly to compare how fast the LWC implementation is. The ratio is defined as the time needed for the implementation to encrypt divided by the time needed for ChaChaPol to encrypt the data. We considered two cases including a 16 byte payload and a 128 byte payload. The larger the ratio, the faster the micro-controller encryption is. The data gathered for this comparison is in the work of (Weatherley, 2021).

1.2 Introduction Side Channel Attacks

In the classical attack models described in the previous section, the attacker only has knowledge of plaintext/ciphertext pairs in which that attacker can choose plaintext or ciphertext values. Common attacks for knowledge of known/chosen plaintext/ciphertext pairs are: Linear cryptanalysis (Matsui, 1993), Differential cryptanalysis (Biham and Shamir, 1991), Algebraic cryptanalysis (Courtois and Bard, 2007), Integral cryptanalysis (Knudsen and Wagner, 2002) and many other classical cryptanalysis techniques. There exist states when the attacker can also gain temporary ac-

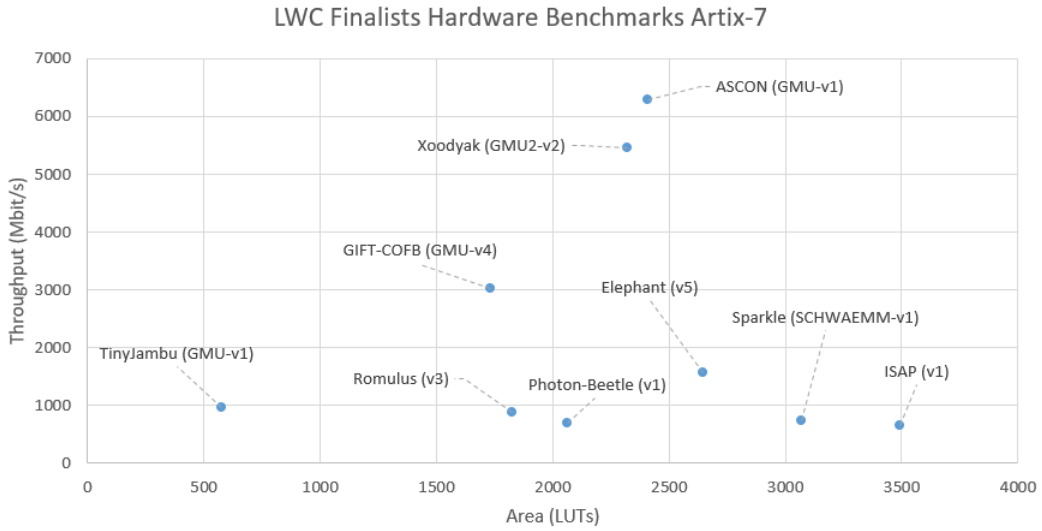

Figure 1.1: LWC Finalists Comparison FPGA

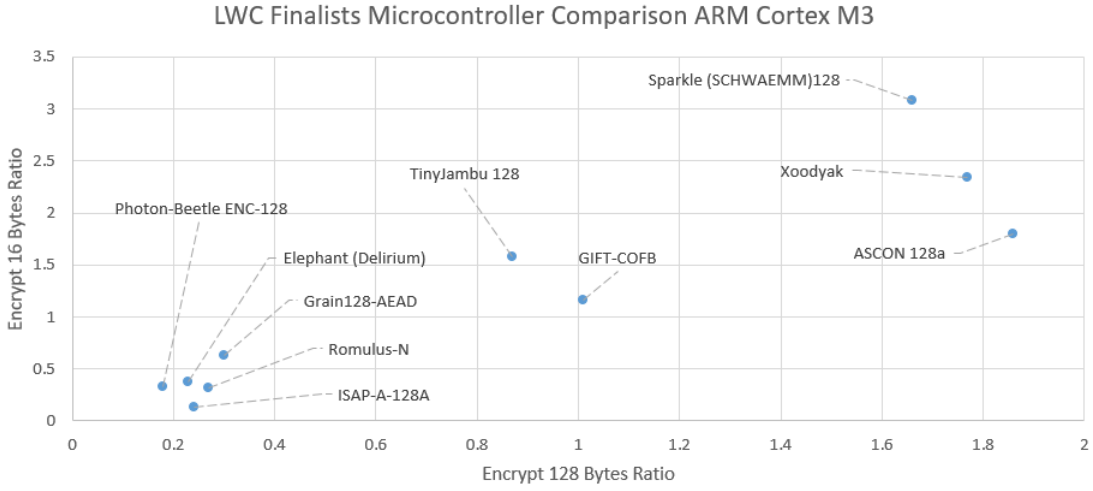

Figure 1.2: LWC Finalists Comparison Micro-controller

cess to a cryptographic device and exploit the device while the encryption/decryption operations are computing to gain more knowledge than the classical input/output.

Side Channel Attacks (SCA) are exploiting the physical effects while a cryptographic device is performing the encryption/decryption operation. The study of SCA started with the novel works of Kocher in (Kocher, 1996; Kocher et al., 1999). Examples of measured physical effects are timing, power draw, and electromagnetic radiation. While the attacker needs physical access to get these physical measurements, once obtained along with the corresponding plaintext/ciphertext pairs, they can break cryptographic schemes that do not implement countermeasures against SCA.

Since the seminal work of Kocher, there has been vast research into SCA included the noted Correlation Power Analysis (CPA) attack (Brier et al., 2004). Originally SCA attacks focused on DES and AES, but in the last decade there has been research into SCA on other symmetric ciphers such as: GIFT (Zhang et al., 2021), SIMON (Shanmugam et al., 2014), and PRINCE (Selvam et al., 2015).

Along with research in SCA, there has also been research into countermeasures on these attacks. Some examples of countermeasures on SCA are creating implementations that have a constant power draw (Sundaresan et al., 2008; Popp et al., 2007) and to implement masking (Golić and Tymen, 2002; Standaert et al., 2005; De Cnudde et al., 2015). Countermeasures applied to cryptographic schemes of ciphers can be seen in the work of AES (Oswald et al., 2005), SPECK (Chen et al., 2016), GIFT (Gupta et al., 2021), and many others.

Research into side channel attacks and countermeasures is crucial for the security of internet connected devices. If a cryptographic key is lost to an attacker, the attacker

can learn the secrets of the said devices. The attacker can then impersonate devices to send forged data and could also have an entry point to attack a victim's computer network. Besides the loss of a cryptographic key, other sensitive information can be obtained using side channel attacks. Examples of real world situations that can use SCA are: acoustic/thermal attacks on manufacturing systems (Al Faruque et al., 2016a,b), attacks on hardware cryptocurrency wallets (San Pedro et al., 2019; Gkaniatsou et al., 2017), attacks on sensor networks (Moabalobelo et al., 2012; Pongaliur et al., 2008), attacks on medical devices (Pycroft and Aziz, 2018; Zhang et al., 2014), attacks on biometric security (Galbally, 2020; Dürmuth et al., 2016), and attacks on critical infrastructure (Tsalis et al., 2019, 2018)

While there is research in lightweight cryptology in the power analysis setting, most of the research is exploiting or mitigations of a single cipher. There is not much research into the security of parameter choices of lightweight ciphers in the power analysis setting.

1.3 Generalizing Cryptographic Schemes

In this dissertation we will be generalizing the GIFT64 (Banik et al., 2017) cryptographic scheme and create a ternary case study. One might wonder what is the point of doing this, but the current most use variants of public key encryption are elliptic curve variants of classical Diffie Hellman key exchange (Diffie and Hellman, 1976) and ElGamal encryption and (ElGamal, 1985). Elliptic curves were first used as a factoring method (Montgomery, 1987; Atkin and Morain, 1993), but then researchers used elliptic curves themselves to create cryptosystems (Koblitz, 1987; Menezes and Vanstone, 1993). At first, the elliptic curve cryptosystems were not mainstream due to the hardware/software constraints at the time but after a few years they were more

practical (López and Dahab, 1998; Hankerson et al., 2000; Harkanson and Kim, 2017). Elliptic curves are now a very popular method of public key encryption used in Bitcoin (Nakamoto, 2008), OpenSSL libraries (Käsper, 2011), wireless communication (Lauter, 2004), and many other transmissions of data (Bos et al., 2014).

1.4 White and Black Box Cryptanalysis

When modeling cryptanalysis there exist two models being white box cryptanalysis and black box cryptanalysis. These models describe how much knowledge the attacker knows when performing the attacks. In black box, the attacker has no underlying knowledge of the algorithm/implementation and only has input and output data. In white box it is assumed that the attacker has knowledge of everything except the private cryptographic key. Cryptology resistant to white box cryptanalysis are white box implementations (Preneel and Wyseur, 2008) and examples are shown in works on AES (Chow et al., 2002a; Billet et al., 2004) and DES (Chow et al., 2002b). However it is shown in literature that white box cryptology is not necessarily secure against SCA techniques (Sasdrich et al., 2016). Grey box is located in the middle in which the attacker has more knowledge then in the black box setting but does not necessarily know as much as compared to an attacker in the white box setting. Side channel cryptanalysis techniques are in the grey box attack model. A comparison of the white and black box cryptogprahic models can be found in (Biryukov et al., 2014)

1.4.1 Black Box Cryptanalysis

In normal cryptanalysis it is assumed that the attacker has knowledge of everything except the private key but this is not the case in black box attacks. Black box cryptanalysis is performing cryptanalysis without knowledge of the underlying cryp-

tographic algorithm. Every case study in Black-Box will vary some of the methodologies can be used in different cases. Based upon ciphertexts an attacker can try and classify the cryptographic primitive (Tan and Ji, 2016). While there is not much literature on attacking unknown home-made encryption schemes, an example case study of Black box cryptanalysis is shown in (Capillon, 2016). Applying some machine learning in the black box setting is considered to be a black box model and can be found in (Alallayah et al., 2012; Xiao et al., 2019), as well as quantum black box cryptanalysis in the works of (Brassard et al., 1998).

1.4.2 White Box Cryptanalysis

In contrast to black box cryptanalysis, the white box cryptanalysis method differs in the knowledge assumptions of the attacker. Classically in white box cryptanalysis, the attacker has access to everything imaginable except for the private key. With knowledge of the scheme/implementation the attacker can conduct chosen plaintext/ciphertext attacks and use classical attacks such as linear, differential, and algebraic cryptanalysis. With this knowledge, the attacker can exploit any weaknesses in the mathematical description of the cryptogprahic primitive.

In use in our research we assume that the attacker also has temporary access to the cryptogprahic device in order to collect traces. In this case, with knowledge of the primitive and the traces the attacker can exploit power analysis attacks.

1.5 Overview

We will complete the introduction of this doctoral dissertation by stating our research questions and outlining chapter structure. This dissertation is attempting to study the security of lightweight cryptographic primitives. In the white box case we will

conduct power analysis attacks in order to quantify the resistance/susceptibility of the substitution parameter choice in lightweight symmetric block ciphers. In the black box case we will conduct a case study which details our attempts at breaking a lightweight cryptogprahic scheme with no knowledge of the scheme/implementation.

1.5.1 Research Questions

1. **How well do the current metrics quantify power analysis leakage?** In literature there exist power analysis metrics that claim to quantify how much a cryptographic implementation will leak without having to perform the actual power analysis attacks. There are many papers that show how to structure the parameters using these metrics to have the lowest possible metric score, but not many papers actually compare metrics with experimental results. In this dissertation, we will compare the metric results with actual experimental results and compare the metrics with the results to show how well the metrics quantify susceptibility/resistance to power analysis attacks.

2. **Given some binary cryptographic S-Boxes from literature which are more susceptible/resistant** We will compare some S-Boxes used in lightweight cryptology in the power analysis setting to measure the resistance/susceptibility of these S-Boxes

3. **Can we generalize cryptographic ciphers and metrics to work on other prime power and not just the binary case?** Like the works which took the Diffie-Hellman key exchange from integers to their elliptic curve equivalences (Menezes and Vanstone, 1993) we plan on giving a generalized construction of block ciphers to prime powers. In these constructions, we also generalize the

metrics to work in the case of prime powers, and attempt to generalize other mathematical proprieties such as equivalence classes. We also prepare a ternary case study and compare the ternary case study to the classical binary versions.

4. **How well correlated are the generalized cryptographic metrics to power analysis resistance/susceptibility** In the binary setting there are several metrics in literature that show a S-Boxes resistance/susceptibility to power analysis attacks. In the generalized setting how correlated are the metrics related to power analysis resistance.

5. **Do mathematical equivalence classes preserve metrics in the generalized setting** In the binary setting some equivalence classes (Affine, Permutation XOR) transforms preserve S-Box metrics. Given generalized metrics and transforms are metrics preserved under cryptographic transforms?

6. **Given an implementation of a prime powered non binary base implemented in hardware, will power analysis attacks work and if so, how well do the attacks work?** We will investigate whether or not power analysis attacks will work in a generalized block cipher scheme implemented in base 2 C code compiled for micro-controllers.

7. **Do different cryptographic implementations of odd prime powers leak less/more?** In modern SPN cryptology there is a clear way of creating a binary implementation of a binary cipher. In implementing a cryptographic scheme in an odd prime power there are many ways to code the prime powered scheme in binary hardware/software. As power analysis are attacks on cryptographic implementations, we will show that the process of representing the cryptographic

odd prime powered scheme in hardware/software will have a relationship of power analysis resistance/susceptibility.

8. **Given a cryptographic implementation of an unknown algorithm how hard is it to model a black box attack to attempt to break the encryption of the unknown algorithm?** While this will not be the same in every case, we were provided with hardware that performs encryption/decryption but not provided any algorithm or source code conveying how the algorithm works. We will investigate if we can figure out how the provided is computing encryption/decryption and attempt to mathematically model and break this unknown cryptographic scheme/protocol.

1.5.2 About This Doctoral Dissertation

This work is divided into 6 chapters in which 3 chapters detail our research and 3 other supplementary chapters for introduction, background, and conclusion. Chapter 1 gave in introduction of what our work will be on and why it is important. Chapter 2 will give us background information and definitions that will be used in this dissertation. Chapter 3 will detail our work in power analysis in the binary setting in which we compare metric values with experimental results. Chapter 4 will state our generalization of block ciphers in show details on our implementation on our ternary case study and our power analysis of the ternary case study. Chapter 5 will go into details on our cryptanalysis of an unknown algorithm. Lastly chapter 6 will provide some concluding statements of our doctoral dissertation.

CHAPTER 2:

BACKGROUND

In this chapter we will be stating background material that will be used throughout this dissertation. Information in this chapter will include but is not limited to: Substitution Permutation Network (SPN) block ciphers, Side Channel Attacks (SCA), mathematical equivalence classes, and S-Box metrics. This chapter will also state the current research done in these areas.

2.1 Substitution Permutation Network Ciphers

This section will give details on what a Substitution Permutation Network (SPN) block cipher as well as detail the GIFT64 cryptographic primitive scheme that will be used in our research.

A SPN is an iterated block cipher in which a plaintext and key are required as input, and over many rounds the plaintext is converted into a ciphertext. During each round, the output of the previous round is used as input to the current round along with a round keys derived from the private key. Crucial to the cipher are the round functions which are a composition of substitution, mixing, and introducing key content into the cipher so that the cipher has the proprieties of confusion and diffusion according to Shannon's principal (Shannon, 1948). Examples of such ciphers are the seminal work of Advanced Encryption Standard (AES) (Daemen and Rijmen, 2001),

which is still the standard for symmetric encryption, and a series of other ciphers that have lightweight proprieties such as PRESENT (Bogdanov et al., 2007), GIFT (Banik et al., 2017) , PICCOLO (Shibutani et al., 2011) , and many others.

The SPN cipher GIFT is a new and upcoming SPN cipher that is the cryptographic primitive for multiple round 2 candidates for NIST's lightweight standard (Banik et al., 2020, 2019) and a LWC finalist GIFT-COFB. All lightweight cryptographic candidates attempt to balance size, speed, and security for devices with resource constraints such as cyber-physical devices.

During each round in GIFT, there are sub-round functions known as S-Box, P-Layer, and the add round key. The GIFT primitive is split into two primitives known as GIFT64 and GIFT128, in which the number denotes the block size of the primitive. Both versions of GIFT takes in a 128-bit key for use in the encryption/decryption process. Also, both primitives have a key scheduling process that takes as input the private key and produces round keys. In GIFT64 the key scheduler takes as input the 128-bit private key and creates 28 64-bit round keys. Figure 2.1 below shows the structure of GIFT64.

The S-Box of both GIFT cryptographic primitives is a 4-bit to 4-bit invertible function and is defined in Table 2.1.

Table 2.1: GIFT S-Box Specification

Input	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Output	1	A	4	C	6	F	3	9	2	D	B	7	5	0	8	E

In GIFT the P-Layer function is a linear bit permutation defined in Table 2.2 below:

The P-Layer function acts as a mechanism that will provide diffusion in the cipher

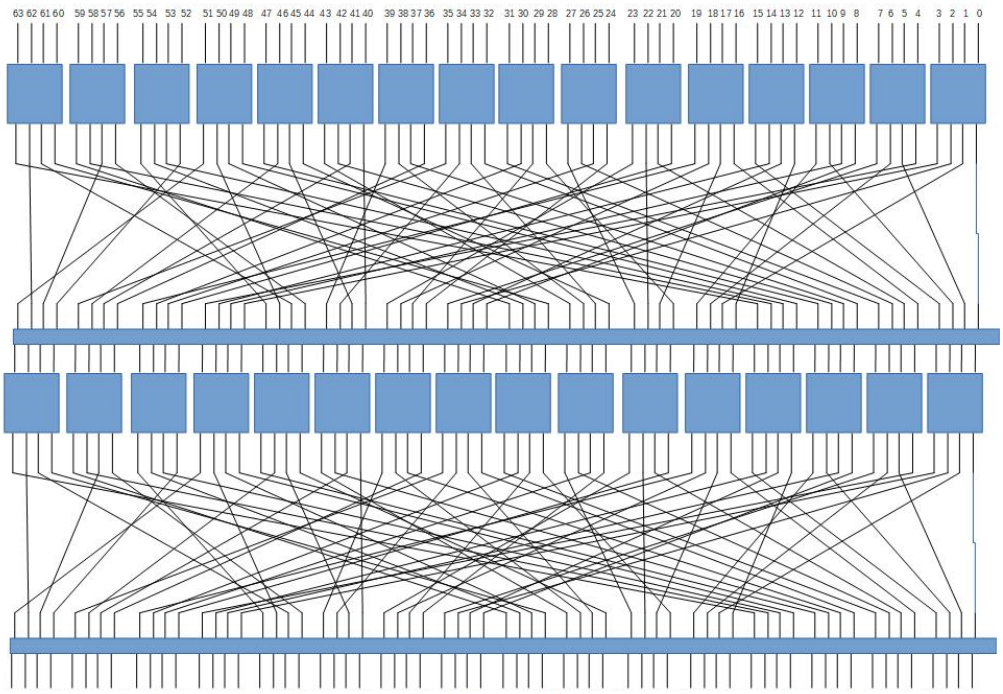

Figure 2.1: 2-rounds of GIFT-64

so even a small change in the plaintext input will lead to an avalanche effect of the ciphertext.

2.1.1 Key Types

In our research we use three types of keys being private, round, and sub-keys. The private key is the classical symmetric key that should be the end result that the attacker is trying to recover.

Round keys are created from the private key through the use of the key scheduler. If the attacker has enough round keys they should be able to recover the private key. In some cases such as AES-128 the first round key is equal to the private key. In cases such as GIFT64, the attacker needs four sequential round keys to recover the private key.

Table 2.2: GIFT64 - P-Layer

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
P(i)	0	17	34	51	48	1	18	35	32	49	2	19	16	33	50	3
i	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
P(i)	4	21	38	55	52	5	22	39	36	53	6	23	20	37	54	7
i	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
P(i)	8	25	42	59	56	9	26	43	40	57	10	27	24	41	58	11
i	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
P(i)	12	29	46	63	60	13	30	47	44	61	14	31	28	45	62	15

Lastly sub-keys are sub-quantities of the round key that are the same size as the input/output of the S-Box layer. One can view the round key to be the concatenation of several sub-keys. During power analysis attacks, the attacker computes predictions on each sub-key and uses those quantities to produce round keys and in turn is able to produce the private key.

2.2 Side Channel Attacks

Since the seminal work by Kocher in (Kocher et al., 1999), Side Channel Attack (SCA) has been a threat to block ciphers. SCA works by attacking the implementation of a cryptographic algorithm instead of the actual mathematical structure. Some examples of SCA are timing attacks, power attacks, and electromagnetic attacks. For the work in this dissertation, we will focus on power attacks such as Differential Power Analysis (DPA) and Correlation Power Analysis (CPA) (Brier et al., 2004). In power analysis, the goal is to use the implementation side effects along with information such as plaintexts and ciphertexts to deduce secret information such as a private key. The side effect of the implementation running that we will use to recover the private key is the device's voltage draw while the device is performing encryption/decryption. An example of a voltage reading over time is shown in Figure 2.2 in which the x-

axis are time units and the y-axis are the voltage readings at the time units. The collection of voltage over time during the encryption process is called a trace or power trace. For use in our research the trace is collected during the S-Box operation of the cryptographic implementation for use in power analysis attacks.

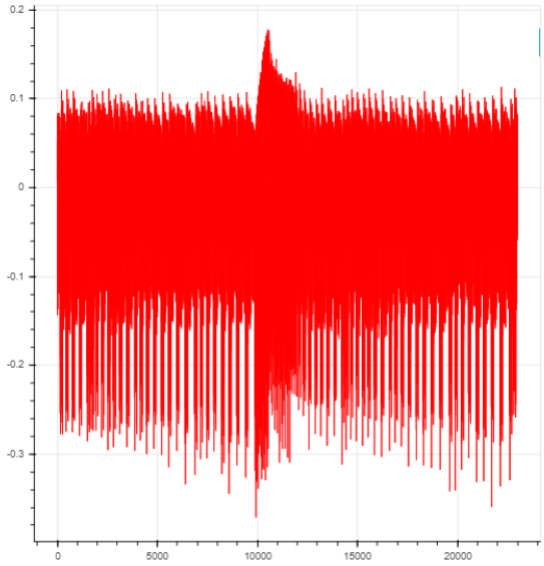

Figure 2.2: A Voltage Reading Over Time Example

2.2.1 Differential Power Analysis

While not used in our research, DPA attacks were a stepping stone to CPA attacks which we use in our research and has a sub-section later in this chapter. The DPA methodology works by attacking sub-keys in a given round of a block cipher. The attacker chooses a point of interest (POI) such as the output of the S-Box layer. The attacker also chooses a leaky bit that will be used in the DPA methodology. For each possible sub-key, the attacker computes a difference of means on the average of the traces in which the bit was 1 vs. the average of the traces in which the bit was 0. Upon analysis of the difference of means graph, if the correct sub-key was chosen

there will be a spike in a time instance of the difference of means trace. The attacker repeats this process for each sub-key in the round key and for as many round keys as needed to recover the private key.

While DPA was initially made for use with DES, it was later applied to other block ciphers such as AES, PRESENT, GIFT, and many others.

2.2.2 Correlation Power Analysis

The CPA methodology is a way of attacking a cryptographic implementation in order to attempt to compute the private key. CPA is an improvement on the DPA attack allowing to break the key successfully, faster, and using less data (Alioto et al., 2008; Lo et al., 2017). Two common leakage models are used to do this and they are Hamming Weight (HW) and Hamming Distance (HD). For use in our research, we will be using the HW model.

As input for the CPA attack, an attacker needs parings of plaintexts and their voltage trace over time units when the cryptographic device is performing the encryption operation. The traces can be interpreted as a graph in which the x-axis are time units and the y-axis are voltage values for that given time unit. All of the voltage traces need to be synchronized which means that the peaks/troughs of the trace occur on the same time instances.

The CPA methodology follows the following steps:

- State Point of Interest (POI)
- Capture plaintext/voltages
- Produce hypothetical intermediary values
- Compute estimated power draw from intermediary values

- Conduct analysis

Point of Interest

When attacking a cryptographic implementation for use with SCA, one needs to define what part of the implementation they will be attacking. It is common in literature to attack the output of the S-Box function (Durvaux and Standaert, 2016), but it is possible to attack other parts of a cryptographic cipher such as after a bit permutation. For the POI, one needs to consider that the attack needs to happen after key content is introduced during the encryption process. All states before key content is incorporated in the cipher is deterministic and computable without any key content. In AES, the add round key is the first operation so the first output of the S-Box can be attacked. For other ciphers such as GIFT, the add round key occurs at the end of each round. In order to attack the private key in GIFT, the output of the S-Boxes in rounds 2 and later need to be attacked.

Capturing Power Traces

After defining the POI, one will then need to collect power traces. In order to collect the traces, one will need a device that can collect this information such as an oscilloscope. For use in literature, one can put a trigger call to the oscilloscope to start/stop the capture in order to have automated and synchronized captures. An example of a voltage trace was shown previously in Figure 2.2. An example of capture setup, Figure 2.3 shows a CWLite that is a control board and oscilloscope and the Device Under Target (DUT) which is a XMEGA 8-bit micro-controller. The trace needs to have voltage reading when the POI is being computed in the cryptogprahic

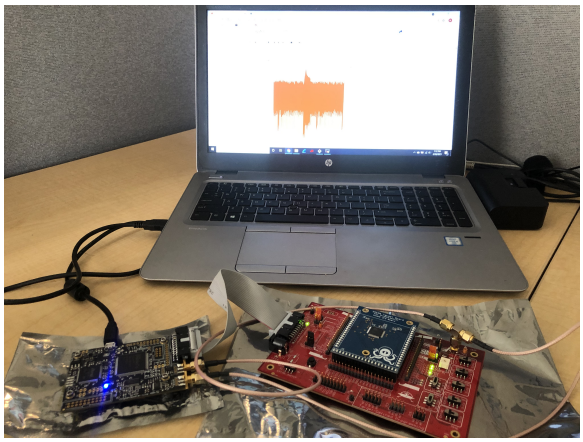

Figure 2.3: An Execution Environment

implementation.

Computing Intermediary Values

After acquiring the plaintexts the attacker will need to compute intermediary values based upon a sub-key guess and repeat the process for each sub-key. In AES, the S-Box outputs are 8-bit values so the sub-keys are 8-bits long. However, in GIFT the S-Box and sub-keys are 4-bits long. For each sub-key, one will compute the intermediary value of the output of the S-Box given that sub-key value. For each possible sub-key and plaintext pair the attacker computes the output of the S-Box after the sub-key has been applied using the XOR instruction.

Computing Hypothetical Power Consumption

Having the intermediary values, one needs to compute the hypothetical power consumption based upon these values. The two common ways of doing this is the Hamming weight HW and Hamming distance HD models. We consider the HW model which is just the number of 1 bits in the binary representation of a given value. For

each of the intermediary values, one will compute the Hamming weight and the power consumption value that are grouped as a pair of values. These values are the hypothetical power consumption which is the output of the HW model and the actual power consumption which is the voltage trace.

Analysis

Lastly, we will have to compare the computed hypothetical power consumption values with the voltage traces. Our comparison is the computation of correlation values to see which of the sub-keys is likely the correct sub-key. In order to do this, we will use Pearson's correlation coefficient (Freedman et al., 2007). The inputs to the correlation coefficient formula will contain two lists. The first list are the real voltage values R. The second list is the other being the hypothetical 'guessed' values G based upon the HW model.

$$\frac{\sum(R_i - R_{avg}) * (G_i - G_{avg})}{\sqrt{\sum(R_i - R_{avg})^2 * \sum(G_i - G_{avg})^2}} \tag{2.1}$$

Because each voltage trace has many voltage readings, the attacker will compute a correlation value for each of the time instances of the trace and compute the correlation for each time instance with the hypothetical values. After doing this for each time instance the attacker takes the maximum correlation for each possible time instance and that is the correlation value for that given possible sub-key. This process is repeated for each possible sub-key guess. The sub-key with the maximum correlation value is the predicted sub-key. This process is repeated for all sub-keys in the round key and for as many round keys as needed.

In AES-128, one round key is enough to recover the private key (the first round

key is the private key), however, in other block ciphers such as GIFT-64, four sequential round keys will be needed to recover the private key. The attacker starts with attacking the first round key and after the first round key is recovered the attacker uses the first round key and another set of pools relating to the second round key to attack the second round key. A similar process is computed for the third and fourth round keys.

When conducting a CPA attack the success rate (SR) indicates if an attack successfully recovered the key used by the cryptographic algorithm running on the DUT. For our uses, we only consider first order attacks. The Success Rate (SR) of CPA attack, defined in (Biryukov et al., 2016), is denoted as

$$\text{SR} = \begin{cases} 1, \text{If derived key is the correct key} \\ 0, \text{Otherwise} \end{cases}$$

For use in our research, we consider the mean SR denoted as a percentage as the average SR.

2.2.3 Side Channel Attack Mitigations

While there is research in SCA attacks on unprotected implementations, such as our work in (Unger et al., 2021), there exist mitigations that give rise to protected implementations against SCA attacks. While this is not part of our research, we wish to state the current state in protected cryptogprahic implementations against SCA

methodologies.

One approach is to add in NOP (No-Operation) instructions to de-synchronize the power traces. While this will initially stop the attack from working, if the attacker knows this mitigation has been applied he/she can re-synchronize the traces and conduct the attack.

Another method of defense is to add in dual-rail logic that attempts to make every instruction call have the same power draw (Sundaresan et al., 2008; Popp et al., 2007).

Lastly there is the countermeasure of masking such as Threshold Implementation (TI). There are several work of 'simple' masking (Golić and Tymen, 2002; Standaert et al., 2005) and TI masking (Bilgin et al., 2014b; De Cnudde et al., 2015). As we use the GIFT cryptographic scheme, there also exist attacks and countermeasures that were implemented in GIFT in the works of (Satheesh and Shanmugam, 2018; Jati et al., 2019; Zhang et al., 2021).

2.3 Equivalence Classes

This section will state different transforms that exist on S-Boxes that give rise to mathematical equivalence classes. For the purposes of our research, we will note three kinds of transforms: Affine transforms, Permutation XOR transforms, and XOR transforms. For the use or our research we mainly consider affine and permutation XOR transforms that are used to make Affine Equivalences (AE) and Permutation Equivalences (PE). Transforms are used in S-Box analysis and generations such as the work of (Biryukov et al., 2003; Canteaut and Roué, 2015).

XOR transform Given a n-bit inevitable S-Box, a XOR transform on S creating a new S-Box S' is shown below in which c and d both to be n-bit vectors and the

operation $\oplus$ to be XOR.

$$S'(x) = S(x \oplus c) \oplus d \tag{2.2}$$

In the XOR transform there are 2^{2n} possible transforms, but not all are always unique.

Permutation XOR Transform Permutation XOR transform is defined below given which A and B are nxn permutation matrices while c and d are n-bit vectors.

$$S'(x) = B(S(A(x \oplus c))) \oplus d \tag{2.3}$$

We note for permutation XOR transforms there are $(n!)^2 * 2^{2n}$ possible transforms but not all are necessarily unique.

Affine Transform Lastly the Affine transform is the last transform we will consider and is defined below in which the matrices A and B are nxn invertible matrices and c and d are n-bit vectors

$$S'(x) = B(S(A(x \oplus c))) \oplus d \tag{2.4}$$

It differs from the permutation XOR transform as the matrices A,B can be chosen from and binary inevitable matrix instead of just the permutation matrices.

2.4 S-Box Metrics

This section will state some S-Box metrics that are used to compare the security of one S-Box to another. While there are many S-Box metrics, for use in this doctoral thesis we will focus on metrics that have correlation with power analysis attacks. The goal of S-Box metrics for SCA is to measure resistance for SCA without having to compute the mean success rate. Some examples of S-Box metrics for SCA resistance

are Non-Linearity (NL) (Matsui, 1993), Transparency Order (TO) (Prouff, 2005), Revisited Transparency Order (RTO) (Li et al., 2020), Signal to Noise Ratio (SNR) and Differential Power Analysis Signal to Noise Ration (DPA-SNR) (Guilley et al., 2004). There exists literature on computing S-Boxes with optimal metric values for these metrics but not much work has been done comparing the metric values with actual experimental data.

2.4.1 Mathematical Background

This sub-section will be detailing building blocks that will be used in the S-Box metrics.

We denote "+" as the addition of integers in $\mathbb{Z}$ and " $\oplus$ " to be addition mod 2 also known as the XOR operation. Given a pair of vectors $a = (a_0, a_1, ..., a_{n-1)}$ and $b = (b_0, b_1, ..., b_{n-1})$ in $\mathbb{F}_2^n$ the product $a \cdot b$ is defined as

$$\sum_{i=0}^{n-1}(a_i * b_i) \tag{2.5}$$

We denote S-Boxes for use in Substitution Permutation Network (SPN) ciphers are n-bit to m-bit functions denoted as $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$. In most cases for our research we will consider the case that $n = m$.

In a n-bit to n-bit S-Box we consider the S-Box to be made up of boolean functions in which the output of an S-Box $F(x)$ can be viewed as the concatenation of the output of the boolean functions that make up the S-Box. Given '||' is the concatenation operation, we describe an S-Box as a concatenation of boolean functions below in

which each boolean function is of the form $\mathbb{F}_2^n \to \mathbb{F}_2$:

$$F(x) = F_1(x)||F_2(x)||...||F_n(x) \tag{2.6}$$

We define that the discrete derivative of a function F with input $a \in \mathbb{F}_2^n$ is a function of the form

$$D_a F : \mathbb{F}_2^n \to \mathbb{F}_2^m \tag{2.7}$$

in which it is defined by the formula

$$D_a F(x) = F(x) \oplus F(x \oplus a) \tag{2.8}$$

The function $W_f(u, v)$ is called a Walsh transform on the function F and is defined as

$$W_F(u, v) = \sum_{x \in \mathbb{F}_2^n} (-1)^{v \cdot F(x) + u \cdot x} \tag{2.9}$$

A generalization of the Walsh transform of a discrete derivative, the cross correlation spectrum is defined as

$$C_{f_1, f_2}(a) = \sum_{x \in \mathbb{F}_2^n} (-1)^{f_1(x) \oplus f_2(x \oplus a)} \tag{2.10}$$

2.4.2 S-Box Metric Definitions

In this sub-section, we will be defining the theoretical metrics that our research or literature show an impact on SCA resistance. While there is not a proof of the relationship between these metrics and SCA resistance, the relationship is shown

through experimental results in literature.

Non Linearity (NL) was defined in (Matsui, 1993) as a metric to measure resistance to linear and differential cryptanalysis but has also shown to be a metric in relationship to SCA. The higher the NL value is, the more resistant a cipher is to linear and differental cryptanalysis but higher NL will also lead to more susceptibility to SCA attacks (Biryukov et al., 2016). The NL of a function F is defined as:

$$NL(F) = 2^{n-1} - \frac{1}{2} \max_{u \in \mathbb{F}_2^n, v \in \mathbb{F}_2^{n*}} |W_F(u, v)| \tag{2.11}$$

In several lightweight SPN ciphers, 4-bit to 4-bit S-Boxes are used. The possible NL values for 4-bit to 4-bit invertible S-Boxes are 0, 2, and 4.

Transparency Order (TO) is another metric introduced in the work of (Prouff, 2005). The Transparency Order (TO) of a function F is defined as

$$TO(F) = \max_{\beta \in \mathbb{F}_2^n} \left(|n - 2H(\beta)| - \frac{1}{2^{2n} - 2^n} \sum_{a \in \mathbb{F}_2^{n*}} | \sum_{v \in \mathbb{F}_2^n, H(v)=1} (-1)^{v \cdot \beta} W_{D_a F}(0, v)| \right) \tag{2.12}$$

In the literature they state that the higher the TO value the more susceptible a S-Box is to SCA (Prouff, 2005). In later works, it was revealed that there are flaws in the definitions in the classical TO and lead to updated definitions of TO (Chakraborty et al., 2017). At the time of writing this dissertation, the newest version of TO is the definition of Revisited Transparency Order (RTO) in the work of (Li et al., 2020).

The Revisited Transparency Order (RTO) is an updated version of TO that fixes

some of the mistakes in the definition and given a function F is defined as:

$$RTO(F) = \max_{\beta \in \mathbb{F}_2^n} \left(n - \frac{1}{2^{2n} - 2^n} \sum_{a \in \mathbb{F}_2^{n*}} | \sum_{j=1}^{n} \sum_{i=1}^{n} (-1)^{\beta_i \oplus \beta_j} C_{F_i, F_j}(a) | \right) \tag{2.13}$$

Similar to TO, the results of the RTO metric are if the output is higher the S-Box should be more susceptible to SCA power analysis attacks.

Signal to Noise Ratio (SNR), (Ziemer and Tranter, 2014) is a probabilistic measurement of signal divided by noise in a cryptographic implementation. The quotient is defined as:

$$SNR = \frac{Var(Signal)}{Var(Noise)} \tag{2.14}$$

Commonly expressed in the units of decibels as $20log(SNR)$. The higher the SNR the stronger the signal and information is relative to the noise or distortion.

Differential Power Analysis Signal to Noise Ratio (DPA-SNR) (Guilley et al., 2004) is another metric that states to measure resistance to cryptanalysis and given a function F is defined as:

$$DPA - SNR(F) = n2^{2n} \left(\sum_{a \in \mathbb{F}_2^n} \left(\sum_{i=0}^{n-1} \left(\sum_{x \in \mathbb{F}_2^n} (-1)^{F_i(x) + x \cdot a} \right) \right)^4 \right)^{(-1/2)} \tag{2.15}$$

The higher the DPA-SNR value is the more resistant the cipher is to classical cryptanalysis leading to more susceptibility to SCA methodologies.

The confusion coefficient (CC) is the last metric that we will be denoting (Fei et al., 2012; Picek et al., 2014b). While this metric is not as widely used compared to the others it still has a following in literature.

Confusion Coefficient Function

$$K(K_a, K_b) = \frac{\sum_{pt \in \mathbb{F}_p^n}((HW(S(pt \oplus K_z)) - HW(S(K_b \oplus pt)))^2)}{2^n} \tag{2.16}$$

Confusion Coefficient Metric (CC) For all K_a, K_b in the sub-key space the variance of all possible outputs of the K function are computed and that is called the confusion coefficient variance.

CHAPTER 3:

CORRELATION POWER ANALYSIS

This chapter will focus on our research into CPA attacks on lightweight SPN primitives. We will analyze the security of the GIFT cryptographic primitive, more specifically analyzing the security of the S-Box layer. For our research we will use the 64-bit version of GIFT and fix all of the parameters except for the S-Box sub-round function. This chapter will utilize techniques in white box cryptanalysis in order to use knowledge of a cryptogprahic scheme/implementation along with plaintext/ciphertext pairs and power traces in order to attempt recovery of the private key. We will then swap our varying S-Boxes to analyze the mean success rate of CPA attacks and analyze the mathematical structure of the varying S-Boxes used in our research.

3.1 Summary

In our research we gather existing S-Boxes and create new ones that have varying mathematical proprieties. We consider some S-Boxes from literature such as GIFT (Banik et al., 2017), PICCOLO (Shibutani et al., 2011), and PRESENT (Bogdanov et al., 2007) as well as other S-Boxes we construct. Also in our research are groups of S-Boxes that use the Bad Output Good Input (BOGI) properties introduced in GIFT (Banik et al., 2017) in the work of (Kim et al., 2020) and we sample two groups of 8 S-Boxes in which the S-Boxes groups are members of the same affine class. As the

S-Boxes from (Kim et al., 2020) lead to similar results and metric scores, we show 1 representative from the affine class and note the S-Boxes as S_5 and S_6. Some of the S-Boxes we analyze, such as linear S-Boxes with NL = 0 should not be used in practice. Even though they might be more resistant to power analysis attacks they are not secure to classical cryptanalysis techniques and are considered weak and only used as comparison for metrics in our research.

A sample of some S-Boxes analyzed are shown in Table 3.1 and their metric scores in Table 3.2

Table 3.1: S-Boxes

Input	GIFT	PICCOLO	PRESENT	S_1	S_2	S_3	S_4	S_5	S_6
0	1	E	C	0	5	0	1	3	C
1	A	4	5	E	1	F	2	E	6
2	4	B	6	7	7	E	3	C	1
3	C	2	B	6	6	D	4	0	8
4	6	3	9	4	4	C	5	8	2
5	F	8	0	5	0	B	6	D	9
6	3	0	A	2	2	A	7	5	E
7	9	9	D	1	E	9	8	2	5
8	2	1	3	3	3	8	9	1	D
9	D	A	E	F	F	7	A	4	3
A	B	7	F	A	B	6	B	F	B
B	7	F	8	B	A	5	C	9	4
C	5	6	4	8	8	4	D	6	0
D	0	C	7	9	9	3	E	B	F
E	8	5	1	C	C	2	F	A	7
F	E	D	2	D	D	1	0	7	A

The goal in our research is to compare/contrast the metric scores of the S-Boxes with the experimental results of the mean SR computations. In doing so we analyze

Table 3.2: S-Boxes with computed metric values

S-Box	NonLinearity	SNR	DPA-SNR	TO	RTO	CC
PICCOLO	4	39.401	3.108	3.666	3.333	0.158
GIFT	4	39.348	2.399	3.466	3.066	0.459
PRESENT	4	34.665	2.129	3.533	3.266	0.660
S_2	2	39.968	2.946	3.4	3.266	0.208
S_4	0	39.252	2.484	2.933	2.933	0.409
S_1	0	38.582	2.579	3.266	3.133	0.359
S_3	0	34.148	2.484	2.933	2.933	0.409
S_5	4	33.122	2.399	3.467	3.067	0.459
S_6	4	41.004	2.579	3.333	3	0.359

which of the metrics has a higher indication of resistance. For use in our research we execute the CPA attack many times with different sets of plaintext and voltage arrays.

Experiment An experiment is a collection of CPA attack Success Rates (SR), with the the input to the experiment being a pool of plaintext/voltage array pairs, a threshold cap, and the known private key. An experiment is defined as a iterated loop in which each during every iteration we add a plaintext/voltage array pair to the internal data structure as input to the CPA attack. After the addition of the pair, the CPA attack is executed using that data-set. The SR of the CPA attack is stored along with the count of plaintext/voltage array pairs used to perform the attack. The experiment continues until either the size of the data-set reaches the threshold cap or until a success limit of 5 consecutive successful CPA attacks is observed. In our study, we used a threshold cap of 150 iterations, and pool of plaintext/voltage array pairs containing 2,000 entries. Example pseudocode is shown in Algorithm 1 :

The success limit was implemented in part to to speed up computation. We chose the constant 5 to be a success limit based upon early experimental data, but the

Algorithm 1: Experiment Pseudocode

```
Result: Success rate for each trace count
count = 0;
successCount = 0;
list initialized;
results structure initialized;
while count < Threshold do
    Add random plaintext-voltage array pair to list;
    Conduct CPA attack using list;
    count++;
    results[count] = CPA Success Rate (1/0);
    if Successful then
        successCount++;
        if successCount == 5 then
            Mark remaining results successful;
            return results;
        end
    end
    else
        successCount = 0;
    end
end
return results;
```

best parameter to ensure success stability is an open question. In our experimental studies, when 5 sequential CPA successes occur, all sequential execution of the CPA attack with additional plaintext/trace pairs were also successful.

A threshold cap of 150 was chosen based upon trial and error. The goal was to chose a cap which would halt execution of an experiment, but also allow each experiment to capture the full progression to a 100% mean success rate. Each of our S-box experiments achieved 100% mean success rate before the reaching the 150^{th} iteration.

The output of an experiment is a set of ordered pairs (x, y) stored in a *Results*

array. The 'x' value is the size of the data-set used for the CPA attack and the 'y' value is either 0 or 1 depending on if the CPA attack was successful or not. This means that the *Results* from an experiment is an array in which the index is the trace/count size and the element in the array is the mean SR.

Trial. A trial is a collection of Result arrays from many experiments. In our study we consider a trial to be a collection of 100 experiments. The output of a trial are ordered pairs similar to the output of the experiment, but the 'y' values hold the mean success rate of the 100 executions of the experiments. The trial algorithm is shown in Algorithm 2.

Algorithm 2: Trial Pseudocode

```
Result: Mean Success Rate for Each Trace Count
count = 0;
results structure initialized;
while count < 100 do
    Execute Experiment;
    Store results of experiment in the results structure;
    count++;
end
Average the results of the experiments for each trace count;
Return the average mean success rates;
```

For our analysis we chose several S-Boxes with varying values for the metrics presented previously and executed at minimum 1 trial for each S-Box.

3.2 Equivalence Classes

As part of our research we show mathematical proofs on equivalence classes given the mathematical background given in the Background chapter. For some metrics, AE and PE transforms retain the metric values of S-Boxes. In this section we will mathematically show that some metrics scores are preserved.

3.2.1 Revisited Transparency Order Preservation

Recall that cross correlation is defined below as shown in the work of (Li et al., 2020).

$$C_{f_1,f_2}(u) = \sum_{x \in \mathbb{F}_2^n} \big(-1\big)^{f_1(x) \oplus f_2(x \oplus u)} \tag{3.1}$$

And RTO is defined as

$$RTO(F) = \max_{\beta \in \mathbb{F}_2^n} \left(n - \frac{1}{2^{2n} - 2^n} \sum_{a \in \mathbb{F}_2^{n*}} |\sum_{j=1}^{n} \sum_{i=1}^{n} (-1)^{\beta_i \oplus \beta_j} C_{F_i,F_j}(a)| \right) \tag{3.2}$$

We consider the affine transformation in which A and B are both inevitable binary nxn matrices and d, e are elements in $\mathbb{F}_2^n$.

Let S be our starting S-Box and T is defined such that

$$T(x) = B(S(A(x) \oplus d)) \oplus e \tag{3.3}$$

We note a proof shown in (Li et al., 2020) that states RTO is preserved under Permutation XOR transformation when the transformation matrices are the identity. Assume that T is a PE transformation of S, we then have $RTO(S) = RTO(T)$.

Proof

Let

$$RTO(T) = \max_{\beta \in \mathbb{F}_2^n} \left(n - \frac{Q_t}{2^{2n} - 2^n} \right) \tag{3.4}$$

where

$$Q_T = \sum_{a \in \mathbb{F}_2^{n*}} \left| \sum_{j=1}^{n} \sum_{i=1}^{n} (-1)^{\beta_i \oplus \beta_j} C_{T_i,T_j}(a) \right| \tag{3.5}$$

By expanding and unpacking the cross correlation function we have:

$$Q_T = \sum_{a \in \mathbb{F}_2^{n*}} \left| \sum_{j=1}^{n} \sum_{i=1}^{n} (-1)^{\beta_i \oplus \beta_j} \sum_{x \in \mathbb{F}_2^n} (-1)^{(B(S(A(x) \oplus d)) \oplus e)_i \oplus (B(S(A(x \oplus a) \oplus d)) \oplus e)_j} \right| \tag{3.6}$$

We extract the e values out of the inner exponent:

$$Q_T = \sum_{a \in \mathbb{F}_2^{n*}} \left| \sum_{j=1}^{n} \sum_{i=1}^{n} (-1)^{\beta_i \oplus \beta_j} \sum_{x \in \mathbb{F}_2^n} (-1)^{(B(S(A(x) \oplus d)))_i \oplus e_i \oplus (B(S(A(x \oplus a) \oplus d)))_j \oplus e_j} \right| \tag{3.7}$$

We extract the e values from the inner sum using the laws of exponentiation

$$Q_T = \sum_{a \in \mathbb{F}_2^{n*}} \left| \sum_{j=1}^{n} \sum_{i=1}^{n} (-1)^{\beta_i \oplus \beta_j \oplus e_i \oplus e_j} \sum_{x \in \mathbb{F}_2^n} (-1)^{(B(S(A(x) \oplus d)))_i \oplus (B(S(A(x \oplus a) \oplus d)))_j} \right| \tag{3.8}$$

We the use the proprieties of reverse distribution under matrices to extract the B matrix

$$Q_T = \sum_{a \in \mathbb{F}_2^{n*}} \left| \sum_{j=1}^{n} \sum_{i=1}^{n} (-1)^{\beta_i \oplus \beta_j \oplus e_i \oplus e_j} \sum_{x \in \mathbb{F}_2^n} (-1)^{B((S_i(A(x) \oplus d) \oplus (S_j(A(x \oplus a)) \oplus d))} \right| \tag{3.9}$$

Let $y = A(x) \oplus d$.

We note that $A(x \oplus a) \oplus d = y \oplus A(a)$ that switches the iterator in the inner sum from x to y that results in.

$$Q_T = \sum_{a \in \mathbb{F}_2^{n*}} \left| \sum_{j=1}^{n} \sum_{i=1}^{n} (-1)^{\beta_i \oplus \beta_j \oplus e_i \oplus e_j} \sum_{y \in \mathbb{F}_2^n} (-1)^{B(S_i(y) \oplus S_j(y \oplus A(a)))} \right| \tag{3.10}$$

We let $u = A(a)$ and change the outside iterator from a to u

$$Q_T = \sum_{u \in \mathbb{F}_2^{n*}} \left| \sum_{j=1}^{n} \sum_{i=1}^{n} (-1)^{\beta_i \oplus \beta_j \oplus e_i \oplus e_j} \sum_{y \in \mathbb{F}_2^n} (-1)^{B(S_i(y) \oplus S_j(y \oplus u))} \right| \tag{3.11}$$

Recalling the definition of cross correlation we switch the $\sum$ notation to the C notation resulting in:

$$Q_T = \sum_{u \in \mathbb{F}_2^{n*}} \left| \sum_{j=1}^{n} \sum_{i=1}^{n} (-1)^{\beta_i \oplus \beta_j \oplus e_i \oplus e_j} C_{BS_i, BS_j}(u) \right| \tag{3.12}$$

We then create new $\beta_i' \beta_j'$ such that $\beta_i' = \beta_i \oplus e_i$ and $\beta_j' = \beta_j \oplus e_j$ that results in

$$Q_T = \sum_{u \in \mathbb{F}_2^{n*}} \left| \sum_{j=1}^{n} \sum_{i=1}^{n} (-1)^{\beta_i' \oplus \beta_j'} C_{BS_i, BS_j}(u) \right| \tag{3.13}$$

If we consider B to be the identity matrix we then have

$$Q_T = \sum_{u \in \mathbb{F}_2^{n*}} \left| \sum_{j=1}^{n} \sum_{i=1}^{n} (-1)^{\beta_i' \oplus \beta_j'} C_{S_i, S_j}(u) \right| \tag{3.14}$$

Which gives us the result that $RTO(S) = RTO(T)$

3.2.2 Transparency Order Preservation

Based upon the RTO preservation proof of equivalence classes by (Li et al., 2020) we show preservation under the classical transparency order in which TO is defined in the works of (Prouff, 2005). However, our results take Proof's initial idea and show that the metric score is preserved under Permutation Equivalence instead of a special case of PE shown in (Prouff, 2005).

We note that the definition of TO is:

$$TO(F) = \max_{\beta \in \mathbb{F}_2^n}(n - 2HW(\beta)) - \frac{1}{2^{2n} - 2^n} \sum_{a \in \mathbb{F}_2^{n*}} | \sum_{v \in \mathbb{F}_2^n; HW(v)=1} (-1)^{v*\beta} W_{D_aF}(0, v)|) \tag{3.15}$$

For the sake of clarity we will denote the TO formula as:

$$TO(F) = \max_{\beta \in \mathbb{F}_2^n}(n - 2HW(\beta)) - \frac{Q_F(\beta)}{2^{2n} - 2^n} \tag{3.16}$$

Where

$$Q_F(\beta) = \sum_{a \in \mathbb{F}_2^{n*}} | \sum_{v \in \mathbb{F}_2^n; HW(v)=1} (-1)^{v*\beta} W_{D_aF}(0, v)|) \tag{3.17}$$

We consider T to be a transformation on F such that:

$$T(x) = B(F(A(x \oplus d))) \oplus e \tag{3.18}$$

Giving us the equation for Q_T shown below:

$$Q_T(\beta) = \sum_{a \in \mathbb{F}_2^{n*}} | \sum_{v \in \mathbb{F}_2^n; HW(v)=1} (-1)^{v*\beta} W_{D_aF}(0, v)|) \tag{3.19}$$

We then expand and unpack the definition of the Walsh Transform of the discrete derivative.

$$Q_T(\beta) = \sum_{a \in \mathbb{F}_2^{n*}} | \sum_{v \in \mathbb{F}_2^n; HW(v)=1} (-1)^{v*\beta} \sum_{x \in \mathbb{F}_2^n} (-1)^{v*D_aT(x)}|) \tag{3.20}$$

$$Q_T(\beta) = \sum_{a \in \mathbb{F}_2^{n*}} | \sum_{v \in \mathbb{F}_2^n; HW(v)=1} (-1)^{v*\beta} \sum_{x \in \mathbb{F}_2^n} (-1)^{v*(T(x) \oplus T(x \oplus a)} |) \tag{3.21}$$

We then expand the T function to write it in terms of the transforms of F.

$$Q_T(\beta) = \sum_{a \in \mathbb{F}_2^{n*}} | \sum_{v \in \mathbb{F}_2^n; HW(v)=1} (-1)^{v*\beta} \sum_{x \in \mathbb{F}_2^n} (-1)^{v*(B(F(A(x \oplus d))) \oplus e \oplus B(F(A(x \oplus a \oplus d))) \oplus e)} |) \tag{3.22}$$

Because of the XOR operation and each element is it's own inverse we remove the $e \oplus e$.

$$Q_T(\beta) = \sum_{a \in \mathbb{F}_2^{n*}} | \sum_{v \in \mathbb{F}_2^n; HW(v)=1} (-1)^{v*\beta} \sum_{x \in \mathbb{F}_2^n} (-1)^{v*(B(F(A(x \oplus d))) \oplus B(F(A(x \oplus a \oplus d))))} |) \tag{3.23}$$

Because the B matrix is applied to both elements in the right exponent we can pull out the B matrix.

$$Q_T(\beta) = \sum_{a \in \mathbb{F}_2^{n*}} | \sum_{v \in \mathbb{F}_2^n; HW(v)=1} (-1)^{v*\beta} \sum_{x \in \mathbb{F}_2^n} (-1)^{v*(B(F(A(x \oplus d)) \oplus F(A(x \oplus a \oplus d))))} |) \tag{3.24}$$

Let $y = A(x \oplus d)$ also leading to the result $y \oplus A(a) = A(x \oplus a \oplus d)$ and by re-indexing x to y

$$Q_T(\beta) = \sum_{a \in \mathbb{F}_2^{n*}} | \sum_{v \in \mathbb{F}_2^n; HW(v)=1} (-1)^{v*\beta} \sum_{y \in \mathbb{F}_2^n} (-1)^{v*(B(F(y) \oplus F(y \oplus A(a))))} |) \tag{3.25}$$

We let $u = A(a)$ and re-index a leading to Q_T being:

$$Q_T(\beta) = \sum_{u \in \mathbb{F}_2^{n*}} | \sum_{v \in \mathbb{F}_2^n; HW(v)=1} (-1)^{v*\beta} \sum_{y \in \mathbb{F}_2^n} (-1)^{v*(B(F(y) \oplus F(y \oplus u)))} |) \tag{3.26}$$

We then re-pack the definition of the discrete derivative

$$Q_T(\beta) = \sum_{u \in \mathbb{F}_2^{n*}} | \sum_{v \in \mathbb{F}_2^n; HW(v)=1} (-1)^{v*\beta} \sum_{y \in \mathbb{F}_2^n} (-1)^{v*D_u BF(y)} |) \tag{3.27}$$

The re-pack the Walsh transform

$$Q_T(\beta) = \sum_{u \in \mathbb{F}_2^{n*}} | \sum_{v \in \mathbb{F}_2^n; HW(v)=1} (-1)^{v*\beta} W_{DuBF}(0, v) |) \tag{3.28}$$

If we consider B to the the identity matrix we have the proven the case of XOR equivalence. If we consider B to be a permutation matrix we can then re-index v and we show that TO is preserved under Permutation XOR equivalence (PE).

3.3 Experimental Results

We gathered data on the 8-bit XMEGA micro-controller. The data gathered were pairs of traces (voltage over time) and plaintext values for each S-Box used in our case study. For each S-Box, we computed theoretical metric scores from literature that are shown to have a connection with the success rate of side channel attacks in experimental results. We computed many executions of the CPA attack and used that to judge the mean success rate of the S-Boxes. Recall the output of the trials are graphs in which the x-axis is the give trace count as input to the CPA attack and the y-axis is the computed mean success rate of the CPA attack. The mean success rate computations are probabilistic meaning that each time re-execution the trial code the numeric outputs will be different, however statically speaking each executions are similar.

The results of the S-Boxes we sampled are below in Figure 3.1.

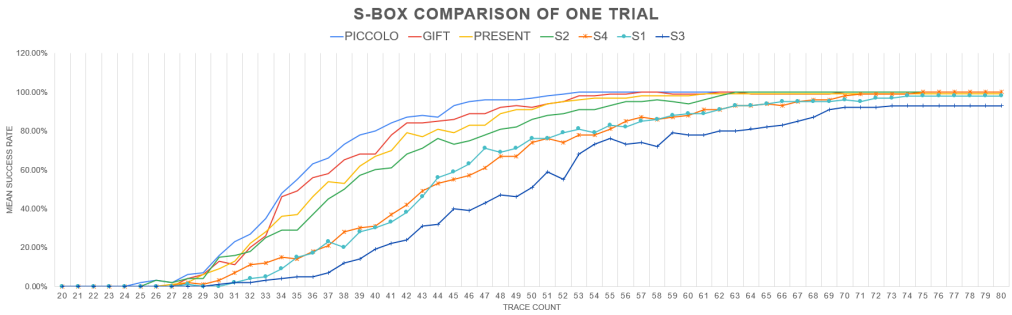

Figure 3.1: The mean success rate of a single trial of S-Boxes

We note that all CPA attacks on our S-Boxes ended up having a 100 % mean success rate given enough traces. To compare the S-Boxes, we compare the mean success rates of the S-Boxes before the 100% mean success rate mark. In Figure 3.1 we compare the trace range from 20 to 80 because before 20 traces no S-Box has a mean SR above 0% and at 80 traces most S-Boxes have a mean SR of 100% . Upon initial observation, we can easily see that some S-Boxes are more resistant to CPA attack and others are more susceptible. In Table 3.2 the S-Boxes are listed in order from most susceptible to CPA attack to most resistant. Note that the metric of non-linearity has clear correlation on how resistant/susceptible the S-Box is to CPA attack. In 4-bit S-Boxes there only exist 3 possible non-linearity values being the set {0,2,4} . Because of the distribution on Non-Linearity group together S-Boxes with the same non-linearity values in Figure 3.2 and Figure 3.3 and there corresponding metric scores in Table 3.3 and Table 3.4.

Table 3.3: S-Boxes computed metric values for NL=4

S-Box	NonLinearity	SNR	DPA-SNR	TO	RTO
PICCOLO	4	39.401	3.108	3.666	3.333
GIFT	4	39.348	2.399	3.466	3.066
PRESENT	4	34.665	2.129	3.533	3.266

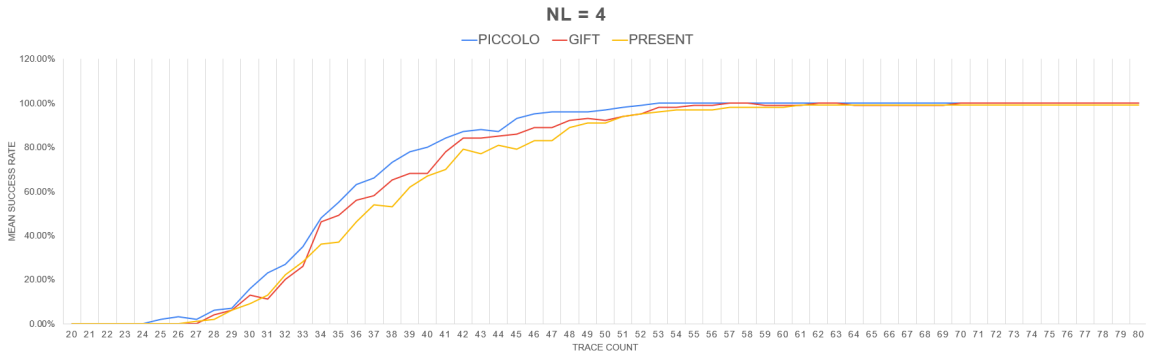

Figure 3.2: The mean success rate of a single trial of S-Boxes with NL=4

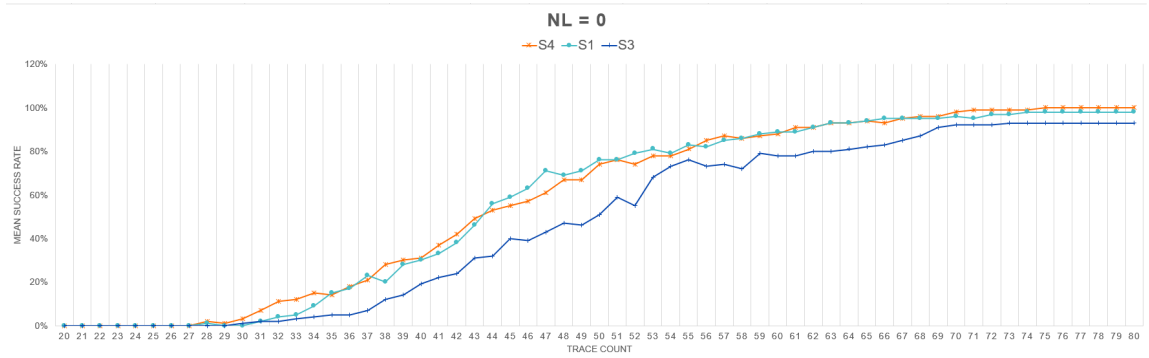

Figure 3.3: The mean success rate of a single trial of S-Boxes with NL=0

Table 3.4: S-Boxes computed metric values for NL=0

S-Box	NonLinearity	SNR	DPA-SNR	TO	RTO
S_4	0	39.252	2.484	2.933	2.933
S_1	0	38.582	2.579	3.266	3.133
S_3	0	34.148	2.484	2.933	2.933

Upon observation of the figures with NL grouping, we try and compare the other metric scores. In our case study, we use all varying metric values for non-linearity but for resistance to classical cryptanalysis such as differental and linear the literature focus on S-Boxes that are 'optimal' having requirements such as non-linearity = 4, degree = 3, no fixed points, and differential uniformity = 4.

The cryptographic scheme of GIFT (Banik et al., 2017) also introduced the con-

cept of bad input good output (BOGI), which is a property of the S-Box to add additional security to the 'optimal' S-Boxes. The work of (Kim et al., 2020) added more research on the BOGI propriety and added other samples of such S-Boxes besides the S-Box for the GIFT cryptographic scheme.

3.4 Correlation Power Analysis Conclusions

Upon analysis of the S-Boxes, we note that the metric of non-linearity has strong correlation with the mean success rate of the CPA attack and having lower non-linearity will lead to more resistance. However, because of classical cryptanalysis one is not always able to choose S-Boxes and to defend against the classical cryptanalysis attacks one might have to choose the higher non-linearity S-Boxes. We note that if non-linearity is 4 there is still variation in the mean success rate of the CPA attack. Analysing the metric scores of SNR, DPA-SNR, TO, and RTO, we note that there is some correlation but none of the metrics have perfect correlation with the mean success rate of the CPA attack.

CHAPTER 4:

GENERALIZED AND TERNARY CORRELATION POWER ANALYSIS

In this chapter, we will build our theoretical construction of a GIFT-Like SPN cipher under a varying prime field and show an application of the theoretical construction using a ternary base. Similar to our binary cryptanalysis, this chapter will utilize white box cryptanalysis techniques with knowledge of the scheme/implementation along with plaintext/ciphertext pairs and power traces to attempt to recover the private key.

4.1 Generalized Mathematical Background

In order to describe our theoretical construction, we first need to describe the building blocks we will be using. We denote the $\oplus_p$ operation to be a generalization of the XOR operation to work on elements in $\mathbb{F}_p^n$ in which we denote modular addition of the n-element vectors. In the binary case, the XOR operation is not distinguished between addition and subtraction modulo 2 because they are the same operation in base 2. In our theoretical constructions, we denote $\oplus_p$ to be the modular addition operation and $\ominus_p$ to be the modular subtraction operation.

We define δ to be a p^{th} root of unity that is described as a unit vector in the complex plane in which there is no positive integer less $a < p$ such that $\delta^a = 1$. Often

working under addition, we consider adding complex numbers are adding their real and imaginary parts such as considering two complex numbers $c_1 = r_1 + g_1 i$ and $c_2 = r_2 + g_2 i$ the sum $c_3 = c_1 + c_2$ is denoted as $c_3 = (r_1 + r_2) + (g_1 + g_2)i$.

We still use the $*$ notation as the dot product and now operates on the finite field $\mathbb{F}_p^n$

4.1.1 Generalized Walsh Transform

The Walsh Transform is a building block for many metrics in S-Boxes and is used in our case study. We generalize the Walsh transform to work in any prime power $\mathbb{F}_p^n$. We consider the value δ to be the first p^{th} root of unity. The generalized Walsh Transform is shown below:

$$W(u, v) \sum_{x \in \mathbb{F}_p^n} \left(\delta^{u \cdot x} * \delta^{v \cdot S(x)} \right) \tag{4.1}$$

4.1.2 Generalized Discrete Derivative

The discrete derivative is used in the computation of the classical transparency order as described in (Prouff, 2005). We take the binary definition of the discrete derivative and use the definition to work in a generalized setting. The discrete derivative is defined as follows:

$$D_a(S, x) = S(x \oplus_p a) \ominus_p S(x) \tag{4.2}$$

4.1.3 Generalized Cross Correlation of Discrete Function

The cross correlation function is used in the computation of the revisited transparency order that is shown in it's binary case in the work of (Li et al., 2020). For our uses, we generalized the cross correlation to the generalized setting as described in the

equation below:

$$C_{F_1,F_2}(u) = \sum_{x \in \mathbb{F}_p^n} (\delta)^{F_1(x) \oplus_p F_2(x \oplus_p u)} \tag{4.3}$$

4.2 Generalized S-Box Metrics

Similar to the work of (Biryukov et al., 2016), we choose S-Box metrics that have a known relationship with power analysis attacks in literature. After choosing such metrics, we generalize them to work in the setting of any prime power as compared to just powers of two. We then compare the metric values of the S-Boxes chosen in our case study with actual experimental data. The S-Box metrics chosen for use with our generalized and ternary case study are shown in this section.

4.2.1 Generalized Non-Linearity

In our work, we generalize the non linearity metric to be able to compute on any prime powered field. This metric was defined in the binary case in the work of (Matsui, 1993) to describe a ciphers resistance/susceptibility to linear cryptanalysis. We edited the equation to work under any prime powered field. This metric is defined below:

$$NL(F) = p^{n-1} - \left(\frac{1}{p} * \max_{u \in \mathbb{F}_p^n, v \in \mathbb{F}_p^{n*}} \left| W_F(u,v) \right| \right) \tag{4.4}$$

4.2.2 Generalized Differential Power Analysis Signal to Noise Ratio

DPA-SNR was a metric that was first noted in (Guilley et al., 2004) that is used in the computation of S-Box susceptibility/resistance under the DPA attacks. In our work, we edited it to a generalized setting and made some choices on the generalization

based upon our other generalizations in the case study. It is an open question on the choice of powers and when to take the absolute value in the case of generalizing DPA-SNR. Our generalization is shown below:

$$DPA - SNR(S) = n * p^{2n}\Bigg(\sum_{a\in\mathbb{F}_p^n}\Bigg|\Bigg(\sum_{i=0}^{n-1}\Bigg(\sum_{x\in\mathbb{F}_p^n}(\delta)^{S_i(x)+(x*a)}\Bigg)\Bigg)^4\Bigg|\Bigg)^{(-1/2)} \tag{4.5}$$

4.2.3 Generalized Transparency Order

The classical binary Transparency Order was introduced in the work of (Prouff, 2005). Transparency Order was one of the first metrics developed to compute the resistance/susceptibility to power side channel attacks. We edited their formula to work in any prime powered fields and our generalized definition is shown below given a S-Box S:

$$TO(S) = \max_{\beta\in\mathbb{F}_p^n}\Bigg(|n - 2HW(\beta)| - \frac{1}{p^{2n}-p^n}\sum_{a\in\mathbb{F}_p^{n*}}\Bigg|\sum_{v\in\mathbb{F}_p^n;HW(v)=1}(\delta)^{v*\beta}W_{DaS}(0,v)\Bigg|\Bigg) \tag{4.6}$$

4.2.4 Generalized Revisited Transparency Order

Following the work of (Prouff, 2005), there were many additions/improvements on the metric of Transparency Order including improved TO revisited TO and modified TO (Chakraborty et al., 2017). The metric Revisited Transparency Order (RTO)(Li et al., 2020) is the most recent improvement of TO at the time of writing this dissertation and is the newest of all S-Box metrics we analyze in this dissertation. We generalized the metric of RTO to work over any prime field and our generalization is stated below

given a S-Box S:

$$RTO(S) = \max_{\beta \in \mathbb{F}_p^n} \left(n - \frac{1}{p^{2n} - p^n} \sum_{a \in \mathbb{F}_p^{n*}} \left| \sum_{j=1}^{n} \sum_{i=1}^{n} (\delta)^{\beta_i \oplus_p \beta_j} * C_{S_i,S_j}(a) \right| \right) \tag{4.7}$$

4.2.5 Signal to Noise Ratio

We also use the classical SNR computation's for our S-Box analysis which is noted and the variance of the signal divided by the variance of the noise.

4.2.6 Generalized Confusion Coefficient

The works of (Fei et al., 2012; Picek et al., 2014b) use the confusion coefficient as a metric for S-Boxes. We generalized it for use in a prime powered S-Box and gathered data for our ternary case study. The generalized definition of CC is shown below:

Confusion Function

$$K(K_a, K_b) = \frac{\sum_{pt \in \mathbb{F}_p^n} ((HW(S(pt \oplus_p K_a)) - HW(S(K_b \oplus_p pt)))^2)}{p^n} \tag{4.8}$$

Similar to the binary version, the generalized version iterates the values K_a, K_b in the sub-key space and commute the variance of all the outputs of the K functions resulting in the CC metric score.

Generalized Confusion Coefficient For all K_a, K_b in the sub-key space the variance of all possible outputs of the K function are computed and that is called the confusion coefficient variance.

4.3 Generalized Equivalence Classes

Similar to the binary case, in the generalized setting we can divide up the set of all S-Boxes in $\mathbb{F}_p^n$ into classes that retain some cryptographic properties. We consider three classes of transforms being: modular addition, permutation modular addition, and affine. It is important to compute these classes as it retains some cryptographic properties, so when choosing an S-Box for use in a cryptographic scheme one can choose an S-Box from a 'good' class instead of searching the whole S-Box space for S-Box candidates.

4.3.1 Equivalence Transforms

In this sub-section, we will be stating the definition of the equivalence transforms that are used to generate classes. For each of the three transforms we are considering.

Modular Addition Transform

Consider an invertible S-Box S $\mathbb{F}_p^n \longrightarrow \mathbb{F}_p^n$ and two vectors $d, e \in \mathbb{F}_p^n$. The transform from S-Box S to S-Box T is shown below:

$$T(x) = S(x \oplus_p d) \oplus_p e \tag{4.9}$$

Permutation Modular Addition Transform

An extension of modular addition, permutation modular addition adds permutation matrices $A, B \in \mathbb{F}_2^{nxn}$ in which each row has a single 1 entry and the rest of the numbers in the row are 0 and similarly for the columns. The permutation modular

addition transform on S to T is shown below:

$$T(x) = B(S(A(x \oplus_p d))) \oplus_p e \tag{4.10}$$

We define this relationship as a Permutation Equivalence Class (PE)

Affine Transform

An extension of the permutation modular addition transform, the affine transform is the last transform we will be considering in our work. Similar to PE transform except instead of A, B being limited to permutation matrices, A, B can be any invertible matrix in $\mathbb{F}_p^{nxn}$. The affine transform is defined below:

$$T(x) = B(S(A(x \oplus_p d))) \oplus_p e \tag{4.11}$$

We define this relationship is a Affine Equivalence (AE)
We will now show an example of a S-Box transform:

Modular Addition Ternary Transform Example

In this example we will apply a modular addition transform in the ternary case. In this we consider 2 ternary digit values and the modular addition is over the two ternary digit values. Consider the S-Box below and consider the following ternary addition of vector d:

First we will show an example of the inner ternary modular addition.

$$S(x \oplus_3 d) \tag{4.12}$$

Table 4.1: S-Box Transform Example

x	00	01	02	10	11	12	20	21	22
S(x)	01	02	10	11	12	20	21	22	00

In which $d = 12$. The matrix notation is shown below:

$$S(x \oplus_3 d) = S\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 1 \\ 2 \end{bmatrix} \right) \tag{4.13}$$

For every input in $\mathbb{F}_3^2$ we will have to substitute the values in for x_1, x_2 and then apply the d vector and then the S-Box to create a look up table for the transformed S-Box. Let us start with $x = 00$:

$$S(00 \oplus_3 d) = S\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 \\ 2 \end{bmatrix} \right) \tag{4.14}$$

We compute the modular addition

$$S(0 \oplus_3 d) = S\left(\begin{bmatrix} 1 \\ 2 \end{bmatrix} \right) \tag{4.15}$$

We now apply the S-Box:

$$S(00 \oplus_3 d) = 20 \tag{4.16}$$

This process is then repeated for every possible input x and our example results in a transformed S-Box shown below:

Table 4.2: S-Box Ternary Modular Addition Transform Example

x	00	01	02	10	11	12	20	21	22
$S(x \oplus_3 d)$	20	11	12	00	21	22	10	01	02

The process of applying the matrices are similar. Let us consider a transform

$$S(A(x)) \tag{4.17}$$

In which

$$A = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{4.18}$$

Our equation for the transform is now:

$$S(A(x)) = S\left(\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \right) \tag{4.19}$$

Let us consider the case of $x = 12$

$$S(A(12)) = S\left(\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix} \right) \tag{4.20}$$

$$S(A(12)) = S\left(\begin{bmatrix} 2 \\ 1 \end{bmatrix} \right) \tag{4.21}$$

$$S(A(12)) = 22 \tag{4.22}$$

If we repeat this process for every possible x we get the transformed S-Box shown

in the table below:

Table 4.3: S-Box Ternary Permutation Transform Example

x	00	01	02	10	11	12	20	21	22
$S(A(x)$	01	11	21	02	12	22	10	20	00

For full transforms we apply the inner and outer transforms but the process of applying matrix and vector transforms are similar to the examples shown.

4.3.2 Properties of Equivalence Classes

In order to be a equivalence class, the transforms must hold the properties of reflexively, symmetry and transitivity. Suppose that a, b, c are S-Boxes and $\sim$ denotes a possible transform.

Reflexive Property

For any A in the class there exists a transform such that:

$$a \sim a \tag{4.23}$$

Symmetric Property

For any a, b in the equivalence class there exist $\sim_1, \sim_2$ such that

$$a \sim_1 b \tag{4.24}$$

and

$$b \sim_2 a \tag{4.25}$$

Transitive Property

Given three S-Boxes a, b, c if there exists

$$a \sim_1 b \tag{4.26}$$

and

$$b \sim_2 c \tag{4.27}$$

There exists a $\sim_3$ such that

$$a \sim_3 c \tag{4.28}$$

We consider our three equivalence classes as subsets of one another. We consider the AE class a union of many PE classes and similarly we can consider PE classes a union of many modular addition classes. Because of this property, if we prove any statement on AE classes that statement holds on the PE subsets and similarly, if we prove anything on PE classes that statement holds on the modular addition subsets. An example of this is the preservation of metric scores like non-linearity.

4.3.3 Generalized Equivalence Classes

In this section, we will be proving that our generalized transforms are indeed mathematical equivalence classes. To do this, we have to prove the proprieties of reflexively, symmetry and transitivity. After doing so, we will prove that the three equivalence classes can be divided up to subsets within the equivalence classes.

Reflexive Proof

In order to show the reflexive propriety, we need to show that there exists a transform in each of the categories that map an S-Box to itself.
In the case of modular addition if we consider the d, e vectors to be the zero vector we show that Modular Addition has the reflexive property.

$$T(x) = S(x \oplus_p d) \oplus e = S(x) \tag{4.29}$$

Similarly in the cases of permutation modular addition and affine, we consider the d, e vectors to be the zero vector and we consider the A, B matrices to be the identity matrix resulting in:

$$T(x) = B(S(A(x \oplus_p d))) \oplus_p e = B(S(A(x))) = S(x) \tag{4.30}$$

Symmetric Proof

In order to prove that symmetry exists in the transform, we have to show that if there exist a transform from the S-Box S to the S-Box T there is also another transform which maps T back to S. In the case of modular addition we have:

$$T(x) = S(x \oplus_p d) \oplus_p e \tag{4.31}$$

$$T(x) \ominus_p e = S(x \oplus_p d) \tag{4.32}$$

$$T(x \ominus_p d) \ominus_p e = S(x) \tag{4.33}$$

We can now take the 'negative' vector and denote them as d', e' and use them in modular addition as shown below:

$$T(x \oplus_p d') \oplus_p e' = S(x) \tag{4.34}$$

In the cases of permutation modular addition and affine we also have to consider the matrices A, B. The initial transform statement for them is shown below:

$$T(x) = B(S(A(x \oplus_p d))) \oplus_p e \tag{4.35}$$

That leads to

$$T(x) \ominus_p e = B(S(A(x \oplus_p d))) \tag{4.36}$$

We then apply the B^{-1} matrix to each side:

$$B^{-1}(T(x) \ominus_p e) = S(A(x \oplus_p d)) \tag{4.37}$$

Now we apply the A^{-1} matrix to the inner call of T and S

$$B^{-1}(T(A^{-1}(x)) \ominus_p e) = S(x \oplus_p d) \tag{4.38}$$

We now let $x = x \ominus_p d$ which leads to

$$B^{-1}(T(A^{-1}(x \ominus_p d)) \ominus_p e) = S(x) \tag{4.39}$$

We now distribute the $B^{-1}matrix$

$$B^{-1}(T(A^{-1}(x \ominus_p d))) \ominus_p B^{-1}(e) = S(x) \tag{4.40}$$

We let $e' = \ominus_p B^{-1}(e)$ and $d' = \ominus_p d$ which results in

$$B^{-1}(T(A^{-1}(x \oplus_p d'))) \oplus_p e' = S(x) \tag{4.41}$$

And we let $B' = B^{-1}$ and $A' = A^{-1}$

$$B'(T(A'(x \oplus d'))) \oplus e' = S(x) \tag{4.42}$$

We know d, e are still elements in $\mathbb{F}_p^n$ and A', B' are just inverse matrices of the original A, B. Also if we consider A, B to be defined as permutation matrices A', B' are also permutation matrices proving symmetry for all 3 classes.

Transitive Proof

In order to prove the transitive property, we have to show if there as a transform from S-Boxes S to T and S-Boxes T to U there also exists a transform from S-Box S to U. We will start with the case of modular addition.

$$T(x) = S(x \oplus_p d_1) \oplus_p e_1 \tag{4.43}$$

$$U(x) = T(x \oplus_p d_2) \oplus_p e_2 \tag{4.44}$$

When we combine the functions we have

$$U(x) = S(x \oplus_p d_1 \oplus_p d_2) \oplus_p e_1 \oplus_p e_2 \tag{4.45}$$

We then let $d' = d_1 \oplus_p d_2$ and $e' = e_1 \oplus_p e_2$ leading to

$$U(x) = S(x \oplus_p d') \oplus_p e' \tag{4.46}$$

d', e' is in $\mathbb{F}_p^n$ which shows the modular addition case does have transitivity.

For the case of permutation modular addition and affine transforms we have:

$$T(x) = B_1(S(A_1(x \oplus_p d_1))) \oplus_p e_1 \tag{4.47}$$

$$U(x) = B_2(T(A_2(x \oplus_p d_2))) \oplus_p e_2 \tag{4.48}$$

When we combine we end up with:

$$U(x) = B_2(B_1(S(A_1(A_2(x \oplus_p d_1 \oplus_p d_2)))) \oplus_p e_1) \oplus_p e_2 \tag{4.49}$$

We extract the e_1 outside the rest of the equation.

$$U(x) = B_2(B_1(S(A_1(A_2(x \oplus_p d_1 \oplus_p d_2))))) \oplus_p B_2(e_1) \oplus_p e_2 \tag{4.50}$$

We let $e' = B_2(e_1) \oplus_p e_2$ and $d' = d_1 \oplus_p d_2$

$$U(x) = B_2(B_1(S(A_1(A_2(x \oplus_p d'))))) \oplus_p e' \tag{4.51}$$

We then combine matrices $B' = B_2B_1$ and $A' = A_1A_2$ leading to

$$B'(S(A'(x \oplus_p d'))) \oplus_p e' \tag{4.52}$$

We know that $d', e' \in \mathbb{F}_p^n$ and that A', B' are both invertible matrices. In the case that A_1, A_2, B_1, B_2 are permutation matrices the matrices A', B' are also permutation matrices. This leads to a new transform using A', B', d', e' which concludes our proof for both PE and AE classes.

Subset Proofs

Now that we have proven that modular addition, PE, and AE classes are in fact mathematical equivalence classes, we will now show they are subsets of one another. Let us first begin with showing modular addition classes are subsets of PE classes. Let us consider a PE class we will denote as P_0 and let us state that there exist an S-Box $S \in P_0$ and that S belongs to the modular addition class M_0 We know that S can generate any other S-Box within the M_0 class and we denote that any other S-Box in M_0 can be denoted with a transform

$$S'(x) = S(x \oplus_p d) \oplus_p e \tag{4.53}$$

Because $S \in P_0$ we can denote the previous transform as a PE transform and let the matrices A, B be the identity matrix. We now have a PE transform on any $S' \in M_0$

$$S'(x) = B(S(A(x \oplus_p d))) \oplus_p e \tag{4.54}$$

Thus the whole modular addition class M_0 must be a sub-class within the PE class P_0

Now consider we have a S-Box S in a affine class A_0 and the S-Box also belongs to the PE class P_0. The S-Box S can generate any other S-Box $S' \in P_0$ using PE transformations in which A, B are permutation matrices and d, e are vectors in $\mathbb{F}_p^n$

$$S'(x) = B(S(A(x \oplus_p d))) \oplus_p e \tag{4.55}$$

In this case, we can use the same constants A, B, d, e but use them in an affine transformation so by having $S \in A_0$ the whole class P_0 can be generated using affine transforms so P_0 is a subset of A_0.

4.4 Preservation for Generalized S-Box Metrics

In this section we will show proofs of S-Box metric preservation under different transformations such as modular addition and permutation modular addition.

4.4.1 Revisited Transparency Order Preservation in Equivalence Classes

Considering the definition of the generalized RTO shown below:

$$RTO(S) = \max_{\beta \in \mathbb{F}_p^n} \left(n - \frac{1}{p^{2n} - p^n} \sum_{a \in \mathbb{F}_p^{n*}} \left| \sum_{j=1}^{n} \sum_{i=1}^{n} (\delta)^{\beta_i \oplus_p \beta_j} * C_{S_i, S_j}(a) \right| \right) \tag{4.56}$$

We consider T to be a transform on S shown below

$$T(x) = B(S(A(x \oplus_p d))) \oplus_p e \tag{4.57}$$

By substituting in T for S we have:

$$RTO(T) = \max_{\beta \in \mathbb{F}_p^n} \left(n - \frac{1}{p^{2n} - p^n} \sum_{a \in \mathbb{F}_p^{n*}} \left| \sum_{j=1}^{n} \sum_{i=1}^{n} (\delta)^{\beta_i \oplus_p \beta_j} * C_{T_i,T_j}(a) \right| \right) \tag{4.58}$$

We then expand the cross correlation equation

$$RTO(T) = \max_{\beta \in \mathbb{F}_p^n} \left(n - \frac{1}{p^{2n} - p^n} \sum_{a \in \mathbb{F}_p^{n*}} \left| \sum_{j=1}^{n} \sum_{i=1}^{n} (\delta)^{\beta_i \oplus_p \beta_j} * \sum_{x \in \mathbb{F}_p^n} (\delta)^{(T_i(x) \oplus_p T_j(x \oplus_p a)} \right| \right) \tag{4.59}$$

For the sake of of saving room we consider RTO to be

$$RTO(T) = \max_{\beta \in \mathbb{F}_p^n} \left(n - \frac{Q_T}{p^{2n} - p^n} \right) \tag{4.60}$$

Where

$$Q_T(\beta) = \sum_{a \in \mathbb{F}_p^{n*}} \left| \sum_{j=1}^{n} \sum_{i=1}^{n} (\delta)^{\beta_i \oplus_p \beta_j} * \sum_{x \in \mathbb{F}_p^n} (\delta)^{(T_i(x) \oplus_p T_j(x \oplus_p a)} \right| \tag{4.61}$$

We then write T in the form of the transformation using S

$$Q_T = \sum_{a \in \mathbb{F}_p^{n*}} \left| \sum_{j=1}^{n} \sum_{i=1}^{n} (\delta)^{\beta_i \oplus_p \beta_j} * \sum_{x \in \mathbb{F}_p^n} (\delta)^{(B(S(A(x \oplus_p d))) \oplus_p e)_i \oplus_p (B(S(A(x \oplus_p d \oplus_p a))) \oplus_p e)_j} \right| \tag{4.62}$$

We then extract the e_i, e_j from the exponents

$$Q_T = \sum_{a \in \mathbb{F}_p^{n*}} \left| \sum_{j=1}^{n} \sum_{i=1}^{n} (\delta)^{\beta_i \oplus_p \beta_j} * \sum_{x \in \mathbb{F}_p^n} (\delta)^{(B(S(A(x \oplus_p d))))_i \oplus_p e_i \oplus_p (B(S(A(x \oplus_p d \oplus_p a))))_j \oplus_p e_j} \right| \tag{4.63}$$

$$Q_T = \sum_{a \in \mathbb{F}_p^{n*}} \left| \sum_{j=1}^{n} \sum_{i=1}^{n} (\delta)^{\beta_i \oplus_p \beta_j \oplus_p e_i \oplus_p e_j} * \sum_{x \in \mathbb{F}_p^n} (\delta)^{(B(S(A(x \oplus_p d))))_i \oplus_p (B(S(A(x \oplus_p d \oplus_p a))))_j} \right| \quad (4.64)$$

The B matrix is then modified using the propriety of reverse distribution.

$$Q_T = \sum_{a \in \mathbb{F}_p^{n*}} \left| \sum_{j=1}^{n} \sum_{i=1}^{n} (\delta)^{\beta_i \oplus_p \beta_j \oplus_p e_i \oplus_p e_j} * \sum_{x \in \mathbb{F}_p^n} (\delta)^{(B(S_i(A(x \oplus_p d))) \oplus_p (S_j(A(x \oplus_p d \oplus_p a))))} \right| \quad (4.65)$$

We then let $y = A(x \oplus_p d)$ also resulting in $y \oplus_p A(a)$ also re-indexing x to y

$$Q_T = \sum_{a \in \mathbb{F}_p^{n*}} \left| \sum_{j=1}^{n} \sum_{i=1}^{n} (\delta)^{\beta_i \oplus_p \beta_j \oplus_p e_i \oplus_p e_j} * \sum_{y \in \mathbb{F}_p^n} (\delta)^{(B(S_i(y) \oplus_p (S_j(y \oplus_p A(a)))))} \right| \quad (4.66)$$

We now let $u = A(a)$ and re-index a to u

$$Q_T = \sum_{u \in \mathbb{F}_p^{n*}} \left| \sum_{j=1}^{n} \sum_{i=1}^{n} (\delta)^{\beta_i \oplus_p \beta_j \oplus_p e_i \oplus_p e_j} * \sum_{y \in \mathbb{F}_p^n} (\delta)^{(B(S_i(y) \oplus_p (S_j(y \oplus_p u))))} \right| \quad (4.67)$$

We let $\beta_i' = \beta_i \oplus_p e_i$ and $\beta_j' = \beta_j \oplus_p e_j$

$$Q_T = \sum_{u \in \mathbb{F}_p^{n*}} \left| \sum_{j=1}^{n} \sum_{i=1}^{n} (\delta)^{\beta_i' \oplus_p \beta_j'} * \sum_{y \in \mathbb{F}_p^n} (\delta)^{(B(S_i(y) \oplus_p (S_j(y \oplus_p u))))} \right| \quad (4.68)$$

We then put the y sum back into the form of cross correlation.

$$Q_T = \sum_{u \in \mathbb{F}_p^{n*}} \left| \sum_{j=1}^{n} \sum_{i=1}^{n} (\delta)^{\beta_i' \oplus_p \beta_j'} * C_{BS_i, BS_j}(u) \right| \quad (4.69)$$

Then we can put Q_T back to the main formula as shown below:

$$RTO(T) = \max_{\beta \in \mathbb{F}_p^n} \left(n - \frac{1}{p^{2n} - p^n} \sum_{u \in \mathbb{F}_p^{n*}} \left| \sum_{j=1}^{n} \sum_{i=1}^{n} (\delta)^{\beta_i' \oplus_p \beta_j'} * C_{BS_i, BS_j}(u) \right| \right) \tag{4.70}$$

If we consider the B matrix to be the identity matrix, we have shown preservation of RTO over the modular addition transformation. Also we can have B to be a permutation matrix and then re-index i, j and we have shown RTO to be preserved under permutation with modular addition. RTO is not preserved under the general case for any invertible matrix B.

4.4.2 Transparency Order Preservation in Equivalence Classes

Consider the base definition of the generalized TO shown below given a S-Box S:

$$TO(S) = \max_{\beta \in \mathbb{F}_p^n} \left(|n - 2HW(\beta)| - \frac{1}{p^{2n} - p^n} \sum_{a \in \mathbb{F}_p^{n*}} \left| \sum_{v \in \mathbb{F}_p^n; HW(v)=1} (\delta)^{v * \beta} W_{DaS}(0, v) \right| \right) \tag{4.71}$$

Where we consider T to be a transform of S shown below:

$$T(x) = B(S(A(x \oplus_p d))) \oplus_p e \tag{4.72}$$

We now write the RTO formula in forms of T

$$TO(T) = \max_{\beta \in \mathbb{F}_p^n} \left(|n - 2HW(\beta)| - \frac{1}{p^{2n} - p^n} \sum_{a \in \mathbb{F}_p^{n*}} \left| \sum_{v \in \mathbb{F}_p^n; HW(v)=1} (\delta)^{v * \beta} W_{DaT}(0, v) \right| \right) \tag{4.73}$$

To save space we consider

$$TO(T) = \max_{\beta \in \mathbb{F}_p^n} \left(|n - 2HW(\beta)| - \frac{Q_T}{p^{2n} - p^n}) \right| \right) \tag{4.74}$$

Where

$$Q_T(\beta) = \sum_{a \in \mathbb{F}_p^{n*}} \left| \sum_{v \in \mathbb{F}_p^n; HW(v)=1} (\delta)^{v*\beta} W_{DaT}(0, v) \right| \tag{4.75}$$

We then unpack the Walsh transform

$$Q_T(\beta) = \sum_{a \in \mathbb{F}_p^{n*}} \left| \sum_{v \in \mathbb{F}_p^n; HW(v)=1} (\delta)^{v*\beta} \sum_{x \in \mathbb{F}_p^n} (\delta)^{v*D_aT(x)} \right| \tag{4.76}$$

We then unpack the discrete derivative

$$Q_T(\beta) = \sum_{a \in \mathbb{F}_p^{n*}} \left| \sum_{v \in \mathbb{F}_p^n; HW(v)=1} (\delta)^{v*\beta} \sum_{x \in \mathbb{F}_p^n} (\delta)^{v*(T(x \oplus_p a) \ominus_p T(x))} \right| \tag{4.77}$$

We now write T in an expanded transform notation

$$Q_T(\beta) = \sum_{a \in \mathbb{F}_p^{n*}} \left| \sum_{v \in \mathbb{F}_p^n; HW(v)=1} (\delta)^{v*\beta} \sum_{x \in \mathbb{F}_p^n} (\delta)^{v*((B(S(A(x \oplus_p a \oplus_p d))) \oplus_p e) \ominus_p (B(S(A(x \oplus_p d))) \oplus_p e)} \right| \tag{4.78}$$

Since the e values are on the outside of the transform we can cancel them because of the $\ominus$ operation and we are left with

$$Q_T(\beta) = \sum_{a \in \mathbb{F}_p^{n*}} \left| \sum_{v \in \mathbb{F}_p^n; HW(v)=1} (\delta)^{v*\beta} \sum_{x \in \mathbb{F}_p^n} (\delta)^{v*((B(S(A(x \oplus_p a \oplus_p d)))) \ominus_p (B(S(A(x \oplus_p d))))} \right| \tag{4.79}$$

Using the proprieties of reverse distribution we then pull out the B Matrix

$$Q_T(\beta) = \sum_{a \in \mathbb{F}_p^{n*}} \left| \sum_{v \in \mathbb{F}_p^n; HW(v)=1} (\delta)^{v*\beta} \sum_{x \in \mathbb{F}_p^n} (\delta)^{v*(B(S(A(x \oplus_p a \oplus_p d)) \ominus_p S(A(x \oplus_p d)))} \right| \tag{4.80}$$

We then let $y = A(x \oplus_p d)$ and that also results in $y \oplus_p A(a) = A(x \oplus_p a \oplus_p d)$. We then re-index x to y resulting in:

$$Q_T(\beta) = \sum_{a \in \mathbb{F}_p^{n*}} \left| \sum_{v \in \mathbb{F}_p^n; HW(v)=1} (\delta)^{v*\beta} \sum_{y \in \mathbb{F}_p^n} (\delta)^{v*(B(S(y \oplus_p A(a)) \ominus_p S(y)))} \right| \tag{4.81}$$

We then let $u = A(a)$ and we re-index a to u resulting in:

$$Q_T(\beta) = \sum_{u \in \mathbb{F}_p^{n*}} \left| \sum_{v \in \mathbb{F}_p^n; HW(v)=1} (\delta)^{v*\beta} \sum_{y \in \mathbb{F}_p^n} (\delta)^{v*(B(S(y \oplus_p u) \ominus_p S(y)))} \right| \tag{4.82}$$

We now re-write the exponent into the discrete derivative suing the definition of the discrete derivative.

$$Q_T(\beta) = \sum_{u \in \mathbb{F}_p^{n*}} \left| \sum_{v \in \mathbb{F}_p^n; HW(v)=1} (\delta)^{v*\beta} \sum_{y \in \mathbb{F}_p^n} (\delta)^{v*D_u BS(y)} \right| \tag{4.83}$$

Now we re-pack the Walsh transform resulting in:

$$Q_T(\beta) = \sum_{u \in \mathbb{F}_p^{n*}} \left| \sum_{v \in \mathbb{F}_p^n; HW(v)=1} (\delta)^{v*\beta} W_{D_u BS(0,v)} \right| \tag{4.84}$$

We now put our computed Q_T value back into the TO definition

$$TO(T) = \max_{\beta \in \mathbb{F}_p^n} \left(|n - 2HW(\beta)| - \frac{1}{p^{2n} - p^n} \sum_{u \in \mathbb{F}_p^{n*}} \left| \sum_{v \in \mathbb{F}_p^n; HW(v)=1} (\delta)^{v*\beta} W_{D_u BS(0,v)} \right| \right) \tag{4.85}$$

Considering B to be the identity matrix, proves preservation under modular addition. If we consider B to be a permutation matrix, we can re-index the v value to account for the permutation and TO is still preserved. TO does not hold under all B that are invertible meaning that affine transformations will not always preserve TO.

4.5 Ternary Equivalence Class Generation

In this section, we will describe how we exhaustively search through all 9! bijective S-Boxes in 3^2 and classify S-Boxes into permutation modular addition classes and affine classes. Our exhaustive search algorithm is an extension of the 2^4 exhaustive search algorithm in the works of (Saarinen, 2011) and (Cheng et al., 2015).

First, we must create a list of balanced component functions. In our case study a component functions is a function on $\mathbb{F}_3^2 \longrightarrow \mathbb{F}_3$. We define a balanced component functions as a component functions that has an equal amount of output given all inputs to the component function. In the case of $\mathbb{F}_3^2$ given all 9 inputs there are three outputs of 0, 1, and 2 making it balanced. We define the list of all balanced component functions as *compFunctionArr* and the component functions are sorted lexicographically based upon output of the component functions.

Next, we define the function *FuncionsCompatable* which denotes a function taking as input two balanced component functions. This function returns if we can combine two balanced component functions into a bijective S-Box as a true/false value. This function concatenates the output of both of the input balanced component functions

and returns true if the concatenation can return all elements in $\mathbb{F}_3^2$ given all inputs in $\mathbb{F}_3^2$.

We then assume that we have code that will take as input a S-Box in $\mathbb{F}_3^2$, invertible matrices, permutation matrices, and trit vectors and outputs the transform on the input S-Box based upon the other input values. For the transform inputs, there exist 48 unique invertible matrices in $\mathbb{F}_3^{2x2}$ for affine transformation and 2 permutation matrices for the permutation modular addition transformation. As for the trit vectors, there exist 9 unique vectors and they are the elements in the field $\mathbb{F}_3^2$.

The algorithm is a nested loop searches through all balanced component functions indexed by i and j and if the component functions can create a S-Box based upon the $FunctionsCompatable$ function we have then found an S-Box. Next, we search through our look-up tables to indicate if we have encountered this S-Box before If the S-Box was in the look-up tables we disregard the S-Box and continue looping. If we have not encountered this S-Box before, we take the S-Box as input and iterate though all possible transform constants and generate the AE/PE class. After generating the class, we store it in a list and count how the size of the AE/PE class and store add the size to a global counter. When the Global counter reaches 9! we have found all S-Boxes and have computed all AE/PE classes in $\mathbb{F}_3^2$.

In Algorithm 3, one will edit the GenerateClassFromSBox step to either generate AE or PE classes resulting in the exhaustive search of AE or PE classes.

We note that we order the AE and PE classes lexicographically. In a AE/PE class we search though all representatives and find the S-Box with the smallest lexicographical representation and let that S-Box represent the class. We then order the

Algorithm 3: Pseudocode to Generate AE/PE Classes in Ternary Case Study

```
Result: List of AE/PE Classes
Generate CompFunctionsArr containing all balances trit functions Initialize
 ClassList structure
for index1 = 0, to len(CompFunctionsArr) do
    CompFunction1 = CompFunctionsArr[index1];
    for index2 = index1 + to len(CompFunctionsArr) do
        CompFunction2 = CompFunctionsArr[index2];
        if CompFunctionsCompatable(CompFunction1,CompFunction2) then
            tempSBox = concat(CompFunction1,CompFunction2);
            if tempSBox not in ClassList then
                TempArr = GenerateClassFromSBox(tempSBox);
                Append TempArr to ClassList;
                if All S-Boxes Accounted For then
                    break our of both for loops
                end
            end
        end
    end
end
return ClassList;
```

AE and PE classes lexicographically based upon their representatives. As an example the Identity S-Box (012345678) is the 'smallest' S-Box lexicographically and the AE/PE classes containing this S-Box will be AE_0 and PE_0

4.5.1 Two Digit Ternary Classes

Because of the proprieties of AE/PE classes, we can show that AE classes are union of PE classes. As we already have the algorithm to generate the classes we will state in this subsection the AE classes as union of PE Classes.

$AE_0 = PE_0 \cup PE_9 \cup PE_{10} \cup PE_{11} \cup PE_{14} \cup PE_{15} \cup PE_{421} \cup PE_{422} \cup PE_{423} \cup PE_{426} \cup PE_{427} \cup PE_{1160} \cup PE_{1161}$

$$AE_1 = PE_1 \cup PE_4 \cup PE_5 \cup PE_6 \cup PE_7 \cup PE_8 \cup PE_{16} \cup PE_{38} \cup PE_{43} \cup PE_{59} \cup PE_{69} \cup PE_{84} \cup PE_{90} \cup PE_{106} \cup PE_{116} \cup PE_{133} \cup PE_{138} \cup PE_{153} \cup PE_{176} \cup PE_{179} \cup PE_{197} \cup PE_{202} \cup PE_{223} \cup PE_{224} \cup PE_{243} \cup PE_{259} \cup PE_{278} \cup PE_{285} \cup PE_{299} \cup PE_{313} \cup PE_{327} \cup PE_{338} \cup PE_{357} \cup PE_{367} \cup PE_{384} \cup PE_{394} \cup PE_{399} \cup PE_{407} \cup PE_{417} \cup PE_{418} \cup PE_{419} \cup PE_{420} \cup PE_{449} \cup PE_{792} \cup PE_{802} \cup PE_{942} \cup PE_{953} \cup PE_{1038} \cup PE_{1048} \cup PE_{1136}$$

$$AE_2 = PE_2 \cup PE_3 \cup PE_{12} \cup PE_{13} \cup PE_{415} \cup PE_{416} \cup PE_{424} \cup PE_{425} \cup PE_{671} \cup PE_{674} \cup PE_{676} \cup PE_{934} \cup PE_{937} \cup PE_{940} \cup PE_{995} \cup PE_{996} \cup PE_{1003} \cup PE_{1005} \cup PE_{1006} \cup PE_{1100} \cup PE_{1102} \cup PE_{1103} \cup PE_{1148} \cup PE_{1149} \cup PE_{1150} \cup PE_{1151} \cup PE_{1152} \cup PE_{1153} \cup PE_{1154} \cup PE_{1155} \cup PE_{1156} \cup PE_{1157} \cup PE_{1158} \cup PE_{1159}$$

$$AE_3 = PE_{17} \cup PE_{23} \cup PE_{25} \cup PE_{27} \cup PE_{28} \cup PE_{30} \cup PE_{33} \cup PE_{39} \cup PE_{42} \cup PE_{47} \cup PE_{48} \cup PE_{50} \cup PE_{52} \cup PE_{55} \cup PE_{57} \cup PE_{58} \cup PE_{68} \cup PE_{70} \cup PE_{72} \cup PE_{74} \cup PE_{77} \cup PE_{79} \cup PE_{80} \cup PE_{85} \cup PE_{91} \cup PE_{94} \cup PE_{97} \cup PE_{98} \cup PE_{99} \cup PE_{101} \cup PE_{103} \cup PE_{105} \cup PE_{115} \cup PE_{118} \cup PE_{120} \cup PE_{121} \cup PE_{124} \cup PE_{126} \cup PE_{128} \cup PE_{134} \cup PE_{137} \cup PE_{141} \cup PE_{143} \cup PE_{146} \cup PE_{148} \cup PE_{149} \cup PE_{151} \cup PE_{154} \cup PE_{161} \cup PE_{162} \cup PE_{164} \cup PE_{166} \cup PE_{168} \cup PE_{169} \cup PE_{171} \cup PE_{175} \cup PE_{180} \cup PE_{186} \cup PE_{188} \cup PE_{189} \cup PE_{191} \cup PE_{193} \cup PE_{198} \cup PE_{201} \cup PE_{209} \cup PE_{211} \cup PE_{212} \cup PE_{214} \cup PE_{216} \cup PE_{222} \cup PE_{225} \cup PE_{227} \cup PE_{228} \cup PE_{233} \cup PE_{237} \cup PE_{238} \cup PE_{242} \cup PE_{244} \cup PE_{249} \cup PE_{250} \cup PE_{251} \cup PE_{254} \cup PE_{256} \cup PE_{258} \cup PE_{262} \cup PE_{265} \cup PE_{269} \cup PE_{274} \cup PE_{277} \cup$$

$$PE_{279} \cup PE_{281} \cup PE_{284} \cup PE_{286} \cup PE_{288} \cup PE_{294} \cup PE_{295} \cup PE_{298} \cup PE_{300} \cup$$
$$PE_{304} \cup PE_{306} \cup PE_{307} \cup PE_{310} \cup PE_{312} \cup PE_{314} \cup PE_{317} \cup PE_{322} \cup PE_{324} \cup$$
$$PE_{328} \cup PE_{330} \cup PE_{332} \cup PE_{335} \cup PE_{337} \cup PE_{341} \cup PE_{344} \cup PE_{348} \cup PE_{353} \cup$$
$$PE_{356} \cup PE_{358} \cup PE_{361} \cup PE_{362} \cup PE_{366} \cup PE_{368} \cup PE_{371} \cup PE_{375} \cup PE_{379} \cup$$
$$PE_{382} \cup PE_{385} \cup PE_{387} \cup PE_{388} \cup PE_{393} \cup PE_{395} \cup PE_{400} \cup PE_{402} \cup PE_{406} \cup$$
$$PE_{410} \cup PE_{413} \cup PE_{429} \cup PE_{432} \cup PE_{433} \cup PE_{435} \cup PE_{436} \cup PE_{440} \cup PE_{441} \cup$$
$$PE_{443} \cup PE_{451} \cup PE_{453} \cup PE_{454} \cup PE_{455} \cup PE_{456} \cup PE_{458} \cup PE_{468} \cup PE_{478} \cup$$
$$PE_{480} \cup PE_{482} \cup PE_{490} \cup PE_{493} \cup PE_{498} \cup PE_{500} \cup PE_{501} \cup PE_{511} \cup PE_{513} \cup$$
$$PE_{527} \cup PE_{528} \cup PE_{533} \cup PE_{539} \cup PE_{550} \cup PE_{555} \cup PE_{563} \cup PE_{564} \cup PE_{568} \cup$$
$$PE_{569} \cup PE_{570} \cup PE_{571} \cup PE_{589} \cup PE_{590} \cup PE_{592} \cup PE_{597} \cup PE_{598} \cup PE_{599} \cup$$
$$PE_{612} \cup PE_{613} \cup PE_{614} \cup PE_{619} \cup PE_{624} \cup PE_{631} \cup PE_{635} \cup PE_{641} \cup PE_{646} \cup$$
$$PE_{648} \cup PE_{657} \cup PE_{659} \cup PE_{661} \cup PE_{678} \cup PE_{683} \cup PE_{686} \cup PE_{687} \cup PE_{690} \cup$$
$$PE_{697} \cup PE_{704} \cup PE_{707} \cup PE_{712} \cup PE_{719} \cup PE_{722} \cup PE_{730} \cup PE_{731} \cup PE_{733} \cup$$
$$PE_{744} \cup PE_{748} \cup PE_{753} \cup PE_{763} \cup PE_{764} \cup PE_{766} \cup PE_{770} \cup PE_{773} \cup PE_{778} \cup$$
$$PE_{780} \cup PE_{781} \cup PE_{783} \cup PE_{787} \cup PE_{788} \cup PE_{790} \cup PE_{799} \cup PE_{800} \cup PE_{804} \cup$$
$$PE_{808} \cup PE_{810} \cup PE_{823} \cup PE_{828} \cup PE_{832} \cup PE_{839} \cup PE_{844} \cup PE_{860} \cup PE_{862} \cup$$
$$PE_{869} \cup PE_{870} \cup PE_{875} \cup PE_{882} \cup PE_{885} \cup PE_{888} \cup PE_{890} \cup PE_{894} \cup PE_{898} \cup$$
$$PE_{906} \cup PE_{910} \cup PE_{912} \cup PE_{917} \cup PE_{925} \cup PE_{929} \cup PE_{946} \cup PE_{950} \cup PE_{951} \cup$$
$$PE_{952} \cup PE_{958} \cup PE_{959} \cup PE_{962} \cup PE_{969} \cup PE_{973} \cup PE_{977} \cup PE_{980} \cup PE_{990} \cup$$
$$PE_{1009} \cup PE_{1014} \cup PE_{1016} \cup PE_{1019} \cup PE_{1024} \cup PE_{1026} \cup PE_{1032} \cup PE_{1034} \cup$$
$$PE_{1039} \cup PE_{1043} \cup PE_{1054} \cup PE_{1057} \cup PE_{1064} \cup PE_{1066} \cup PE_{1072} \cup PE_{1073} \cup$$
$$PE_{1075} \cup PE_{1078} \cup PE_{1079} \cup PE_{1082} \cup PE_{1091} \cup PE_{1093} \cup PE_{1104} \cup PE_{1110} \cup$$
$$PE_{1112} \cup PE_{1118} \cup PE_{1126} \cup PE_{1127} \cup PE_{1139} \cup PE_{1141}$$

$$AE_4 = PE_{18} \cup PE_{21} \cup PE_{22} \cup PE_{26} \cup PE_{31} \cup PE_{32} \cup PE_{35} \cup PE_{36} \cup PE_{41} \cup PE_{45} \cup PE_{46} \cup PE_{49} \cup PE_{53} \cup PE_{54} \cup PE_{61} \cup PE_{62} \cup PE_{64} \cup PE_{67} \cup PE_{71} \cup PE_{75} \cup PE_{78} \cup PE_{81} \cup PE_{82} \cup PE_{86} \cup PE_{88} \cup PE_{92} \cup PE_{95} \cup PE_{96} \cup PE_{100} \cup PE_{104} \cup PE_{108} \cup PE_{109} \cup PE_{111} \cup PE_{114} \cup PE_{119} \cup PE_{122} \cup PE_{123} \cup PE_{127} \cup PE_{130} \cup PE_{131} \cup PE_{136} \cup PE_{140} \cup PE_{144} \cup PE_{145} \cup PE_{147} \cup PE_{150} \cup PE_{155} \cup PE_{158} \cup PE_{165} \cup PE_{167} \cup PE_{170} \cup PE_{174} \cup PE_{178} \cup PE_{181} \cup PE_{184} \cup PE_{185} \cup PE_{192} \cup PE_{194} \cup PE_{195} \cup PE_{199} \cup PE_{204} \cup PE_{205} \cup PE_{207} \cup PE_{208} \cup PE_{213} \cup PE_{217} \cup PE_{218} \cup PE_{221} \cup PE_{226} \cup PE_{231} \cup PE_{232} \cup PE_{234} \cup PE_{239} \cup PE_{240} \cup PE_{245} \cup PE_{247} \cup PE_{248} \cup PE_{253} \cup PE_{257} \cup PE_{260} \cup PE_{263} \cup PE_{266} \cup PE_{268} \cup PE_{271} \cup PE_{273} \cup PE_{276} \cup PE_{282} \cup PE_{287} \cup PE_{289} \cup PE_{290} \cup PE_{292} \cup PE_{296} \cup PE_{301} \cup PE_{303} \cup PE_{305} \cup PE_{309} \cup PE_{315} \cup PE_{318} \cup PE_{320} \cup PE_{323} \cup PE_{326} \cup PE_{329} \cup PE_{331} \cup PE_{334} \cup PE_{336} \cup PE_{339} \cup PE_{342} \cup PE_{345} \cup PE_{347} \cup PE_{350} \cup PE_{352} \cup PE_{355} \cup PE_{360} \cup PE_{364} \cup PE_{369} \cup PE_{373} \cup PE_{376} \cup PE_{378} \cup PE_{381} \cup PE_{386} \cup PE_{389} \cup PE_{391} \cup PE_{398} \cup PE_{401} \cup PE_{403} \cup PE_{405} \cup PE_{408} \cup PE_{411} \cup PE_{428} \cup PE_{434} \cup PE_{437} \cup PE_{438} \cup PE_{444} \cup PE_{447} \cup PE_{448} \cup PE_{452} \cup PE_{469} \cup PE_{481} \cup PE_{502} \cup PE_{512} \cup PE_{535} \cup PE_{553} \cup PE_{581} \cup PE_{588} \cup PE_{606} \cup PE_{623} \cup PE_{649} \cup PE_{685} \cup PE_{699} \cup PE_{715} \cup PE_{721} \cup PE_{743} \cup PE_{754} \cup PE_{762} \cup PE_{779} \cup PE_{785} \cup PE_{786} \cup PE_{789} \cup PE_{791} \cup PE_{793} \cup PE_{798} \cup PE_{801} \cup PE_{803} \cup PE_{806} \cup PE_{809} \cup PE_{811} \cup PE_{826} \cup PE_{840} \cup PE_{854} \cup PE_{876} \cup PE_{905} \cup PE_{920} \cup PE_{933} \cup PE_{941} \cup PE_{944} \cup PE_{948} \cup PE_{954} \cup PE_{957} \cup PE_{984} \cup PE_{1018} \cup PE_{1030} \cup PE_{1036} \cup PE_{1042} \cup PE_{1044} \cup PE_{1047} \cup PE_{1052} \cup PE_{1071} \cup PE_{1077} \cup PE_{1081} \cup PE_{1119} \cup PE_{1121} \cup PE_{1137}$$

$$AE_5 = PE_{19} \cup PE_{20} \cup PE_{34} \cup PE_{37} \cup PE_{40} \cup PE_{44} \cup PE_{60} \cup PE_{63} \cup$$

$$PE_{65} \cup PE_{66} \cup PE_{83} \cup PE_{87} \cup PE_{89} \cup PE_{93} \cup PE_{107} \cup PE_{110} \cup PE_{112} \cup$$
$$PE_{113} \cup PE_{129} \cup PE_{132} \cup PE_{135} \cup PE_{139} \cup PE_{156} \cup PE_{157} \cup PE_{159} \cup PE_{160} \cup$$
$$PE_{173} \cup PE_{177} \cup PE_{182} \cup PE_{183} \cup PE_{196} \cup PE_{200} \cup PE_{203} \cup PE_{206} \cup PE_{219} \cup$$
$$PE_{220} \cup PE_{229} \cup PE_{230} \cup PE_{235} \cup PE_{241} \cup PE_{246} \cup PE_{252} \cup PE_{261} \cup PE_{267} \cup$$
$$PE_{270} \cup PE_{272} \cup PE_{283} \cup PE_{291} \cup PE_{293} \cup PE_{297} \cup PE_{302} \cup PE_{308} \cup PE_{319} \cup$$
$$PE_{321} \cup PE_{325} \cup PE_{333} \cup PE_{340} \cup PE_{346} \cup PE_{349} \cup PE_{351} \cup PE_{359} \cup PE_{363} \cup$$
$$PE_{372} \cup PE_{374} \cup PE_{377} \cup PE_{383} \cup PE_{390} \cup PE_{392} \cup PE_{397} \cup PE_{404} \cup PE_{409} \cup$$
$$PE_{414} \cup PE_{430} \cup PE_{431} \cup PE_{442} \cup PE_{445} \cup PE_{446} \cup PE_{450} \cup PE_{457} \cup PE_{459} \cup$$
$$PE_{463} \cup PE_{471} \cup PE_{472} \cup PE_{477} \cup PE_{479} \cup PE_{483} \cup PE_{487} \cup PE_{489} \cup PE_{494} \cup$$
$$PE_{497} \cup PE_{499} \cup PE_{505} \cup PE_{510} \cup PE_{514} \cup PE_{517} \cup PE_{522} \cup PE_{524} \cup PE_{525} \cup$$
$$PE_{529} \cup PE_{536} \cup PE_{540} \cup PE_{543} \cup PE_{546} \cup PE_{554} \cup PE_{557} \cup PE_{560} \cup PE_{562} \cup$$
$$PE_{574} \cup PE_{577} \cup PE_{582} \cup PE_{583} \cup PE_{586} \cup PE_{587} \cup PE_{593} \cup PE_{603} \cup PE_{604} \cup$$
$$PE_{607} \cup PE_{608} \cup PE_{618} \cup PE_{621} \cup PE_{629} \cup PE_{630} \cup PE_{634} \cup PE_{636} \cup PE_{642} \cup$$
$$PE_{651} \cup PE_{654} \cup PE_{656} \cup PE_{662} \cup PE_{666} \cup PE_{667} \cup PE_{669} \cup PE_{672} \cup PE_{673} \cup$$
$$PE_{675} \cup PE_{679} \cup PE_{689} \cup PE_{692} \cup PE_{693} \cup PE_{695} \cup PE_{698} \cup PE_{702} \cup PE_{703} \cup$$
$$PE_{706} \cup PE_{714} \cup PE_{718} \cup PE_{724} \cup PE_{727} \cup PE_{729} \cup PE_{732} \cup PE_{734} \cup PE_{740} \cup$$
$$PE_{741} \cup PE_{747} \cup PE_{750} \cup PE_{752} \cup PE_{755} \cup PE_{760} \cup PE_{765} \cup PE_{767} \cup PE_{769} \cup$$
$$PE_{774} \cup PE_{777} \cup PE_{782} \cup PE_{784} \cup PE_{794} \cup PE_{795} \cup PE_{796} \cup PE_{805} \cup PE_{807} \cup$$
$$PE_{813} \cup PE_{815} \cup PE_{818} \cup PE_{820} \cup PE_{821} \cup PE_{825} \cup PE_{834} \cup PE_{835} \cup PE_{836} \cup$$
$$PE_{841} \cup PE_{848} \cup PE_{850} \cup PE_{851} \cup PE_{852} \cup PE_{856} \cup PE_{857} \cup PE_{859} \cup PE_{871} \cup$$
$$PE_{872} \cup PE_{874} \cup PE_{877} \cup PE_{883} \cup PE_{884} \cup PE_{886} \cup PE_{887} \cup PE_{891} \cup PE_{896} \cup$$
$$PE_{899} \cup PE_{900} \cup PE_{902} \cup PE_{913} \cup PE_{914} \cup PE_{918} \cup PE_{919} \cup PE_{921} \cup PE_{926} \cup$$
$$PE_{927} \cup PE_{930} \cup PE_{931} \cup PE_{935} \cup PE_{936} \cup PE_{938} \cup PE_{939} \cup PE_{943} \cup PE_{945} \cup$$
$$PE_{947} \cup PE_{955} \cup PE_{960} \cup PE_{961} \cup PE_{963} \cup PE_{965} \cup PE_{971} \cup PE_{972} \cup PE_{974} \cup$$

$PE_{976} \cup PE_{979} \cup PE_{982} \cup PE_{983} \cup PE_{987} \cup PE_{991} \cup PE_{992} \cup PE_{993} \cup PE_{994} \cup PE_{997} \cup PE_{998} \cup PE_{999} \cup PE_{1000} \cup PE_{1001} \cup PE_{1002} \cup PE_{1007} \cup PE_{1011} \cup PE_{1015} \cup PE_{1022} \cup PE_{1025} \cup PE_{1029} \cup PE_{1033} \cup PE_{1035} \cup PE_{1037} \cup PE_{1040} \cup PE_{1045} \cup PE_{1055} \cup PE_{1056} \cup PE_{1058} \cup PE_{1060} \cup PE_{1061} \cup PE_{1062} \cup PE_{1067} \cup PE_{1069} \cup PE_{1074} \cup PE_{1076} \cup PE_{1080} \cup PE_{1083} \cup PE_{1085} \cup PE_{1087} \cup PE_{1089} \cup PE_{1094} \cup PE_{1095} \cup PE_{1097} \cup PE_{1099} \cup PE_{1106} \cup PE_{1109} \cup PE_{1115} \cup PE_{1116} \cup PE_{1117} \cup PE_{1122} \cup PE_{1124} \cup PE_{1125} \cup PE_{1128} \cup PE_{1130} \cup PE_{1131} \cup PE_{1132} \cup PE_{1133} \cup PE_{1134} \cup PE_{1135} \cup PE_{1138} \cup PE_{1140} \cup PE_{1143} \cup PE_{1144} \cup PE_{1145} \cup PE_{1147}$

$AE_6 = PE_{24} \cup PE_{29} \cup PE_{51} \cup PE_{56} \cup PE_{73} \cup PE_{76} \cup PE_{102} \cup PE_{117} \cup PE_{125} \cup PE_{142} \cup PE_{152} \cup PE_{163} \cup PE_{172} \cup PE_{187} \cup PE_{190} \cup PE_{210} \cup PE_{215} \cup PE_{236} \cup PE_{255} \cup PE_{264} \cup PE_{275} \cup PE_{280} \cup PE_{311} \cup PE_{316} \cup PE_{343} \cup PE_{354} \cup PE_{365} \cup PE_{370} \cup PE_{380} \cup PE_{396} \cup PE_{412} \cup PE_{439} \cup PE_{797} \cup PE_{949} \cup PE_{956} \cup PE_{1041} \cup PE_{1046}$

$AE_7 = PE_{460} \cup PE_{461} \cup PE_{462} \cup PE_{464} \cup PE_{467} \cup PE_{470} \cup PE_{473} \cup PE_{474} \cup PE_{475} \cup PE_{476} \cup PE_{484} \cup PE_{486} \cup PE_{488} \cup PE_{491} \cup PE_{492} \cup PE_{495} \cup PE_{496} \cup PE_{504} \cup PE_{506} \cup PE_{507} \cup PE_{509} \cup PE_{515} \cup PE_{516} \cup PE_{518} \cup PE_{519} \cup PE_{521} \cup PE_{523} \cup PE_{526} \cup PE_{530} \cup PE_{531} \cup PE_{537} \cup PE_{538} \cup PE_{541} \cup PE_{542} \cup PE_{544} \cup PE_{545} \cup PE_{547} \cup PE_{548} \cup PE_{551} \cup PE_{552} \cup PE_{556} \cup PE_{558} \cup PE_{559} \cup PE_{561} \cup PE_{565} \cup PE_{566} \cup PE_{573} \cup PE_{575} \cup PE_{576} \cup PE_{578} \cup PE_{579} \cup PE_{580} \cup PE_{585} \cup PE_{594} \cup PE_{595} \cup PE_{600} \cup PE_{601} \cup PE_{602} \cup PE_{605} \cup PE_{609} \cup PE_{610} \cup PE_{611} \cup PE_{615} \cup PE_{616} \cup PE_{617} \cup PE_{622} \cup PE_{625} \cup PE_{626} \cup PE_{628} \cup PE_{633} \cup PE_{637} \cup$

$$PE_{638} \cup PE_{639} \cup PE_{640} \cup PE_{643} \cup PE_{644} \cup PE_{645} \cup PE_{647} \cup PE_{650} \cup PE_{652} \cup PE_{655} \cup PE_{658} \cup PE_{660} \cup PE_{663} \cup PE_{664} \cup PE_{665} \cup PE_{677} \cup PE_{680} \cup PE_{682} \cup PE_{684} \cup PE_{688} \cup PE_{691} \cup PE_{696} \cup PE_{700} \cup PE_{701} \cup PE_{705} \cup PE_{708} \cup PE_{709} \cup PE_{710} \cup PE_{711} \cup PE_{716} \cup PE_{717} \cup PE_{720} \cup PE_{723} \cup PE_{726} \cup PE_{728} \cup PE_{735} \cup PE_{736} \cup PE_{737} \cup PE_{739} \cup PE_{742} \cup PE_{745} \cup PE_{749} \cup PE_{751} \cup PE_{756} \cup PE_{758} \cup PE_{759} \cup PE_{761} \cup PE_{768} \cup PE_{771} \cup PE_{772} \cup PE_{776} \cup PE_{812} \cup PE_{814} \cup PE_{816} \cup PE_{817} \cup PE_{822} \cup PE_{824} \cup PE_{829} \cup PE_{830} \cup PE_{833} \cup PE_{837} \cup PE_{838} \cup PE_{842} \cup PE_{843} \cup PE_{845} \cup PE_{846} \cup PE_{849} \cup PE_{855} \cup PE_{858} \cup PE_{861} \cup PE_{863} \cup PE_{865} \cup PE_{866} \cup PE_{867} \cup PE_{873} \cup PE_{879} \cup PE_{881} \cup PE_{889} \cup PE_{892} \cup PE_{895} \cup PE_{897} \cup PE_{901} \cup PE_{903} \cup PE_{904} \cup PE_{907} \cup PE_{909} \cup PE_{911} \cup PE_{915} \cup PE_{916} \cup PE_{922} \cup PE_{924} \cup PE_{928} \cup PE_{932} \cup PE_{964} \cup PE_{967} \cup PE_{970} \cup PE_{978} \cup PE_{981} \cup PE_{985} \cup PE_{989} \cup PE_{1010} \cup PE_{1013} \cup PE_{1017} \cup PE_{1020} \cup PE_{1023} \cup PE_{1027} \cup PE_{1028} \cup PE_{1031} \cup PE_{1050} \cup PE_{1053} \cup PE_{1063} \cup PE_{1070} \cup PE_{1084} \cup PE_{1086} \cup PE_{1090} \cup PE_{1096} \cup PE_{1098} \cup PE_{1105} \cup PE_{1111} \cup PE_{1113} \cup PE_{1114}$$

$$AE_8 = PE_{465} \cup PE_{466} \cup PE_{485} \cup PE_{503} \cup PE_{508} \cup PE_{520} \cup PE_{532} \cup PE_{534} \cup PE_{549} \cup PE_{567} \cup PE_{572} \cup PE_{584} \cup PE_{591} \cup PE_{596} \cup PE_{620} \cup PE_{627} \cup PE_{632} \cup PE_{653} \cup PE_{668} \cup PE_{670} \cup PE_{681} \cup PE_{694} \cup PE_{713} \cup PE_{725} \cup PE_{738} \cup PE_{746} \cup PE_{757} \cup PE_{775} \cup PE_{819} \cup PE_{827} \cup PE_{831} \cup PE_{847} \cup PE_{853} \cup PE_{864} \cup PE_{868} \cup PE_{878} \cup PE_{880} \cup PE_{893} \cup PE_{908} \cup PE_{923} \cup PE_{966} \cup PE_{968} \cup PE_{975} \cup PE_{986} \cup PE_{988} \cup PE_{1004} \cup PE_{1008} \cup PE_{1012} \cup PE_{1021} \cup PE_{1049} \cup PE_{1051} \cup PE_{1059} \cup PE_{1065} \cup PE_{1068} \cup PE_{1088} \cup PE_{1092} \cup PE_{1101} \cup PE_{1107} \cup PE_{1108} \cup PE_{1120} \cup PE_{1123} \cup PE_{1129} \cup PE_{1142} \cup PE_{1146}$$

4.6 Generalized GIFT Encryption Scheme

In this section, we will give a general description on how we take a SPN cipher such as GIFT and generalized it to work over any prime power. This is more of a theoretical section and will not have any implementation details but we then will show how to implement such a structure in our ternary case study on binary hardware.

First, we will go over the parameters of the GIFT64 SPN cipher in the classical binary setting. The GIFT64 cipher operates on a block size of 64 bits in which the blocks can be represented as elements in $\mathbb{F}_2^{64}$. The sub-round function S-Box layer is a function of $\mathbb{F}_2^4 \to \mathbb{F}_2^4$ in which 4 is a factor of the block size 64. The permBits layer acts as a bit permutation on the state of bits in the block $\mathbb{F}_2^{64}$. Lastly, the add round key performs the XOR operation on the block state with a given round key in which both are elements in $\mathbb{F}_2^{64}$ and the XOR operation is addition modulo 2 to the two 64-bit states.

With the base parameter set known in the binary case we will now describe a generalisation of the GIFT64 cryptosystem by going into detail on how to generalize each of the parameters.

4.6.1 Generalized Block Structure

In our generalization, instead of considering 64-bit blocks, we consider a set of m elements in $\mathbb{F}_p$ which can be denoted as $\mathbb{F}_p^m$. In order to operate on real world data, one can convert any base 2 or base 10 number to be a base p number for operation on this cryptographic scheme.

4.6.2 Generalized S-Box Structure

The generalized S-Box will be bijective S-Boxes of the form $\mathbb{F}_p^n \to \mathbb{F}_p^n$ in which n is a factor of the block size m. We make the relationship of n, m because we want there to not be 'fractional' S-Boxes in the cryptographic scheme and in this structure there will be $\frac{m}{n}$ full S-Boxes being computed on each round. The propriety of being a bijective S-Box is needed in order to invert the S-Box for decryption. While one can use $\frac{m}{n}$ different S-Boxes, we will consider all S-Boxes to be the same to keep lightweight proprieties since this is based upon the GIFT64 scheme and the binary construction of this scheme is made for the lightweight setting.

As for S-Box proprieties, we note that a S-Box in $\mathbb{F}_p^n$ can be denoted as a concatenation of n different component function acting in $\mathbb{F}_p^n \to \mathbb{F}_p$. Because of this, in determining the security of a generalized S-Box one can consider the S-Box itself and the component functions to evaluate the security of the S-Box in the generalized setting.

4.6.3 Generalized Permutation Structure

While the classical sub-round function is denoted as permBits in the general setting, we are not permuting bits so we should not note this layer is 'permBits' but is is a permutation on the block state in $\mathbb{F}_p^m$. This layer operates a permutation on the elements in the vector $\mathbb{F}_p^m$ and must be invertible.

4.6.4 Add Round Key

As the last sub-round function, the add round key is also similar to the binary case except instead of the block state and key being elements in $\mathbb{F}_2^m$ they are elements in $\mathbb{F}_p^m$. In order to do the addition we consider the addition of the vectors element

wise mod p which is also denoted as $\oplus_p$. Unlike the binary case of XOR in which addition ($\oplus$) and subtraction ($\ominus$) are the same operating we denote modular addition and subtraction to be ($\oplus_p$) and ($\ominus_p$) During the encryption process, we use modular addition to add the round key and during the decryption process, we use modular subtraction to undo the key addition.

4.7 Ternary GIFT Encryption Scheme

In this section, we will be applying the theoretical foundations on generalized GIFT cryptographic scheme and develop a creation of ternary GIFT. This ternary GIFT cryptographic scheme will operate on a block of 32-trits, also noted as $\mathbb{F}_3^{32}$. The S-Box layer of this scheme will be a two trit to two trit function described as $\mathbb{F}_3^2 \to \mathbb{F}_3^2$. The permutation layer will be a trit permutation on the 32 trit block state.

This section will go into detail of the mathematical structure of our ternary case study as well as give details on how we construct our implementation of the ternary cryptosystem to work on binary micro-controllers which were coded up in C for our case study. Similar to the binary version of GIFT, our case study has a SPN structure and uses iterated round in which each round has S-Box, PermTrits, and add round key layers.

4.7.1 S-Box

In our case study, we use varying S-Boxes that will be defined in a later section. We note that any S-Box in our study will need to be a bijective function in $\mathbb{F}_3^2 \to \mathbb{F}_3^2$. A sample of our first S-Box in the trit notation is shown below in Table 4.4 The notation shown above, in Table 4.4, is the two trit notation but as it takes up a significant amount of room to display we will note the ASCII notation which is the

Table 4.4: Ternary GIFT Sample S-Box

x	00	01	02	10	11	12	20	21	22
S1(x)	02	12	11	00	01	10	21	20	22

the decimal output of the S-Box in order of the inputs from 0 to 8. The ASCII representation of the S-Box S1 shown in Table 4.4 is 254013768 which indication $S1(0) = 2, S1(1) = 4, ..., S1(8) = 8$.

4.7.2 PermTrits

Compared to the binary case, we are permuting trits (the BCT two bit quantities) instead of bits. The permutation table is shown below for our 32-trit GIFT implementation: As there is no 32-bit version of GIFT, we constructed a PermTrits 32-trit

Table 4.5: Ternary GIFT - (PermTrits) Permutation

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
P(i)	0	8	16	24	1	9	17	25	2	10	18	26	3	11	19	27
i	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
P(i)	4	12	20	28	5	13	21	29	6	14	22	30	7	15	23	31

version. If we were to extend our work to be a 64-trit (128-bit), we could use the permutation table lookup of GIFT64.

4.7.3 Add Round Key

For use in encryption, we use modular addition of a vector of length n that can also be represented as $\mathbb{F}_3^n$. In our case, we consider a 32-trit block cipher so $n = 32$.

As noted previously, we will be computing addition mod 3 on the vectors and our notation of adding two vectors mod 3 is $c = a \oplus_3 b$ When performing the decryption operation, we will use modular subtraction instead of modular addition which is noted as $c = a \ominus_3 b$ or $c = a \oplus_3 (-b)$ that denotes adding the inverse vector with respect to

addition mod 3.

Example single and two trit addition Cayley tables are shown below:

Table 4.6: Single Trit Modular Addition Cayley Table

$\oplus_3$	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

Table 4.7: Two Trit Modular Addition Cayley Table

$\oplus_3$	00	01	02	10	11	12	20	21	22
00	00	01	02	10	11	12	20	21	22
01	01	02	00	11	12	10	21	22	20
02	02	00	01	12	10	11	22	20	21
10	10	11	12	20	21	22	00	01	02
11	11	12	10	21	22	20	01	02	00
12	12	10	11	22	20	21	02	00	01
20	20	21	22	00	01	02	10	11	12
21	21	22	20	01	02	00	11	12	10
22	22	20	21	02	00	01	12	10	11

4.7.4 Key Scheduler

For our case study we needed a way to crate 28 round keys from the 64-trit private key. Taking inspiration from (Banik et al., 2017) we created a key scheduler for our ternary case study and it is defined below:

In our ternary case study the key scheduler takes as input a 64-trit key and produces 28 round keys from the 64-trit private key. The private key and round keys are interpreted as binary coded trits (BCT) in our implementation for micro-controllers that have an 8-bit native data type. In the beginning of the key scheduling process, the BCT private key is stored in what we define as a keyState(KS) that holds

16 4-trit chunks of data and we denote || to be the concatenation operation.

$$KS = ks_{15}||ks_{14}||ks_{13}||ks_{12}||ks_{11}||ks_{10}||ks_9||ks_8|| \\ ks_7||ks_6||ks_5||ks_4||ks_3||ks_2||ks_1||ks_0 \quad (4.86)$$

For each round we extract quantities from the key-state that are noted as U and V each being 8-trit quantities. The first 8 trit quantities are stored as U quantity and the second 8 trit quantities are stored as the V quantity such that U and V combined store 16-trits of information relating to the private key.

$$U = ks_1||ks_0 \quad (4.87)$$

$$V = ks_3||ks_2 \quad (4.88)$$

Next, we create a round key by extracting data from the U/V quantities into a round key that will be 32-trits long. We note that in our 32-trit round keys 16-trits are set to 00_3 and the other 16-trits are extracted from the U/V quantities. Let us consider the U/V quantities concatenated together and viewed as trits to be:

$$U||V = u_7||u_6||u_5||u_4||u_3||u_2||u_1||u_0||v_7||v_6||v_5||v_4||v_3||v_2||v_1||v_0 \quad (4.89)$$

In which the u_i and v_i are 1-trit quantities.

We initialize a round key to be a 32-trit value in which all trits start with the

value of 00_3 for each trit value b_i and we consider the round key to be:

$$RK = b_{31}||b_{30}||b_{29}||b_{28}||b_{27}||b_{26}||b_{25}||b_{24}||b_{23}||b_{22}||b_{21}||b_{20}||b_{19}||b_{18}||b_{17}||b_{16}|| \\ b_{15}||b_{14}||b_{13}||b_{12}||b_{11}||b_{10}||b_9||b_8||b_7||b_6||b_5||b_4||b_3||b_2||b_1||b_0 \quad (4.90)$$

Using our U/V quantities we store the 16-trits of active key information into the round key as follows:

$$b_{2*i+16} \leftarrow v_i, \;\; b_{2*i} \leftarrow u_i, \;\; \forall i \in \{0, 1, ..., 7\} \quad (4.91)$$

After the extraction of the U/V values into the round key this round is complete and the key state will need to be updated in preparation for the creation of the next round key. Consider the key state in binary as a concatenation of 16 4-trit values:

$$KS = ks_{15}||ks_{14}||ks_{13}||ks_{12}||ks_{11}||ks_{10}||ks_9||ks_8|| \\ ks_7||ks_6||ks_5||ks_4||ks_3||ks_2||ks_1||ks_0 \quad (4.92)$$

The key state is updated as follows in which $ks_i <<_3 j$ and $ks_i >>_3 j$ denotes a left/right trit shift (not-cyclical) of ks_i by j trits, $||$ denotes concatenation and $|$ denoted the OR operation which in this use case is the same as addition:

$$KS \leftarrow (ks_3 >>_3 1|ks_2 <<_3 3)||(ks_2 >>_3 1|ks_3 <<_3 3)||(ks_0 >>_3 2|ks_1 <<_3 2)|| \\ (ks_1 >>_3 2|ks_0 <<_3 2)||ks_{15}||ks_{14}||ks_{13}||ks_{12}|| \\ ks_{11}||ks_{10}||ks_9||ks_8||ks_7||ks_6||ks_5||ks_4 \quad (4.93)$$

In our implementation we did not add round constants unlike what they do in the GIFT64 scheme (Banik et al., 2017). This process is repeated for the other 27 round keys.

4.8 Case Study Ternary S-Boxes

In our case study, we exhaustively search all bijective S-Boxes in 3^2 and classify all 9! S-Boxes into permutation modular addition equivalence classes (PE) and affine equivalence classes (AE).

Within the PE classes there exist 1,162 classes and their distribution is shown in Table 4.8 below: We note that within the PE classes the PE class size is always

Table 4.8: Ternary S-Box PE Class Distribution

Size	Number of Classes
18	2
36	11
54	4
108	30
162	14
324	1101

divisible by the number of inner PE transformation that is $2! * 3^2 = 18$ in which 3^2 the number of elements in $\mathbb{F}_3^2$ and 2! is the number of trit permutation of degree 2. Similarly, we do the same for the affine classes and their results are shown below in which there exist 9 affine classes in our 3^2 case study:

Similar to the PE class, all of the AE class sizes are divisible by $48 * 3^2 = 432$ which is equal to the number of inner transformations in AE because there are 48 invertible 2x2 matrices with elements in $\mathbb{F}_3$ and 3^2 are the number of elements in $\mathbb{F}_3^2$

We will now list the S-Boxes in AE/PE classes with S-Box representatives, AE/PE

Table 4.9: Ternary S-Box AE Class Distribution

Size	Number of Classes
432	1
3,456	1
11,664	1
15,552	1
20,736	1
62,208	2
93,312	2

size and class number, metric scores, and CPA status. We denote Success to be successful after a large amount of trances meaning that the mean SR of that given execution is larger than 10%. In our case study we conduct our experiments with a fixed trace/plaintext input of 400 pairs. Fail will denote never being able to recover the full private key given 400 traces which mans that the mean SR of Fail S-Boxes is 0%. Near Success Denotes having some success in recovering the private key and the mean SR value is in the interval of 1% to 10%.

4.9 Ternary GIFT: A Case Study

In the binary case the transformation of a theoretical structure of a SPN cipher to be coded into a micro-controller of FPGA, in our ternary case study it is not straight forward and in this section we will be stating details we used in our implementation of our ternary case study using base 2 micro-controllers.

4.9.1 Software Implementation of Ternary Values

Firstly we wanted to keep the concatenation of data form trits to a collection of trits so we consider a trit to be a 2-bit number. We chose our two bit values carefully so that each of the trits have a different Hamming weight in the base 2 implementation.

Table 4.10: Ternary S-Boxes with PE Classes and Metrics

	PE Class Number	PE Class Size	S-Box	CPA Status	NL	SNR	DPA SNR	TO	RTO	CC
PE Classes with multiple representatives										
S1	273	324	254013768	Success	1	20.749	18.342	3.417	1.53	1.239
S16			062835417	Fail	1	20.075	18.342	3.417	1.53	0.113
S17			102647853	Fail	1	9.348	22.045	3.417	1.53	0.338
S18			306281754	Fail	1	17.907	18.342	3.417	1.53	0.338
S19	519	324	042685713	Success	1.267	17.986	22.045	3.417	1.53	0.338
S20			157820634	Success	1.267	12.449	18.343	3.417	1.53	1.239
S21			340781625	Fail	1.267	20.638	18.342	3.417	1.53	0.113
S3			013247865	Fail	1.267	23.474	18.342	3.417	1.53	1.239
PE Classes with two representatives										
S2	1160	36	048561723	Success	0	25.904	12.727	2.25	2	1.014
S23			570624138	Success	0	28.731	12.727	2.25	2	1.014
S8	589	324	854013762	Success	1	14.439	18.342	3.417	1.53	1.239
S24			074823516	Fail	1	13.617	19.092	3.417	1.53	1.239
S12	409	324	184306275	Success	1	21.016	18.342	3.25	1.53	0.113
S25			073265481	Success	1	17.555	18.342	3.25	1.53	0.338
S13	108	324	184723056	Success	1	16.713	22.045	3.417	1.53	0.338
S26			051237648	Success	1	20.398	18.343	3.417	1.53	0.113
S15	608	324	184730625	Near Success	1	12.522	18.342	3.417	1.53	1.239
S27			073562418	Success	1	27.326	18.342	3.417	1.53	1.239
S9	953	324	548061723	Success	0	13.401	22.045	3.417	2	1.014
S28			061548723	Near Success	0	21.31	22.045	3.417	2	1.014
S11	1035	324	184256703	Success	1	16.694	18.342	3.25	1.53	0.338
S29			064817532	Near Success	1	19.109	18.342	3.25	1.53	0.113
S4	1004	324	013652748	Fail	1.267	33.538	18.342	3.25	1.53	1.239
S30			064781352	Success	1.267	11.588	18.342	3.25	1.53	0.113
S5	2	108	012345786	Fail	0	11.551	15.558	2.5	2	0.338
S31			186420753	Success	0	18.102	15.558	2.5	2	0.338
S6	517	324	013246785	Fail	1	26.499	22.045	3.75	1.333	2.126
S32			065348127	Fail	1	9.619	22.045	3.75	1.333	2.126
S7	151	324	178026354	Fail	1	8.339	19.091	3.417	1.53	1.014
S33			056124783	Success	1	14.009	18.343	3.417	1.53	0.338
S10	302	324	184027356	Fail	1	13.149	19.342	3.417	1.53	1.239
S34			062781453	Success	1	14.739	18.343	3.417	1.53	0.113
S14	131	324	184725063	Fail	1	23.865	18.342	3.417	1.53	0.113
S35			056174283	Fail	1	17.922	18.342	3.417	1.53	1.239
S22	311	324	012487563	Success	1	12.341	19.091	3.333	1.666	0.638
S36			058327614	Fail	1	17.759	19.091	3.333	1.666	0.413

Table 4.11: Ternary S-Boxes with AE Classes and Metrics

	AE Class Number	AE Class Size	S-Box	CPA Status	NL	SNR	DPA SNR	TO	RTO	CC
AE Classes with multiple representatives										
S1	4	62,208	254013768	Success	1	20.749	18.342	3.416	1.53	1.239
S13			184723056	Success	1	16.713	22.045	3.417	1.53	0.338
S26			051237648	Success	1	20.398	18.343	3.417	1.53	0.113
S16			062835417	Fail	1	20.075	18.342	3.417	1.53	0.113
S17			102647853	Fail	1	9.348	22.045	3.417	1.53	0.338
S14			184725063	Fail	1	23.865	18.342	3.417	1.53	0.113
S18			306281754	Fail	1	17.907	18.342	3.417	1.53	0.338
S35			056174283	Fail	1	17.922	18.342	3.417	1.53	1.239
S19	7	62,208	042685713	Success	1.267	17.986	22.045	3.417	1.53	0.338
S20			157820634	Success	1.267	12.449	18.343	3.417	1.53	1.239
S21			340781625	Fail	1.267	20.638	18.342	3.417	1.53	0.113
S3			013247865	Fail	1.267	23.474	18.342	3.417	1.53	1.239
S8	3	93,312	854013762	Success	1	14.439	18.342	3.417	1.53	1.239
S33			056124783	Success	1	14.009	18.343	3.417	1.53	0.338
S7			178026354	Fail	1	8.339	19.091	3.417	1.53	1.014
S24			074823516	Fail	1	13.617	19.092	3.417	1.53	1.239
S12	5	93,312	184306275	Success	1	21.016	18.342	3.25	1.53	0.113
S15			184730625	Near Success	1	12.522	18.342	3.417	1.53	1.239
S25			073265481	Success	1	17.555	18.342	3.25	1.53	0.338
S27			073562418	Success	1	27.326	18.342	3.417	1.53	1.239
S34			062781453	Success	1	14.739	18.343	3.417	1.53	0.113
S29			064817532	Near Success	1	19.109	18.342	3.25	1.53	0.113
S11			184256703	Success	1	16.694	18.342	3.25	1.53	0.338
S6			013246785	Fail	1	26.499	22.045	3.75	1.333	2.126
S10			184027356	Fail	1	13.149	19.342	3.417	1.53	1.239
S32			065348127	Fail	1	9.619	22.045	3.75	1.333	2.126
AE Classes with two representatives										
S2	0	432	048561723	Success	0	25.904	12.727	2.25	2	1.014
S23			570624138	Success	0	28.731	12.727	2.25	2	1.014
S28	1	15,552	061548723	Near Success	0	21.31	22.045	3.417	2	1.014
S9			548061723	Success	0	13.401	22.045	3.417	2	1.014
S30	8	20,736	064781352	Success	1.267	11.588	18.342	3.25	1.53	0.113
S4			013652748	Fail	1.267	33.538	18.342	3.25	1.53	1.239
S31	2	3,456	186420753	Success	0	18.102	15.558	2.5	2	0.338
S5			012345786	Fail	0	11.551	15.558	2.5	2	0.338
S22	6	11,664	012487563	Success	1	12.341	19.091	3.333	1.666	0.638
S36			058327614	Fail	1	17.759	19.091	3.333	1.666	0.413

Our conversion from ternary trit to a 2 bit binary coded trit is shown below:

$$0_3 = 00_2 \quad 1_3 = 01_2 \quad 2_3 = 11_2 \tag{4.94}$$

Since S-Boxes in our case study are $\mathbb{F}_3^2 \rightarrow \mathbb{F}_3^2$ there are 9 'active' BCT values and 7 values that denote error. The conversion from elements in $\mathbb{F}_3^2$ to binary nibbles is shown below:

$$\{0, 1, 2, 3, 4, 5, 6, 7, 8\} \longrightarrow \{0, 1, 3, 4, 5, 7, c, d, f\} \tag{4.95}$$

Because the 10_2 is not used in our implementation that leads to the 7 error conditions so the nibbles $\{2, 6, 8, 9, a, b, e\}$ because either the left two or right two bits are the error value 10_2 when a nibble is represented in binary.

Because we can concatenate two trits to a nibbled binary, from that it is straightforward to concatenate two BCT nibbles to a BCT byte and concatenate a BCT byte to a 64-bit (32-trit) block that will be used in our ternary GIFT cipher.

This implementation leads to trivial permTrits as instead of permuting 1-bit values we permute 2-bit BCT values. Similarly for the key scheduling algorithm instead of dealing with bits it deals with BCT trits but the conversion is trivial. As for the S-Box layer, if the input is a BCT nibble the output will also lead to a BCT nibble based upon the S-Box look up table in the BCT format.

We would also like to note that the implementation of the BCT effects the side channel susceptibility. Originally we had the BCT values be 0b00, 0b01, and b010 but we were not able to have any success in CPA attacks while using thousands of plaintext/trace pairs. While we only tried a few S-Boxes with this older implementation we stuck with the BCT described earlier because in each BCT value has a different HW compared to our original implementation in which two of the three trits have identical HW in the implementation for micro-controllers.

4.9.2 Software Implementation of Ternary Addition

Because during the encryption process the traces need to be synchronized, during the coding process of the ternary GIFT scheme we want to avoid if statements because it will change the locations of the peaks and troughs during the collection of the power trance based upon the input to the scheme. This is not a big deal for the most part during the implementation except when we will need to perform ternary modular addition during the add round key sub-round function. This is difficult because we represent the trits as BCT values but when we perform addition it needs to be evaluated as ternary addition. In order to solve this problem instead of using if statements we created a look up table on all possible pairs of BCT values and the lookup table holds the ternary sum of the given BCT values.

4.9.3 Error Cases of Inputs in Software Implementation

Because of the error conditions of the 10_2 valued BCT, when given input to plaintext or key to the cryptographic scheme the inputs must be sanitized. This is needed because during data collection we generate a random 64-bit plaintext and in some cases we change the 64-bit key but we need to make sure when choosing these values randomly that they are in the BCT from expressed earlier in this section. In order to do this the sanitizing function takes in a 64-bit or 128-bit values and extract all 2-bit values and checks if they are the 'error' value of 10_2. If it is the error value we replace the value with one of the other 3 valid choices. For this case study any input of 10_2 is replaced with 00_2 and those values that the sanitization occur on are keys and plaintext values. In doing so we avoid the error conditions as input and because of the BCT coding in the implementation it will not have BCT error conditions in intermediary states in the cipher because of input sanitization.

The follwing was added to the dissertaion to address prlaintext generation:

4.9.4 Algebraic Normal Form

Algebraic Normal Form (ANF) is a way to represent the component functions of S-Boxes as polynomials modulo p which in our case is $p = 3$ for our case study. ANF representation of binary S-Boxes are well studies such as the work of (Picek et al., 2014a; Bakoev, 2017) but we will be extending their work to represent the S-Boxes of $GF(3^2) \rightarrow GF(3^2)$. A S-Box in 3^2 can be viewed as a concatenation of component functions in which each component function is a function of the form $GF(3^2) \rightarrow GF(3)$.

A classical binary case of component functions $GF(2^4) \rightarrow GF(2)$ in which the ANF form of the component functions take the form of:

$$f_1(x_1, x_2, x_3, x_4) = c_0 + c_1x_1 + c_2x_2 + c_3x_3 + c_4x_4 + c_5x_1x_2 + c_6x_1x_3 + c_7x_1x_4 + c_8x_2x_3 + c_9x_2x_4 + c_{10}x_3x_4 + c_{11}x_1x_2x_3 + c_{12}x_1x_2 + x_4 + c_{13}x_1x_3x_4 + c_{14}x_2x_3x_4 + c_{15}x_1x_2x_3x_4 \text{ modulo } 2 \quad (4.96)$$

In which the constants to the monomials are $c_i \in \mathbb{Z}_2$ and there are 2^4 different coefficients to consider.

In the case of component functions in $GF(3^2) \rightarrow GF(3)$ the structure is similar in which they are $3^2 = 9$ coefficients, but the ANF polynomials are in the form:

$$f_1(x_1, x_2) = c_0 + c_1x_1 + c_2x_1^2 + c_3x_2 + c_4x_2^2 + c_5x_1x_2 + c_6x_1^2x_2 + c_7x_1x_2^2 + c_8x_1^2x_2^2 \text{ modulo } 3 \quad (4.97)$$

In which the $c_i \in \mathbb{Z}_3$.

The ANF form can also be described as a summation

$$f(x_1, x_2) = \sum_{u \in \mathbb{F}_3^2} a_u x^u \text{ modulo } 3$$

Where $x^u = x_1^{u_1} x_2^{u_2}$ and $a_u \in \mathbb{F}_3$

Any component function of an S-Box can be represented in ANF form and when done can give valuable information of the algebraic construction of the component function such as Min/Max degree and a representation of the non-linearly of the function.

Let us consider an example using the ternary S-Box S1. The S-Box is defined as

Table 4.12: Ternary GIFT Sample

x	00	01	02	10	11	12	20	21	22
S1(x)	02	12	11	00	01	10	21	20	22

We will now separate the S-Box into it's component functions:

Table 4.13: Ternary GIFT Sample

x_1x_2	00	01	02	10	11	12	20	21	22
$f_1(x_1, x_2)$	2	2	1	0	1	0	1	0	2
$f_2(x_1, x_2)$	0	1	1	0	0	1	2	2	2

The ANF polynomials for these two component functions are:

$$f_2(x_1, x_2) = x_1^2 + 2x_2 + x_2^2 + 2x_1xx_2 + x_1^2x_2 + 2x_1x_2^2 \text{ modulo } 3 \tag{4.98}$$

In which we can apply the 9 options for input to the component functions:

$$f_2(0, 0) = 0 \tag{4.99}$$

$$f_2(1,0) = (1)^2 \text{ modulo } 3 \ = 1 \tag{4.100}$$

$$f_2(2,0) = (2)^2 \text{ modulo } 3 \ = 1 \tag{4.101}$$

$$f_2(0,1) = 2 * (1) + (1)^2 \text{ modulo } 3 \ = 0 \tag{4.102}$$

$$f_2(1,1) = (1)^2 + 2 * (1) + (1)^2 + 2 * (1) * (1) + (1)^2 * (1) + 2 * (1) * (1)^2 = 9 \text{ modulo } 3 \ = 0 \tag{4.103}$$

$$f_2(2,1) = (2)^2 + 2(1) + (1)^2 + 2(2)(1) + (2)^2(1) + 2(2)(1)^2 \text{ modulo } 3 \ = 1 \tag{4.104}$$

$$f_2(0,2) = 2(2) + (2)^2 \text{ modulo } 3 \ = 2 \tag{4.105}$$

$$f_2(1,2) = (1)^2 + 2(2) + (2)^2 2(1)(2) + (1)^2(2) + 2(1)(2)^2 \text{ modulo } 3 \ = 2 \tag{4.106}$$

$$f_2(2,2) = (2)^2 + 2(2) + (2)^2 + 2(2)(2) + (2)^2(2) + 2(2)(2)^2 \text{ modulo } 3 \ = 2 \tag{4.107}$$

This example illustrated the 9 possible inputs to the component function f_2 of S1 and shows the ANF form equals the component function in Table 4.13.

For our use we use ANF to find the min/max degree of the S-Box but it also can be used to algebraically describe the S-Box.

4.10 Ternary GIFT Correlation Power Analysis Attack

Show how we adapted the CPA methodology to work in the ternary setting. In order for fair evaluation across S-Boxes data was gathered in the following way.

One constant private key was used across all of our ternary experiments. That constant private key was the sanitized default ChipWhisperer default key shown below in base 10 form as BCT:

$$PrivateKey : [3, 124, 21, 20, 0, 12, 208, 4, 3, 247, 21, 0, 1, 207, 79, 60] \tag{4.108}$$

Which is also mathematically viewed as each byte being 4 trits as shown in below in the ternary form:

$$PrivateKey_3 : [0002, 1220, 0111, 0110, 0000, 0020, 2100, 0010, 0002, 2212, 0111, 0000, 0001, 2022, 1022, 0220] \tag{4.109}$$

For each S-Box we gathered a pool of 5,000 plaintext/trace pairs for use in our CPA attack. Because of our ChipWisperer Lite's restrictions we are only able to gather around 23,000 voltage samples per trace so we must gather 4 separate data pools, one for each of the first 4 round keys we are attacking and we use the 23,000

samples per trace in all of our experiments. The plaintexts were gathered using the ChipWisperer's ktp.next() call (O'Flynn and Chen, 2014) using the ChipWhisperer python library and then the plaintexts were truncated to be the correct bit length and then are sanitized and used is input for encryption on our target device. Our experiments for each S-Box require 5,000 plaintexts per pool and 4 pools for each S-Box resulting in 20,000 plaintexts gathered per S-Box. Because we have 36 S-Boxes in our case study that leads to 720,000 plaintext gathered for our case study.

$$5,000 * 4 * 36 = 720,000 \tag{4.110}$$

It is worth noting that since each plaintext is chosen randomly the pools plaintext are going to differ being the pools have different plaintexts during the data collection process. We assume that the attacker has knowledge of the cryptographic scheme and it's implementation along with the 4 data pools gathered for each S-Box.

4.10.1 Correlation Power Analysis Attack Definitions in Ternary Case Study

In our case study not every S-Box was able to successfully recover the private key using the CPA attack. For S-Boxes that were successful in extracting the private key we can use the mean success rate to perform the comparison on the resistance/susceptibility to attack. For those S-Boxes that performed poorly during the CPA attack we will use Guessing Entropy to compare the results of the CPA attack like they do in the work of (Biryukov et al., 2016). Cite from theory to practice paper and give justification on any changes/additions/reductions we make.

Similar to the base 2 chapter we run 100 experiments in our computation as one

Algorithm 4: Experiment Pseudocode in Ternary Case Study

```
Result: SR and GE for each trace count
count = 0;
successCount = 0;
list initialized;
SRresults structure initialized;
GEresults structure initialized;
while count < Threshold do
    Add random plaintext-voltage array pair to list;
    Conduct CPA attack using list;
    count++;
    results[count] = CPA Success Rate (1/0);
    GEresults[count] = Mean GE 0 ≤ x ≤ 2
    if Successful then
        successCount++;
        if successCount == 5 then
            Mark remaining SR results successful;
            Mark remaining GE results 0.0;
            return SR/GE results;
        end
    end
    else
        successCount = 0;
    end
end
return SR/GE results;
```

trail and a trial in our ternary case study is defined in Algorithm 5

4.10.2 Correlation Power Analysis Attack in Ternary Case Study

Similar to the binary CPA attack, the ternary attack on our Ternary GIFT implementation attacks sub-keys nibbles at a time. One difference we note as in the binary case we have to consider all 16 possible nibbles, in our ternary case study we only have to consider 9 because the other nibbles are error conditions. We still use the

Algorithm 5: Ternary Trial Pseudocode

```
Result: Mean Success Rate for Each Trace Count
count = 0;
SRresults structure initialized;
GEresults structure initialized;
while count < 100 do
    Execute Experiment;
    Store results of experiment in the SR/GE results structure;
    count++;
end
Average the results of the experiments for each trace count;
Return the mean GE/SR results;
```

hamming weight model and in doing so we consider the hamming weight of the BCT nibble values. The correlation computation is the same as in the binary case.

In the binary case, given enough traces we were always able to compute the private key in less than 250 traces. In the ternary case we have S-Boxes that are able to reach 100% mean success rate in 400 traces, some that are able to have mean success rate larger than 0% but never reach 100%, and some S-Boxes never were successful in extracting the key in our case study. We use two leakage models to compare the CPA attack results on our ternary S-Boxes. The models are mean success rate and mean guessing entropy. The mean success rate is used to compare the CPA attack results of successfully extracting the key and the mean guessing entropy is used to compare CPA results when we are unable to recover the key using the given S-Box. Both of these metrics give numerical results that we can use and analyze to compare a S-Boxes resistance/susceptibility to cryptographic power analysis attacks.

4.11 Evaluation Framework

4.11.1 Measurement Setup

All experiments reported in this chapter were performed on target board being an 8-bit ATXMEGA micro-controller mounted on a CW308 UFO board. The control board in our experiments is a CW-Lite FPGA that acts both as a control board for our experiments and as a low cost oscilloscope. The control board has two connections to the target board being a 20-pin ribbon cable for data transfer and a SMA cable to measure voltage draw on the target device.

We mount the CPA attacks on a 8-bit native data type TernaryGIFT written in C that runs as an extension of the CW simple serial protocol that is available for use with Chip Whisperer software/hardware by NewAE. Within the C code also exist a trigger that indicates to the oscilloscope to begin/end the trace. The only modification to the C source code across our experiments is the S-Box look up tables. In order for fair comparison the attacker needs to extract the same 128-bit private key being the CW default key which is $0x037C1514000CD00403F7150001CF4F3C$. Also in all of our experiments in our ternary case study we use the same number of trace/plaintext pairs being $q = 400$.

4.11.2 Experimental Result Metrics

In order to have a fair and uniform test we use some evaluation methodologies from (Standaert et al., 2009). In this case study we use three different evaluation metrics to experimentally measure how much information an attacker recovers when performing the CPA attack. These metric measure if the attacker recovers the key, the statistical positions of the correct sub-keys, and the difference of the correction coefficients in

each possible sub-keys. Trough these metrics we experimentally measure how much information an attacker gains during the CPA attack.

We recall that during the CPA attack, the attacker computes each of the sub-keys separately in order to recover the round key. We use the selection function $\phi(x, k)$ to be the output of the S-Box functions when the input to the S-Box is $x \oplus_p k$ where k is the sub-key and x is a plaintext state. The following three metrics quantify the information the attacker gains during the CPA attack as well is knowing these values the amount of effort/computational time the attacker needs to break the private key.

Success Rate

Extending from the work of (Biryukov et al., 2016) we denote the Success Rate of given a CPA attack the SR of an attack is one if the attack successfully computed the private key and is 0 if the attack does not compute the private key. This is noted in the formula below:

$$SR = \begin{cases} 1, \text{If derived key is the correct key} \\ 0, \text{Otherwise} \end{cases} \tag{4.111}$$

As we compute this over many experiment we note that the mean success rate is an average over many experiments leading to a percentage value. We run experiments 100 times so each experiment success/failure is noted as a single percentage point. In this description is a "all or nothing" compared with models such as GE which indicate how much of the key was successfully recovered.

Guessing Entropy

In order to compare the S-Boxes that had low mean SR even with a large amount of traces we use Guessing Entropy (GE) to compare the results of the CPA attack. GE was fist noted in the work of (Massey, 1994). Let K be the set of all possible sub-keys. Let G = $(g_0, g_1, ..., g_{|K|-1})$ be the CPA ranking during the correlation computation and the g values be sorted by their correlation value and let k^* be the known sub-key value.

$$GE(k^*, G) = i, \text{ such that } k^* = g_i \text{ for } g_i \in G \tag{4.112}$$

In our ternary case study there are only 3 options for each sub-key so the values for GE are either 0,1, or 2. Zero denotes that the sub-key was recovered correctly and 2 denotes that the correct sub-key was ranked last, while 1 is in the middle of the two conditions. We compute this GE computation for each sub-key in the round key and for the first 4 round keys of our ternary GIFT cryptosystem and then take the average to compute the mean GE values. Because both GE and SR are leakage models dependent on CPA attack we can compute them both at the same time when performing CPA computations.

Upon observation if the mean success rate reaches 100% the mean GE will result in 0.0. If one is choosing sub-keys randomly the GE should tend towards 1.0 given enough computations. If the mean GE is larger than 1 and the attacker is aware, the attacker can invert all sub-key guess list and then have mean GE that is less than 1 if there attacking scheme is performing worse than random guessing. We apply the GE computation on experiments that give a mean GE and SR computation for a

given trace counts on inputs to the CPA attack and the definition of an experiment is shown in Algorithm 4.

Delta Value

Definition 4.1 *Introduced in the work of (Biryukov et al., 2016), the computation of the δ value measures the difference in the correct sub-key correlation value with the most likely sub-key value. We denote k^* to be the correct sub-key value and $k^\diamond$ to be the most likely sub-key. The definition is split up into two sub-cases on if the correct sub-key is ranked highest or not. Similar to GE we rank the set of all sub-keys K into an ordered list $G = (g_0, g_1, ..., G_{|K|-1}$ based upon the CPA correlation value. We note for any given sub-key g_i that c_{g_i} is the correlation value of the sub-key g_i during the CPA attack.*

Case 1: Correct Sub-key has the highest correlation value

In this case we describe the computation of δ when the correct key has the highest correlation value. We let $k^\diamond = g_1$ as $k^* = g_0$ and we compute the delta value shown below.

$$\delta = c_{k^*} - c_{k^\diamond} \tag{4.113}$$

We note that in this case the value $\delta \geq 0$.

Case 2: Correct Sub-key does not have the highest correlation

In this case the correct sub-key does not have the highest correlation. We let $k^\diamond = g_0$ and $k^* = g_i$ for $1 \leq i \leq |K| - 1$. Similar to the other case δ as computed as follows:

$$\delta = c_{k^*} - c_{k^\diamond} \tag{4.114}$$

We note that in this case the value of $\delta \leq 0$.

This δ *value measures how strong the correlation is compared to other correlation values and gives us a better measurement of success of partial key recovery We take the mean values of these experimental result metrics over 100 executions using 400 trace/plaintext pairs randomly chosen from a the data pool.*

4.12 Quantifying Power Leakage

Using the measurements of mean SR/GE/Delta results, we quantify the leakage of different S-Boxes to analyze which are more resistant and which are more susceptible to cryptographic power analysis attacks. Based upon our results we will show which ternary S-Box in our study will be the 'best' for real use in the setting of defense against cryptographic side channel attacks. For the measurements we consider plaintext/trace pairs gathered experimentally on our underlying hardware. Our measurements were all gathered under the same known cryptographic key to have a uniform execution environment in our collection. Our plaintexts were randomly generated 32-trit values.

We followed the evaluation methodology of (Biryukov et al., 2016) we define our CPA evaluation a collection of experiments. Experiment are described in the pseudocode below in Algorithm 6:

To understand and compare the leakage of S-Boxes in our ternary setting, many S-Boxes were collected. All of our S-Box leakage functions consider the output of a given S-Box, in which the input to the S-Box is a plaintext state modular added with a sub-key. Based upon our experimental results we categorized the S-Boxes using mean SR and we compare the experimental results shown in Table 4.14 and

Algorithm 6: Experiment Pseudocode in Ternary Case Study

```
Result: SR and GE for each trace count
count = 0;
SRresults structure initialized;
GEresults structure initialized;
Deltaresults structure initialized;
while count < Threshold do
    Pick 400 random plaintext/trace pairs from the data pool;
    Conduct CPA attack using the randomly chosen pairs;
    Store the resulting SR/GE/Delta values in the data structures;
    count++;
end
return mean SR/GE/Delta results;
```

Table 4.15.

4.12.1 Understanding the Leakage

The leakage in our case study comes from the Hamming weight (HW) of the values that are stored after the S-Box calls in the cryptographic scheme. Based upon the Hamming weight, on average a higher hamming weight will have more power draw and a lower Hamming weight will have less power draw. Because the POI of the cipher is the output of the S-Box, the chosen S-Box function will influence the power draw of the cipher. In our case study we compare different S-Box with the rest of the cipher being fixed for fair evaluation and we measure our experimental results for each S-Box to compare the leakage of the S-Box within the cipher.

4.12.2 Comparison of Different Ternary S-Boxes

In our ternary case study we chose 36 S-Boxes randomly in which the first few were input randomly from the keyboard other remaining S-Boxes in the first 15 were chosen to have varying S-Box metric scores with multiple representatives for each score. The other 21 were chosen to be in PE classes similar to the first 15 S-Boxes. The

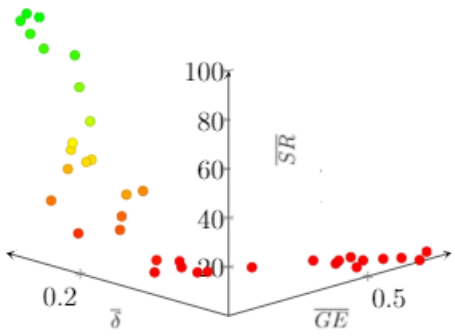

Figure 4.1: A 3D Figure comparing the Experimental Results

other 21 S-Boxes were chosen so that each represented PE/AE class has at least 2 representatives. In order to gather the experimental results, we used Boise State's R2 computation cluster. We compare our S-Boxes used in our case study with mean SR/GE/δ values. After the trials were computed containing 100 experiments in each trial we have our initial results in Table 4.14. We also note that the selection functions used in Table 4.14 is $\varphi_i(x \oplus_3 k) = S_i(x \oplus_3 k)$ for $i \in \mathbb{Z}_{36}$. We show our results in different tables to compare metric scores, experimental results, and mathematical equivalence classes. In Figure 4.1 shows the relationship between the experimental results SR/GE/Delta. The 3D figure has axes for SR/GE/Delta as well as having a color gradient corresponding to the SR value that takes a gradient of the colors green, yellow, and red.

We also represent the experimental results in a 2D graph with color representing a third dimension shown in the figure below: This figure also shows a regression line that indications a line of the equation:

$$y = -0.3349x + 0.2409 \tag{4.115}$$

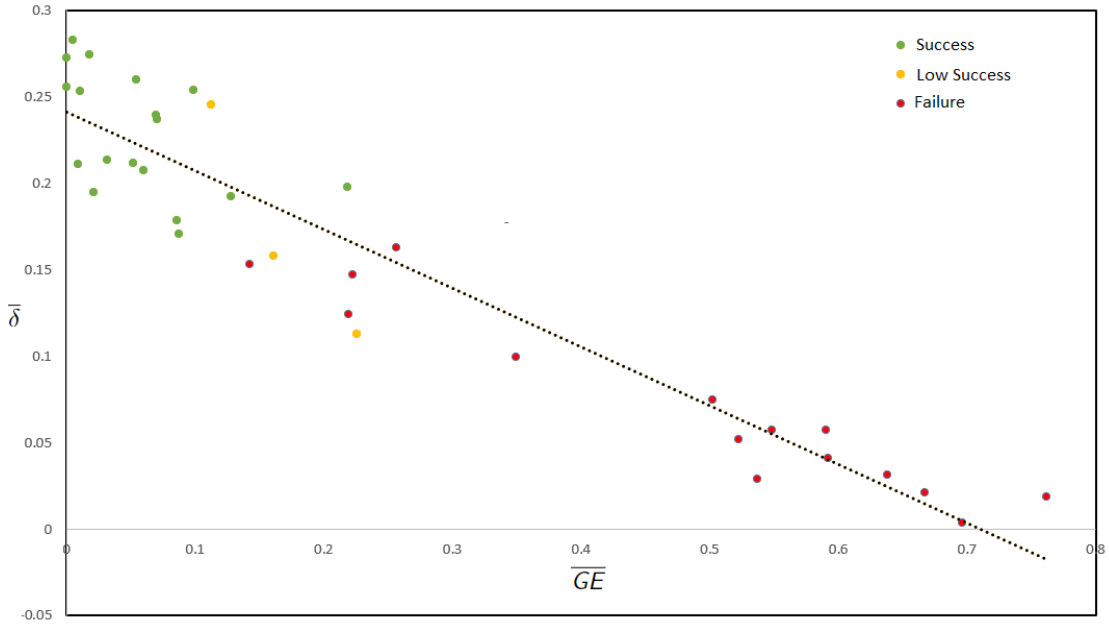

Figure 4.2: Experimental Results

In this figure you can clearly see that there is correlation between all three experimental results and the exact values of that correlation are also listed in a table later on. Note the clustering of the data, in the top left of the graph all S-Boxes are more susceptible to CPA attack and in the bottom right there are clusters of S-Boxes that are more resistant to attack. We will use this observation of clustering to make categories for the GE and δ results.

The following tables will compare experimental results with one another.

4.12.3 Correlation of Metric Values

We noted that SR is the 'gold standard' that measures if the private key was successfully recovered or not. GE and δ also measure the success of the CPA in different ways. We computed correlation values using the Pearson's correlation coefficient described in the introduction and used that to compute the correlation of all metric

Table 4.14: Experimental Results

	S-Box	$\overline{SR}$	$\overline{GE}$	$\overline{\delta}$	NL
S1	254013768	62%	0.02125	0.19477	1
S2	048561723	88%	0.00906	0.21088	0
S3	013247865	0%	0.677	0.0211	1.267
S4	013652748	0%	0.5375	0.02914	1.267
S5	012345786	0%	0.3492	0.0997	0
S6	013246785	0%	0.6964	0.0039	1
S7	178026354	0%	0.5925	0.0413	1
S8	854013762	100%	0.0	0.27255	1
S9	548061723	23%	0.05484	0.2601	0
S10	184027356	0%	0.7617	0.01899	1
S11	184256703	35%	0.09859	0.2539	1
S12	184306275	91%	0.018125	0.274493	1
S13	184723056	27%	0.2184	0.19759	1
S14	184725063	0%	0.502	0.07451	1
S15	184730625	9%	0.1128	0.24523	1
S16	062835417	0%	0.2194	0.1244	1
S17	102647853	0%	0.2229	0.1471	1
S18	306281754	0%	0.63828	0.0312	1
S19	042685713	87%	0.01047	0.25336	1.267
S20	157820634	96%	0.005467	0.28289	1.267
S21	340781625	0%	0.25672	0.16253	1.267
S22	012487563	45%	0.0702	0.2391	1
S23	570624138	100%	0.0	0.2555	0
S24	074823516	0%	0.1425	0.1534	1
S25	073265481	74%	0.0320	0.21345	1
S26	051237648	32%	0.0879	0.17089	1
S27	073562418	20%	0.1278	0.19339	1
S28	061548723	4%	0.1611	0.1578	0
S29	064817532	1%	0.2258	0.11291	1
S30	064781352	48%	0.0703	0.23703	1.267
S31	186420753	17%	0.0856	0.1787	0
S32	065348127	0%	0.5228	0.0519	1
S33	056124783	44%	0.06	0.20738	1
S34	062781453	43%	0.0519	0.2114	1
S35	056174283	0%	0.5481	0.0572	1
S36	058327614	0%	0.5905	0.0571	1

Table 4.15: Experimental Results with Leakage Class

	CPA Status	$\overline{SR}$	$\overline{GE}$	$\overline{\delta}$
S1	Success	62%	0.02125	0.19477
S2	Success	88%	0.00906	0.21088
S3	Fail	0%	0.677	0.0211
S4	Fail	0%	0.5375	0.02914
S5	Fail	0%	0.3492	0.0997
S6	Fail	0%	0.6964	0.0039
S7	Fail	0%	0.5925	0.0413
S8	Success	100%	0.0	0.27255
S9	Success	23%	0.05484	0.2601
S10	Fail	0%	0.7617	0.01899
S11	Success	35%	0.09859	0.2539
S12	Success	91%	0.018125	0.274493
S13	Success	27%	0.2184	0.19759
S14	Fail	0%	0.502	0.07451
S15	Near Success	9%	0.1128	0.24523
S16	Fail	0%	0.2194	0.1244
S17	Fail	0%	0.2229	0.1471
S18	Fail	0%	0.63828	0.0312
S19	Success	87%	0.01047	0.25336
S20	Success	96%	0.005467	0.28289
S21	Fail	0%	0.25672	0.16253
S22	Success	45%	0.0702	0.2391
S23	Success	100%	0.0	0.2555
S24	Fail	0%	0.1425	0.1534
S25	Success	74%	0.0320	0.21345
S26	Success	32%	0.0879	0.17089
S27	Success	20%	0.1278	0.19339
S28	Near Success	4%	0.1611	0.1578
S29	Near Success	1%	0.2258	0.11291
S30	Success	48%	0.0703	0.23703
S31	Success	17%	0.0856	0.1787
S32	Fail	0%	0.5228	0.0519
S33	Success	44%	0.06	0.20738
S34	Success	43%	0.0519	0.2114
S35	Fail	0%	0.5481	0.0572
S36	Fail	0%	0.5905	0.0571

Table 4.16: Branch and Degree of Sampled S-Boxes

	DBN	LBN	MinDegree	MaxDegree
S1	2	2	3	3
S2	3	3	1	1
S3	2	2	3	3
S4	2	2	2	3
S5	2	2	1	2
S6	2	2	3	3
S7	2	2	3	3
S8	2	2	3	3
S9	2	2	3	3
S10	2	2	3	3
S11	2	2	2	3
S12	2	2	2	3
S13	2	2	3	3
S14	2	2	3	3
S15	2	2	3	3
S16	2	2	3	3
S17	2	2	3	3
S18	2	2	3	3
S19	2	2	3	3
S20	2	2	3	3
S21	2	2	3	3
S22	2	2	3	3
S23	3	3	1	1
S24	2	2	3	3
S25	2	2	2	3
S26	2	2	3	3
S27	2	2	3	3
S28	2	2	3	3
S29	2	2	2	3
S30	2	2	2	3
S31	2	2	1	2
S32	2	2	3	3
S33	2	2	3	3
S34	2	2	3	3
S35	2	2	3	3
S36	2	2	3	3

pairs. Table 4.17 compares each metric pair wise with all metrics. This was conduced with our metrics and experimental results on all 36 S-Boxes in our case study. We note that the metrics (SR/GE/δ) have high correlation with each-other and the other pairings have lower correlation.

Table 4.17: Pairwise Correlation of Experimental and S-Box Metrics

Element1	Element2	Correlation Value
GE	SR	-0.710
GE	δ	-0.946
SR	δ	0.767
GE	NL	0.235
GE	SNR	0.077
GE	DPA-SNR	0.227
GE	TO	0.308
GE	RTO	-0.308
GE	CC	0.321
SR	NL	-0.083
SR	SNR	0.077
SR	DPA-SNR	-0.336
SR	TO	-0.336
SR	RTO	0.157
SR	CC	-0.075
δ	NL	-0.161
δ	SNR	-0.152
δ	DPA-SNR	-0.163
δ	TO	-0.238
δ	RTO	0.257
δ	CC	-0.276

4.13 Results of Ternary GIFT Case Study

In this section we will interpret the results from this chapter on Generalized GIFT and our ternary case study. We will compare metric scores and the varying equivalence classes in comparison with the success/failure of our CPA attack and the mean success

rate of the successful attacks on varying S-Boxes sampled in our case study. We will then note on which S-Boxes in our ternary case study would be the 'optimal' choices compared with others that would not be a What is a practical decision? 1 of, involving, or concerned with experience or actual use; not theoretical. 2 of or concerned with ordinary affairs, work, etc. practical choice to use in real world situations.

4.13.1 Metric Comparison with Experimental Mean Success Rate

Recall in the binary case of S-Box metric analysis the non-linearity metric had a large correlation with the mean CPA success rate and other metrics has smaller correlation. However in our ternary case none of the metrics we sampled had notable correlation with the mean success rate.

In the case of non-linearity we note that non-linearity has three possible options: 0,1,1.267. Within each of these non-linearity metric scores we have S-Boxes that are successful and S-Boxes that were not able to recover the key fully given a large plaintext/trace pair size. The same statement on no noticeable correlation happened with all other sampled metric scores in comparison with CPA success in our ternary case study. While we did not sample the whole S-Box space because it was not practical to do, we do have numerous S-Boxes with similar metric scores showing vastly different CPA status and CPA mean success rate.

4.13.2 Equivalence Classes in comparison with Metrics and CPA Success/Fail

AE Class Analysis

Upon observation of the AE Table 4.11 we note that we can see the preservation of metrics or lack there of. We note that in the overlapping S-Boxes within a AE class that they always have the same NL metric score. As for the other metrics there can be some overlap but there does not have to be. For metrics other than NL we can find counterexamples of metric scores not holding such as S12 and S15 are both in AE Class 5 but have differing TO/RTO/CC values. Also in S-Boxes S6 and S10 within AE class 5 we note different DPA-SNR metric values.

Within each of the overlapping representatives we also note that there is a variation on the CPA status in which each class with more than one entry has at leas one success and one fail on the CPA attack to recover the private key. Upon this observation we state that there is no clear correlation that AE classes preserve the CPA status on key recovery.

4.13.3 What We Expected vs. Experimental Results

Initially we knew that the higher delta values would lead to a higher SR along with lower delta values leading to a lower SR based upon the definition of delta. We also knew that lower GE will lead to higher SR and higher GE will lead to lower SR based upon the GE values. This is because delta measures the difference in correlation of the correct key with the next best key. Having a high correlation means that on average the correct key guess has a higher correlation value by a margin then the next best key. Similarly GE measured the index of the correct sub-key. The more

sub-keys that have GE 0 means that the GE will be small leading to a higher success rate.

We did not know the rate of change when GE/Delta changes when compared to the success rate. Based upon our data, as GE changes delta also changes linearly. However bases upon Figure 4.1 the change in SR when compared to Delta/GE seems to be more of an exponential decay.

4.13.4 S-Box Metrics Alterations Attempted

Our initial goal when generalizing S-Box metrics was to find a metric generalization that could distinguish wither or not a S-Box is resistant to power analysis attacks. Our generalizations listed in this document do not do that however we tried. Most of our metrics had straight forward generalizations except the metric DPA-SNR in which we tired editing the metric many ways in hopes of getting better results to compare to the experimental data. We know from observations that changing the multiplicative constants will not lead to any substantial change of the metric scores so during our edits of DPA-SNR we focused on edits that would have a wider change being the exponents and where the ABS call is made. For the inside exponent we tired values (2,3,4, and 9) and for the outer constant we attempted values (-1/2,-1/3,-1/4,-1/9) as well as changing the ABS call before and after any possible summation call. After attempting all possible combinations of the changes listed we left editing/creating a S-Box that strongly correlates with power analysis resistance as future work.

4.14 Conclusion

Based upon our analysis of the experimental results we have classified the S-Boxes into categories based upon the mean Success Rate (SR). We note that the S-Boxes

in the success category are S-Boxes that leak the most in our case study and are the worst choice for practical use. S-Boxes in the Fail category however are more resistant and would be our choices for actual use.

Comparing the S-Boxes the two worst choices are S8 and S23 as they have 100% mean success rate given 400 traces. The best choice for S-Boxes based upon our experimental results are S6 and S10 in which both have 0% SR but also the Delta values are less than 0.01 and the Ge values are greater than 0.69 which indicates the least amount of leakage of S-Boxes surveyed in our case study.

CHAPTER 5:

BLACK BOX CRYPTANALYSIS

In this chapter we will be describing our black box case study on an unknown algorithm that is called AEA. We do not have access to the source code, but we have access to devices in which we can encrypt/decrypt packets of data and capture the packets using the wireshark packet sniffing software. In capturing data, our goal is to mathematically describe the underlying functions that are used in the encryption/decryption process without the source code and without utilizing reverse engineering. In this chapter, we will describe our initial data collection process and our methodologies that were used to understand the process of the underlying algorithm. We will then test our conjectures on what we believe that is going on in the underlying cryptosystem.

5.1 Introducing Black Box

5.1.1 Attack Models

In attacking cryptographic systems, there are a few knowledge models that one can use when performing cryptanalysis. These models describe the information that the attacker has access to. Some sample models are ciphertext only, known plaintext/ciphertext pairs, chosen plaintext and knowledge of the corresponding ciphertext, and chosen ciphertext and knowledge of the corresponding plaintext.

Attacking in the ciphertext only model, one has less information but if one is able to break a cryptosystem in this method the cryptosystem is proven to be weak. Breaking a cryptographic scheme in the black box model is much harder than the variants of grey/white but depending on information available and the skills of the attacking group it is still possible to do so.

5.2 Mathematical Definitions

In our attempt of this black box cryptanalysis, we will use some mathematical definitions that are described in this section. We will define these statements in this chapter but their use might not be clear until it is applied in our case study.

5.2.1 Modular Hamming Distance

In our case study, we will be comparing how close our predictions are with the actual 'true' values. In order to do this comparison we will be defining what is referred to as Modular Hamming Distance (MHD). Similar to the well-known Hamming distance (HD) model, MHD measures how close two values are to each other. But unlike the classical HD, our MHD will work in bases other than 2. While it will work in any base, for our research we will focus on a byte MHD which is base 256.

First we will show the comparison of two byte values using MHD. We consider two values $a, b \in \mathbb{Z}_{256}$. We define the byte-wise MHD to be

$$MHD_{256} = Min((a-b) \bmod 256, -(a-b) \bmod 256) \tag{5.1}$$

In this definition we see we are measuring how far away a is to b modulo 256, but we include both the positive and negative values of this distance using modular arithmetic. By doing this, we show that 0 is 1 away from 255 because $0 + (-1) = 255$

modulo 256.

In our research, we consider 8-byte values which can be viewed as elements in $\mathbb{Z}_{256}^8$. In order to compute byte-wise MHD on these values, we just add up the single byte MHD values and sum over the 8 byte pairs. A mathematical description of this is below:

$$\text{let } A, B \in \mathbb{Z}_{256}^8;\ MHD_{256}(A, B) = \sum_{i=0}^{7} \Big(MHD_{256}(A_i, B_i) \Big) \tag{5.2}$$

In computing MHD, we measure how close two 8-byte values are to each other when comparing values modulo 256.

5.2.2 Error Bounds

In our case study, we will use MHD to compare 8-byte chunks to each other. When computing the MHD value, we state if it is within a given error bound. Let us consider an error bound of 8. We state that when comparing two values A, B are within an error bound of 8 if

$$MHD_{256}(A, B) < 8 \tag{5.3}$$

If the MHD values of A and B are 8 or greater we state that the pair A,B are outside of the error bound.

In our case study, we will be making predictions on values and comparing them with 'true' values, but our predictions are not perfect. We need to measure how close the predictions are and if they are close enough to the true value to show that are prediction software is accurate. By measuring if our predictions are within the error bound, we will show while the predictions are not perfect, they are close to the true values.

5.3 Case Study SEL Information Gathering

In this section, we will be describing how we captured data in this case study. Initially we were given equipment that produced a cryptographic tunnel between two end-to-end nodes that were communicating to each other. In between the end nodes are 2 pieces of hardware, one takes in a plaintext and produces ciphertext and the other takes in ciphertext and produces plaintext. These pieces of hardware are paired with each other. The algorithm for encryption and decryption are unknown. A figure explaining the initial configuration is shown in Figure 5.1 While no source code of the compiled libraries were given to us, the code to call the library does take in the serial number of the other RTAC device in the RTAC configuration.

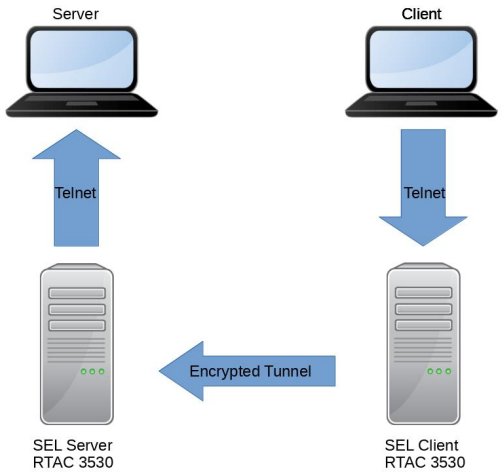

Figure 5.1: The original network setup for the SEL AEA encryption

In the default configuration the protocol to send data from the RTAC devices to the client/server was the unsecured telnet protocol. Initially we started sending data though the telnet protocol. The RTAC equipment would then encrypt/decrypt the

data and the end server node would receive what was initially sent. We captured the packets in the encrypted tunnel and noted that the encrypted packets were 4 bytes longer than the plaintext packets. We gathered the packets using the packed sniffer Wireshark (Lamping and Warnicke, 2004). Several plaintexts were sent for encryption but no pattern was noted in the initial setup depicted in Figure 5.1. Each time we encrypted a plaintext, a different ciphertext was produced. We noted that the encryption process must be probabilistic. We gathered several packets in which we encrypted 8 bytes in which each byte had the same value. This process was repeated for the digits 0-9 and single character letters a-z and A-Z. Because the encryption is probabilistic we then decided to perform chosen ciphertext attacks.
We edited the network topology to send a $n+4$ byte ciphertext to be decrypted and we recover a n byte plaintext.

Observation 5.1 *Encrypting any 'n' byte plaintext will result in the encrypted text being an 'n+4' byte ciphertext. This was tested for 'n' valued bytes lengths of 3,4, and 8.*

Observation 5.2 *AEA's encryption is probabilistic meaning that when encrypting the same plaintext the result is a different cipher text each time.*

5.3.1 Chosen Ciphertext Attack

In order to change our attack model to chosen ciphertext, we needed to modify our network topology. We eliminated the end node client as well as the client RTAC and added in a 'RTAC imposter'. The imposter will send ciphertext to the RTAC server and the RTAC server will decrypt the data for use and produce the plaintext. The

modified network topology for use in the chosen ciphertext data collection and attack scheme is shown in Figure 5.2

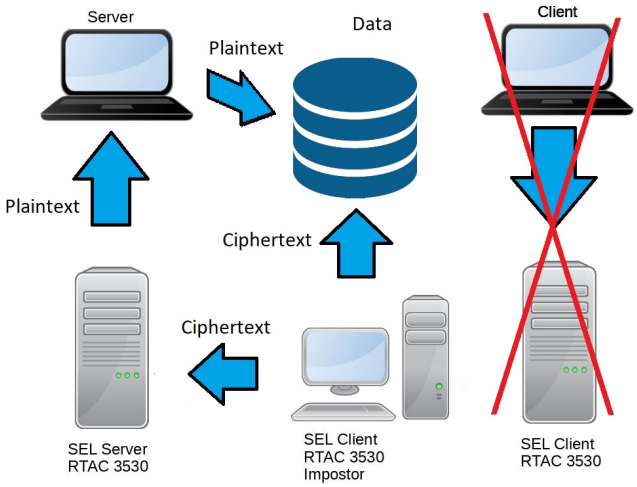

Figure 5.2: The modified network setup for chosen ciphertext attack on SEL AEA encryption algorithm

In the beginning of our data-collection, much was done by manually imputing data into a telnet terminal to get an idea of the underlying process that must be computed in the cryptographic algorithm. After some initial data collection we decided to automate the process and write a program in Java that would generate a random 12-byte ciphertext to be decrypted into a 8-byte plaintext. We took note that the ciphertext was 4 bytes larger than the plaintext. We started to investigate why the ciphertext was 4 byte larger and collected data in which the last 4 bytes would remain constant. Some of our noted observations on our initial data collection in the chosen ciphertext model are shown below:

Observation 5.3 *A ciphertext of all zeros when decrypted results in a plaintext of*

all zeros.

$$Dec(0x00...00) = 0x00...00 \tag{5.4}$$

We also noted some homomorphic proprieties in a 12-byte ciphertext affecting the first 8 bytes.

Observation 5.4 *In the case of a 12-byte ciphertext the first 8 bytes are malleable. Consider:*

$$Dec(C_1||C_2||...||C_8||B_1||...||B_4) = P_1||P_2||...||P_8 \tag{5.5}$$

we choose 'A' (random or pre-selected) such that:

$$A = A_1||A_2||...||A_8 \tag{5.6}$$

In which each A_i is one byte of data.

$$Dec((C_1 \oplus A_1)||(C_2 \oplus A_2)||...||(C_8 \oplus A_8)||B_1||...||B_4) = (P_1 \oplus A_1)||(P_2 \oplus A_2)||...||(P_8 \oplus A_8) \tag{5.7}$$

Each of the byte XOR operations or $\oplus$ are addition in the group:

$$\mathbb{Z}_2 \; x \; \mathbb{Z}_2 \; x \; \mathbb{Z}_2 \; x \; \mathbb{Z}_2 \; x \; \mathbb{Z}_2 \; x \; \mathbb{Z}_2 \; x \; \mathbb{Z}_2 \; x \; \mathbb{Z}_2 \tag{5.8}$$

In which each of the sub-groups represent a bit of data and all eight combined create a byte.

Note: The last four bytes: B_1, B_2, B_3, B_4 do not have this property.

Because of this homomorphic propriety, we noted that the last 4 bytes of the ciphertext must hold information relating to the decryption process. Also we noted

first 8 bytes hold information on the underlying plaintext. We then thought that the last 4 bytes could act as input to a pseudorandom function that generates an OTP like value to XOR with the other 8 bytes as listed in the proposition below:

Proposition 5.5 *In decryption the last four bytes create a One Time Pad (OTP) to decrypt the data. An example of a 12-byte ciphertext is shown below*

$$Dec(C_1||C_2||...C_8||B_1||...||B_4) = (C_1||C_2||...||C_8) \oplus (f_{OTP}(B_1, B_2, B_3, B_4)) \quad (5.9)$$

Next, we started to compute what values would be if the first 8 bytes of the ciphertext were zeros and the last 4 bytes differed.

In testing our idea on malleability, we gathered data in which the first 8 bytes of the ciphertext were zero while also having the hamming weight of the last 4 bytes was 1. This lead to 32 options on which bit was active in the last 4 bytes. An example is shown below:

Example 5.6 *Below is an example of a ciphertext in which one bit is active in the last four bytes*

$$F_1 \ = \ 0x00\ 00\ 00\ 00\ 00\ 00\ 00\ 00\ 80\ 00\ 00\ 00 \quad (5.10)$$

Which leads the last four bytes (in binary) to be:

$$0b10000000\ \ 0b00000000\ \ 0b00000000\ \ 0b00000000 \quad (5.11)$$

Example 5.7 *Consider when the 1 bit is shifted right in order to create all of the 32*

options.

$$F_i \ = \ (F_1 \ >> \ (i-1)) \tag{5.12}$$

And the decryption of F_i:

$$G_i = Dec(F_i) \tag{5.13}$$

After obtaining the decryption values of these 32 values, we asked ourselves if by knowing these values, we could approximate the function that we are assuming takes in the 4 byte quantity and produces the XOR pad. For each of the bits in the last 4 bytes of the cipher we have an 8 byte value. We then started to attempt an approximation on what would be the value if a ciphertext's first 8 bytes were zero and the last 4 bytes had a hamming weight of 2.

Example 5.8 *Consider if we have two values for the following 4 last bytes of the ciphertext:*

$$0x10\ 00\ 00\ 00 \ and \ \ 0x01\ 00\ 00\ 00 \tag{5.14}$$

Reusing the naming convention from Example 5.7, we can refer to these two values as F_4 and F_8. This implies the decrypted values are G_4 and G_8. Our speculation was that by knowing G_4 and G_8 we could produce the value for:

$$Dec(0x00\ 00\ 00\ 00\ 00\ 00\ 00\ 00\ 11\ 00\ 00\ 00) \tag{5.15}$$

Though trial and error, we ended up with two ways of approximating the XOR pad from values known in look up tables. The ways to combine the values are addition

with/without carry, and their definitions are listed below:

Definition 5.9 ***Addition without carry***

Considering an n-byte plaintext, addition without carry is the direct product of:

$$\mathbb{Z}_{2^8} \; x \; \mathbb{Z}_{2^8} \; x ... \; \mathbb{Z}_{2^8} \tag{5.16}$$

In which there are n $\mathbb{Z}_{2^8}$ groups.

Definition 5.10 ***Addition with carry***

Addition in the group $\mathbb{Z}_{2^{8*n}}$ in numbers with n byte plaintext.

In later sections in this chapter we will go into how these definitions are applied and their results in breaking this cipher in the black box setting.

5.4 Prediction Programs for AEA Decryption

In this section, we will describe how we took look-up tables and made an approximation on the decryption algorithm in the 12-byte ciphertext/ 8-byte plaintext setting. During our research into this approximation, we made different version of the decryption approximation that we note as Special Ciphertext Prediction, Plaintext Prediction, and XOR Pad Prediction. They ordered chronologically as for when they were created in our research. We are also able to compare the approximation with known plaintext values as we have access to chosen ciphertext/plaintext pairs. In doing so we can mathematically state how close our prediction are to the true XOR pad value.

5.4.1 Building Blocks for Prediction

We will start to describe how the addition without carry combines two predicted values into a single predicted value. One assumes that each of the two predicted values are obtained from the look-up tables are are being combined. An example of addition without carry is shown below:

Example 5.11 *Consider two plaintexts A,B such that:*

$$A = A_1||A_2||...||A_8 \tag{5.17}$$

$$B = B_1||B_2||...||B_8 \tag{5.18}$$

The prediction using addition without carry works as follows:

$$C' = A \odot B = (A_1 + B_1 \ mod\ 256)||(A_2 + B_2 \ mod\ 256)||...||(A_8 + B_8 \ mod\ 256) \tag{5.19}$$

Similarly our addition with carry procedures classically adds two 8 byte values. We ignore any carry over values in the addition process.

With access to the hardware, we are able to get the true XOR pad and plaintext values. We then needed a way to compare how close our approximated XOR pad value is to the true XOR pad value and a similar process for the plaintext values. In doing so we created an accuracy formula that is defined earlier in the error rate:

The accuracy computed the MHD of each of the 8 bytes and adds up the MHD values to get a total accuracy value. I the case of MHD = 0 the prediction is the same as the true value and the closer the value is to zero the closer the predicted pad

is to the true pad.

In this study, we started with the 32 look up tables, one for each bit in the 4-byte quantity of the last 4 bytes.

Look Up Tables used in Prediction Software

If approximating low hamming weight values, the accuracy was close to the true value, but as the hamming weight rose the prediction became less accurate. We then decided against using the 32 look up table values. Instead we created a list of 4 separate look up tables, one for each of the 4 bytes at the end of the ciphertext. Each table contains all 256 possible values for each of the extra 4 bytes resulting in 1024 values. We combine 4 of the values from the tables to compute our predicted XOR pad value. The look up tables that are used in the prediction software are generated as follows:

Consider the first of the 4 'extra bytes' to be B_1. B_1 is shown in the ciphertext as:

$$0x00\ 00\ 00\ 00\ 00\ 00\ 00\ 00\ B_1\ 00\ 00\ 00 \tag{5.20}$$

We loop though all 256 possible values for B_1 and store the decryption of those 256 values in a look up table. This process is repeated for the other 3 bytes B_2, B_3, B_4 and those also have a 256 value look up table for each of them.

We then use the look up tables to compute a prediction of a decryption value on a ciphertext in which the first 8 bytes are zero and the last 4 bytes are random such as what is displayed below:

$$0x00\ 00\ 00\ 00\ 00\ 00\ 00\ 00\ B_1\ B_2\ B_3\ B_4; \tag{5.21}$$

To do this, we have the 4 look up tables for the B values and we use addition either with or without carry which is denoted with the $\odot$ symbol. This creates a prediction by performing the formula below:

$$Table_1(B_1) \odot Table_2(B_2) \odot Table_3(B_3) \odot Table_4(B_4) \tag{5.22}$$

Next, we compare the prediction with the true value to get a numerical representation of how close the prediction is to the true value (MHD error value).

5.5 Prediction Program Versions

Using the building block presented in the previous section, we will now describe how they are applied to approximate decryption. In this case study we created three versions of the prediction software which are described in the sub-sections below.

5.5.1 Prediction Software - Special Ciphertext Prediction

The first version of our prediction software (Special Ciphertext Prediction) applies the calculations of the look up table and produces a prediction of the decryption in the special case of a 12-byte ciphertext when first 8 bytes are zero. We then apply the accuracy formula to compare our predicted decryption value with the actual value that has been obtained.

Given a ciphertext with eights bytes of 0 followed by four random bytes we compute the accuracy of:

$$Err(Dec(0||...||0||B_1||B_2||B_3||B_4), Pred(B_1, B_2, B_3, B_4)) \tag{5.23}$$

In which the *Pred* function applies the look up tables and combines the output of

the look up tables with addition without carry resulting in:

$$Pred(B_1, B_2, B_3, B_4) = Table_1(B_1) \odot Table_2(B_2) \odot Table_3(B_3) \odot Table_4(B_4) \quad (5.24)$$

During computations with the hardware, errors occur in data collecting in which some packets are dropped when running our computations. This results in a differing number of predictions compared to the number of plaintexts captured. In order to deal with this disparity, we look for when errors occur and re-synchronized the tables to pair together as shown in Figure 5.3

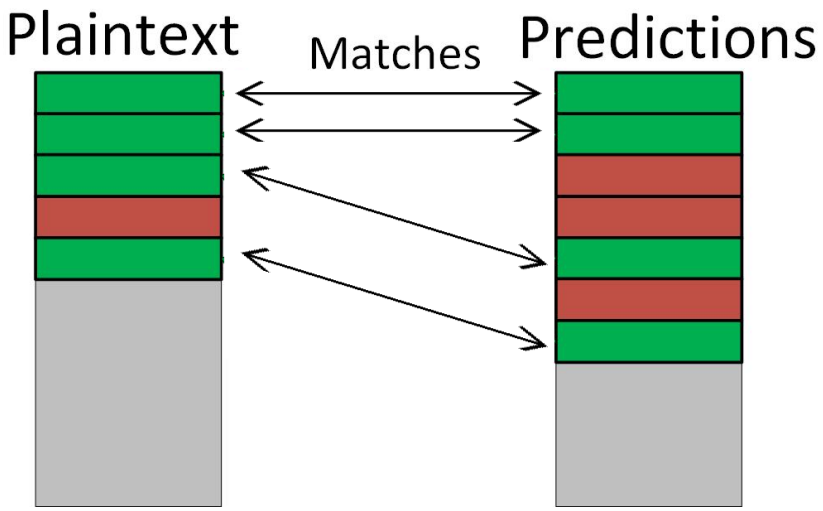

Figure 5.3: Special Ciphertext Prediction Software Accounting for Error in Packet Capture

We then apply our accuracy formula for each paired prediction to true value. We compute the comparison and get an error value interpreting the modular hamming distance (MHD). We collect different data-sets and then represent the results of that data-set in the heat map table. The results of our Special Ciphertext Prediction data-sets are shown below:

Experiment 1 *A data sample results using Special Ciphertext Prediction with 992,570 samples are shown below. It was our first large-scale data collection.*

Table 5.1: Special Ciphertext Prediction Sample 1

v1 data set 1: Prediction size 1 million, Plaintext size 992,570									
MHD	0	1	2	3	4	5	6	7	Total
Value	38,047	208,887	382,594	273,683	81,654	7,705	0	0	992,570

Special Ciphertext Prediction with data set 1 resulted in 992,570 matches (successful predictions within an error bound)

$$\frac{992,570}{992,570} = 100\% \tag{5.25}$$

Of data within the error bound.

Experiment 2 *111,833 matches were in the allowed error bound leading to a success rate of:*

Table 5.2: Special Ciphertext Prediction Sample 2

Special Ciphertext Prediction data set Apr15: Prediction size 112,007, Plaintext size 111,833									
MHD	0	1	2	3	4	5	6	7	Total
Value	4,256	23,459	43,177	31,011	9,065	865	0	0	111,833

$$\frac{111,833}{111,833} = 100\% \tag{5.26}$$

Of data within the error bound.

We note that in the Special Ciphertext Prediction of our case study, we are able to match 100% of the value together after accounting for the packet loss error. While only some have 100% accuracy, all of our predictions are within a MHD value of 5 across the 8-byte predicted plaintext value.

While our Special Ciphertext Prediction prediction software only works in the special case that the first 8 bytes are zero, The other versions work in cases where all 12-bytes of the ciphertext are random.

5.5.2 Prediction Software - Plaintext Prediction

Our Plaintext Prediction software was our first attempt at predicting the decrypted value of a random 12-byte ciphertext. This version is a generalization of t as it the Special Ciphertext Prediction as not restricted to the 'special' ciphertext. Because Special Ciphertext Prediction had a ciphertext of 8 leading zeros, the decryption value is the same as the predicted XOR pad if Proposition 5.5 holds. In Plaintext Prediction, we similarly use the last 4 bytes to compute the predicted XOR pad and we then use that prediction to attempt to decrypt the ciphertext when the first 8 bytes are random.

Because we know from Special Ciphertext Prediction, the prediction of the XOR pad is not always accurate, we will have to account for this error in the Plaintext Prediction. We do this by acquiring three 8-byte quantities when computing the prediction. Those three values are the true known plaintext, the prediction using addition without carry, and the prediction without carry with '1' are added to each of the bytes. We consider in this case that P' is the true decrypted value, while P is the prediction, and P^E is the prediction with '1' added to each of the elements. In order to compute the accuracy with these values we edit the accuracy formula as

shown below:

Definition 5.12 *Let P' be the true known plaintext, P be our predicted plaintext, and P^E be the plaintext prediction with 1 added to each byte value. Our error computation Err_{V2} is shown below*

$$Err_{V2}(P', P, P^E) = \sum_{i=1}^{n} (Min(|P_i - P'_i|, |P_i^E - P'_i|)) \tag{5.27}$$

And the P values are computed as follows given $\odot$ is modular addition:

$$Pad = Table_1(B_1) \odot Table_2(B_2) \odot Table_3(B_3) \odot Table_4(B_4) \tag{5.28}$$

$$P \ = \ Pad \oplus Cipher \tag{5.29}$$

Assuming P_i is the i^{th} byte of P and $||$ denotes the concatenation operation P^E will be:

$$P^E = (P_1+)||(P_2 + 1)||...(P_n + 1) \tag{5.30}$$

In the equations above, we note that Plaintext Prediction attempts to deal with the error by adding 1 to the predicted plaintext value like what was done in Special Ciphertext Prediction. This leads to errors that will be described in a later sub-section and is corrected in the last version.

In Plaintext Prediction the process of accounting for packet error is the same in Special Ciphertext Prediction Figure 5.3, except we have two sets of predictions, the classical and the error predictions.

Plaintext Prediction did not perform as well as the first version when measuring the accuracy. In our experimental results, we gathered a single data-set for Plaintext Prediction because after observing the error in Plaintext Prediction we created a new version of the software that results in better accuracy.

Experiment 3

Table 5.3: Plaintext Prediction Sample 1

Plaintext Prediction data set Feb1: Prediction size 83,766, Plaintext size 83,666									
MHD	0	1	2	3	4	5	6	7	Total
Value	3,081	17,382	30,010	17,666	3,276	0	0	0	71,415

71,415 matches were in bound leading to a success rate of:

$$\frac{71,415}{83,666} = 85.4\% \tag{5.31}$$

Of data within the error bound.

5.5.3 Prediction Software - XOR Pad Prediction

Our XOR Pad Prediction software is our latest version. It works similar to Plaintext Prediction, in which it attempts to make predictions on the decryption value of a random 12-byte ciphertext. However it performs better than Plaintext Prediction and the computations on the error values differ when compared to Plaintext Prediction. The majority of our results on research for the black-box cryptanalysis comes from XOR Pad Prediction. It has high accuracy rate and works on 12-bytes of random ciphertext as compared to Version 1 that only works on special ciphertexts.

When comparing results on XOR Pad Prediction, instead of comparing known plaintext values, we compare known XOR pad values with the predicted XOR pad values. In the first version, these values are the same but because the first 8 bytes are random. This differs when compared to the other two versions. When comparing accuracy, we use the same error formula from Special Ciphertext Prediction, but the inputs to the formula differ in XOR Pad Prediction. We define D to be the prediction of the XOR pad coming from addition without carry of the 4 look-up table values. E is defined to be the true XOR pad by performing the XOR operation on the first 8 bytes of the Ciphertext with the 8-byte plaintext resulting in the 'true' XOR pad.

Definition 5.13

$$D = Pred(B_1, B_2, B_3, B_4) \tag{5.32}$$

$$E = (P_1||P_2||...||P_n) \oplus (C_1||C_2||...||C_n) \tag{5.33}$$

XOR Pad Prediction works better than Plaintext Prediction because instead of performing the error computations on the predicted plaintext value the error computations are computing on the predicted XOR pad leading to better computational results. XOR Pad Prediction deals with packet error/loss the same way that the other versions dealt with the packet error/loss.

Because XOR Pad Prediction performed better than Plaintext Prediction, we spent more time gathering different data-sets and also expanded the experiment to see in which byte the errors occur in the generation of the XOR pad. The results in our experimentation are shown below:

Experiment 4

Table 5.4: XOR Pad Prediction using Sample 1

XOR Pad Prediction data set Feb1: Prediction size 83,766, Plaintext size 83,665									
MHD	0	1	2	3	4	5	6	7	Total
Value	3,082	17,473	32,413	23,183	6,897	617	0	0	83,665

83,665 matches were in bound leading to a success rate of:

$$\frac{83,665}{83,665} = 100\% \tag{5.34}$$

Of data within the error bound.

In XOR Pad Prediction we compared the true OTP value with the predicted one and came up with these detailed results showing in which byte the error occurs on

Table 5.5: XOR Pad Prediction Error Sample 1

XOR Pad Prediction data set Feb1: 83,665 Matches, Error Table				
Byte Number	Err=0	Err=1	Err=2	Err=3
0	83,665	0	0	0
1	83,665	0	0	0
2	42,285	41,380	0	0
3	59,938	23,727	0	0
4	83,665	0	0	0
5	16,116	55,663	11,886	0
6	45,686	37,979	0	0
7	83,665	0	0	0

Table 5.6: XOR Pad Prediction Sample 2

XOR Pad Prediction data set Jan25: Prediction size 88,784, Plaintext size 88,573									
MHD	0	1	2	3	4	5	6	7	Total
Value	3,437	18,779	34,239	24,283	7,162	673	0	0	88,573

Experiment 5

88,573 matches were in bound leading to a success rate of:

$$\frac{88,573}{88,573} = 100\% \tag{5.35}$$

Of data within the error bound.

The error distribution is shown below:

Table 5.7: XOR Pad Prediction Error Sample 2

XOR Pad Prediction data set Jan25 88,573 Matches, Error Table				
Byte Number	Err=0	Err=1	Err=2	Err=3
0	88,573	0	0	0
1	88,573	0	0	0
2	44,807	43,766	0	0
3	63,877	24,696	0	0
4	88,573	0	0	0
5	17,507	58,515	12,551	0
6	48,533	40,040	0	0
7	88,573	0	0	0

In our experiments in XOR Pad Prediction we note that we are able to recover the XOR pad with 100% success rate when accounting for error. When error does occur we note that we consider which byte the error occurs on. Based upon the data we gathered in XOR Pad Prediction, we made some observations on the experimental data we gathered.

Observation 5.14 *In the case of a 12-byte ciphertext and using 4 look up tables, 1 for each of the last 4 bytes, bytes 0,1,4, and 7 have no error during the prediction of the XOR pad.*

Observation 5.15 *In the case of a 12-byte ciphertext and using 4 look up tables, 1 for each of the last 4 bytes, bytes 2,3 and 5 have no error or have a error of 1 during the prediction of the XOR pad.*

Observation 5.16 *In the case of a 12-byte ciphertext and using 4 look up tables, 1 for each of the last 4 bytes, bytes 5 has the most possible error during the XOR pad prediction in which it is either correct or can have an error up to 2.*

Observation 5.17 *In the case of a 12-byte ciphertext and using 4 look up tables, 1 for each of the last 4 bytes, bytes 2 and 6 have close to 50% of the time correctly predicting the byte and the other 50% has an error of 1.*

Based upon these observations, if the attacker does not have access to the plaintext the attacker can create the possible XOR pad values accounting for error leading to $2^3 * 3 = 24$ possible pad values. The search that space to find the true plaintext of the possible 24 XOR pad values. Because of the distribution of the error values one can order the 24 possible values statistically by the likelihood of the outcome of the values based upon the error percentage in our experimental data.

CHAPTER 6:

CONCLUSION

In this work, we looked into cryptanalysis of binary ciphers, generalized/ternary ciphers, and an unknown cryptographic implementation. Chapter 3 focuses on CPA attacks on of the binary GIFT64 cryptographic in which we swapped out the S-Box layer of the cipher and evaluated the power analysis susceptibility/resistance of the S-Box parameter in the power analysis setting. Chapter 4 showed a mathematical generalization of a GIFT like scheme and out implementation TRITGIFT32 in the ternary setting. We also studied the power analysis resistance/susceptibility of the S-Box layer in the ternary setting. Chapter 5 showed how to approach the black-box cryptographic of an unknown cryptographic implementation and results we found in this case study. In this final chapter we review and answer our research questions as well as restate some open questions that could be used that our research did not answer.

6.1 Research Results and Contribution

- **How well do the current metrics quantify power analysis leakage?** In our binary case study we sampled several S-Boxes in our implementation with varying S-Box metric values. We showed that Non-Linearity had the strongest correlation with any of the metrics that we surveyed. However, other metrics

such as Transparency Order (and it's variations) and DPA-SNR did have a relationship with the CPA mean success rate. Non-Linearity had clear correlation with CPA attack but as there are only 3 possible Non-Linearity values there was a lot of variations with the Non-Linearity classes. Because the implementation should be resistant to other cryptanalysis attacks besides power anaysis the Non-Linearly should be as high as possible which and choose a S-Box within the high Non-Linearity class that is more resistant in the class.

- **Given some binary cryptographic S-Boxes from literature, which are more susceptible/resistant**

  In literature S-Boxes are chosen to have the highest Non-Linearity to defend against classical cryptanalysis attacks. Of the S-Boxes we sampled PICCOLO was the most susceptibility to attack and PRESENT was most resistant. It is worth noting that we did find S-Boxes not used in literature that were more resistant in the power analysis setting but they should not be used because of the weaknesses to classical cryptanalysis techniques.

- **Can we generalize cryptographic ciphers and metrics to work on other prime power and not just the binary case?** In our work we did generalize the base cryptographic scheme and applied the generalization to work in the ternary setting. We mathematically generalized the cryptographic metrics to work in the generalized setting and also applied them to work in the ternary setting. While the S-Box metrics did not work well in the ternary setting the general metrics that are obtained by the experimental CPA results worked well to compare S-Boxes in the ternary case study.

- **How well correlated are the generalized cryptographic metrics to power analysis resistance/susceptibility** In the ternary case study, none of the ternary S-Box metrics had strong correlation with the CPA success rate. However, our experimental result metrics ($SR/GE/\delta$) were used to compare power analysis susceptibility/resistance well because they are the result of the CPA attack being executed.

- **Do mathematical equivalence classes preserve metrics in the generalized setting** Yes, the S-Box Non-Linearity metric is preserved in the equivalences of affine, permutation modular addition, and modular addition. Transparency Order and Revisited Transparency Order are preserved in permutation modular addition and modular addition transforms. While we have proofs to show this in the generalized setting we also executed proofs by computation in the ternary setting as well as showing contradictions when the transforms do not preserve the metrics.

- **Given an implementation of a prime powered non binary base implemented in hardware, will power analysis attacks work and if so, how well do the attacks work?** In our ternary case study we did have an implementation that lead to success in the CPA setting. We sampled several S-Box parameters in the ternary implementation and had results ranging from 100% mean success rate to 0% mean success rate given the same amount of input data to the differing S-Boxes. Because of this we also used the experimental result metrics of GE and δ to compare the different S-Boxes. Given the experimental result metrics we show that the choice of the S-Box parameter in the ternary setting has a relationship with the CPA resistance/susceptibility.

- **Do different cryptographic implementations of odd prime powers leak less/more?** In our first ternary implementation the CPA attacks were not successful given any cryptographic S-Boxes that were tests. In our second ternary implementation we caused all three trit options to have unique Hamming weights in out implementation which caused us to have high success given some S-Boxes. Because of this the implementation definitely has a relationship with the CPA resistance/susceptibility.

- **Given a cryptographic implementation of an unknown algorithm how hard is it to model a black box attack to attempt to break the encryption of the unknown algorithm?** While every black box cryptanalysis case will differ, we showed our approach of black box cryptanalysis on our unknown implementation. Through chosen ciphertext attack we were able to gather sufficient data to model the scheme and predict what the plaintext will be given the ciphertext.

CHAPTER 7:

FUTURE WORK

7.1 Correlation Power Analysis

For future work, one could look into the following

One can expand on this work and consider a larger number of S-Boxes for comparing with the experimental result of SR and the S-Box metrics. We only considered a few S-Boxes and one could select a larger set of S-Boxes to analyze. While it is not practical to sample the whole S-Box space (16!), one could sample a larger set. Also we only selected a few S-Box metrics. One could also sample more metrics and compare them to experimental results.

While this work focused on 4-bit S-Boxes one could look at other S-Boxes sizes. In literature there exist 8-bit S-Boxes for use in ciphers like AES and other sized including 5-bit S-Boxes.

This research only used micro-controllers but one could extend this research into other hardware such as FPGAs and ASIC devices. This is important because both micro-controllers and FPGA/ASIC are used in real world use cases and we did not investigate S-Box security for FPGA/ASIC architectures.

One can also extend this research to work on protected version of ciphers using masking or threshold implementations. Also available for research is to use other

methodologies such as mutual information and profiled attacks as well as higher ordered attacks.

7.2 Generalized Correlation Power Analysis

In this work we generalized a few metrics that were noted to have a connection with base 2 power analysis such as TO/RTO and non-linearity. One can work on generalizing other metrics such as differential uniforming and other metrics with a connection with classical cryptanalysis.

The focus of this chapter was on cryptographic side channels but one could extend classical cryptanalysis techniques such as linear, differental, and algebraic cryptanalysis on non-binary SPN structured cryptosystems. In doing so one can find/prove that the generalization of classical crypto metrics could have a relationship on the generalization of generalized classical cryptanalysis techniques.

We also only considered changing the underlying structure from a base 2 powered field to a prime powered field, one could extend this and produce a SPN structure on different mathematically structures such as elliptic curves, Dihedral groups, Lucas's groups or other mathematically sets and measure their security.

7.3 Ternary Correlation Power Analysis

The goal in this ternary research was to find a metric that indicated resistance/susceptibility to power analysis without the need to compute actual mean SE/GE rates. In the binary case there was clear correlation between some of the metrics and experimental results but in this case study in the ternary case that was not true. In future work one could design/modify a metric and show that has a relationship with the mean SR/GE results. We also only sampled some of the PE classes and one could extend

this work so sample more of them and derive results on the relationship between some PE classes and experimental SR/GE results.

Also in this work we set the permutation layer constant and only changed the S-Boxes during the experiments. One could fix a S-Box and do computations on how other parameters effect the security in relationship with power analysis. In doing so one can also perform computations on other bases besides 2 and 3.

One can also look into changing the trit implementation in base 2 hardware. We chose this implementation based upon the experimental results that lead to an early success with the initial S-Boxes but one could change the implementation and measure the resistance to power analysis attacks. We computed all of the ternary experiments on a 8-bit XMEGA micro-controller but one could try different micro-controllers or FPGAs and get different results in doing so.

In this research we only consider attacking the output of the S-Box using the hamming weight model. One can use the hamming distance model or attack another part of the cipher and produce results doing so.

We only computed the results on unprotected GIFT but we could apply power analysis mitigation techniques shown in the works of (Bilgin et al., 2014a,b; Sasdrich et al., 2018) and show that they also work on protecting ternary based ciphers.

7.4 Black Box Cryptanalysis

In this work we never had access to the source code so if we were granted access a lot more research could be done. Also because we do not have access to the code we cannot 'prove' any statements but based upon the data and observation we have a decent idea of what is going on 'under the hood' of the underlying cryptographic algorithm.

For this case study we only really considered the 12-byte ciphertext leading to an 8-byte plaintext. Some initial steps have been performed on a generalized size but that work remains unfinished.

During the data collection, some plaintext packets are dropped and the reason for this is unknown. One could research what causes the dropped packets.

The last version of the prediction software are able to predict the XOR pad but we have to account for some error as the predictions are not perfect. One can dig deeper to predict when the error of off by one or two will occur and expand on this work.

In this case study we only used one RTAC Server but one could use another RTAC device and collect the look up tables for that device to show that this methodology is correct when given different RTAC devices using the same cryptographic libraries but having a differing RTAC serial number.

Lastly one could hook the device to a more real world examples such as a power grid or micro-grid compared to this program just inputting random ciphertexts and see how this prediction program performs in a more real world situation.

REFERENCES

Al Faruque, M., Regazzoni, F., and Pajic, M. (2015). Design methodologies for securing cyber-physical systems. In *2015 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ ISSS)*, pages 30–36. IEEE.

Al Faruque, M. A., Chhetri, S. R., Canedo, A., and Wan, J. (2016a). Acoustic side-channel attacks on additive manufacturing systems. In *2016 ACM/IEEE 7th international conference on Cyber-Physical Systems (ICCPS)*, pages 1–10. IEEE.

Al Faruque, M. A., Chhetri, S. R., Canedo, A., and Wan, J. (2016b). Forensics of thermal side-channel in additive manufacturing systems. *University of California, Irvine*, 12:13.

Alallayah, K. M., Alhamami, A. H., AbdElwahed, W., and Amin, M. (2012). Applying neural networks for simplified data encryption standard (sdes) cipher system cryptanalysis. *Int. Arab J. Inf. Technol.*, 9(2):163–169.

Alioto, M., Poli, M., and Rocchi, S. (2008). Power analysis attacks to cryptographic circuits: a comparative analysis of dpa and cpa. In *2008 international conference on microelectronics*, pages 333–336. IEEE.

Anderson, R., Biham, E., and Knudsen, L. (1998). Serpent: A proposal for the advanced encryption standard. *NIST AES Proposal*, 174:1–23.

Atkin, A. O. L. and Morain, F. (1993). Finding suitable curves for the elliptic curve method of factorization. *Mathematics of Computation*, 60(201):399–405.

Bakoev, V. (2017). Fast bitwise implementation of the algebraic normal form transform. *Serdica Journal of Computing*, 11(1):045p–057p.

Banik, S., Bogdanov, A., Peyrin, T., Sasaki, Y., Sim, S. M., Tischhauser, E., and Todo, Y. (2019). Sundae-gift. *Submission to Round*, 1.

Banik, S., Chakraborti, A., Iwata, T., Minematsu, K., Nandi, M., Peyrin, T., Sasaki, Y., Sim, S. M., and Todo, Y. (2020). Gift-cofb. *IACR Cryptol. ePrint Arch.*, 2020:738.

Banik, S., Pandey, S. K., Peyrin, T., Sasaki, Y., Sim, S. M., and Todo, Y. (2017). Gift: a small present. In *International Conference on Cryptographic Hardware and Embedded Systems*, pages 321–345. Springer.

Bao, Z., Chakraborti, A., Datta, N., Guo, J., Nandi, M., Peyrin, T., and Yasuda, K. (2019). Photon-beetle authenticated encryption and hash family. *NIST lightweight competition round*, 1:115.

Bayrak, A. G., Regazzoni, F., Brisk, P., Standaert, F.-X., and Ienne, P. (2011). A first step towards automatic application of power analysis countermeasures. In *2011 48th ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 230–235. IEEE.

Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., and Wingers, L. (2015). The simon and speck lightweight block ciphers. In *Proceedings of the 52nd Annual Design Automation Conference*, pages 1–6.

Beierle, C., Biryukov, A., dos Santos, L. C., Großschädl, J., Perrin, L., Udovenko, A., Velichkov, V., and Wang, Q. (2020). Lightweight aead and hashing using the sparkle permutation family. *IACR Transactions on Symmetric Cryptology*, pages 208–261.

Biham, E. and Shamir, A. (1991). Differential cryptanalysis of des-like cryptosystems. *Journal of CRYPTOLOGY*, 4(1):3–72.

Bilgin, B., Gierlichs, B., Nikova, S., Nikov, V., and Rijmen, V. (2014a). Higher-order threshold implementations. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 326–343. Springer.

Bilgin, B., Gierlichs, B., Nikova, S., Nikov, V., and Rijmen, V. (2014b). A more efficient aes threshold implementation. In *International Conference on Cryptology in Africa*, pages 267–284. Springer.

Billet, O., Gilbert, H., and Ech-Chatbi, C. (2004). Cryptanalysis of a white box aes implementation. In *International workshop on selected areas in cryptography*, pages 227–240. Springer.

Biryukov, A., Bouillaguet, C., and Khovratovich, D. (2014). Cryptographic schemes based on the asasa structure: Black-box, white-box, and public-key. In *International conference on the theory and application of cryptology and information security*, pages 63–84. Springer.

Biryukov, A., De Canniere, C., Braeken, A., and Preneel, B. (2003). A toolbox for cryptanalysis: Linear and affine equivalence algorithms. In *International conference on the theory and applications of cryptographic techniques*, pages 33–50. Springer.

Biryukov, A., Dinu, D., and Großschädl, J. (2016). Correlation power analysis of lightweight block ciphers: From theory to practice. In *International Conference on Applied Cryptography and Network Security*, pages 537–557. Springer.

Bogdanov, A., Knudsen, L. R., Leander, G., Paar, C., Poschmann, A., Robshaw, M. J., Seurin, Y., and Vikkelsoe, C. (2007). Present: An ultra-lightweight block cipher. In *International workshop on cryptographic hardware and embedded systems*, pages 450–466. Springer.

Borghoff, J., Canteaut, A., Güneysu, T., Kavun, E. B., Knezevic, M., Knudsen, L. R., Leander, G., Nikov, V., Paar, C., Rechberger, C., et al. (2012). Prince–a low-latency block cipher for pervasive computing applications. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 208–225. Springer.

Bos, J. W., Halderman, J. A., Heninger, N., Moore, J., Naehrig, M., and Wustrow, E. (2014). Elliptic curve cryptography in practice. In *International Conference on Financial Cryptography and Data Security*, pages 157–175. Springer.

Bovy, E., Daemen, J., and Mennink, B. (2020). Comparison of the second round candidates of the nist lightweight cryptography competition. *Bachelor Thesis, Radboud University*.

Brassard, G., Høyer, P., and Tapp, A. (1998). Quantum cryptanalysis of hash and

claw-free functions. In *Latin American Symposium on Theoretical Informatics*, pages 163–169. Springer.

Brier, E., Clavier, C., and Olivier, F. (2004). Correlation power analysis with a leakage model. In *International workshop on cryptographic hardware and embedded systems*, pages 16–29. Springer.

Canteaut, A. and Roué, J. (2015). On the behaviors of affine equivalent sboxes regarding differential and linear attacks. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 45–74. Springer.

Capillon, P. (2016). Black-box cryptanalysis of home-made encryption algorithms: a practical case study. In *Symposium sur la sécurité des technologies de l'information et des communications (SSTIC)*.

Chakraborty, K., Sarkar, S., Maitra, S., Mazumdar, B., Mukhopadhyay, D., and Prouff, E. (2017). Redefining the transparency order. *Designs, codes and cryptography*, 82(1-2):95–115.

Chari, S., Jutla, C. S., Rao, J. R., and Rohatgi, P. (1999). Towards sound approaches to counteract power-analysis attacks. In *Annual International Cryptology Conference*, pages 398–412. Springer.

Chen, C., Inci, M. S., Taha, M., and Eisenbarth, T. (2016). Spectre: A tiny side-channel resistant speck core for fpgas. In *International Conference on Smart Card Research and Advanced Applications*, pages 73–88. Springer.

Cheng, L., Zhang, W., and Xiang, Z. (2015). A new cryptographic analysis of 4-bit s-

boxes. In *International Conference on Information Security and Cryptology*, pages 144–164. Springer.

Chow, S., Eisen, P., Johnson, H., and Van Oorschot, P. C. (2002a). White-box cryptography and an aes implementation. In *International Workshop on Selected Areas in Cryptography*, pages 250–270. Springer.

Chow, S., Eisen, P., Johnson, H., and Van Oorschot, P. C. (2002b). A white-box des implementation for drm applications. In *ACM Workshop on Digital Rights Management*, pages 1–15. Springer.

Courtois, N. T. and Bard, G. V. (2007). Algebraic cryptanalysis of the data encryption standard. In *IMA International Conference on Cryptography and Coding*, pages 152–169. Springer.

Daemen, J., Hoffert, S., Peeters, M., Assche, G. V., and Keer, R. V. (2020). Xoodyak, a lightweight cryptographic scheme.

Daemen, J. and Rijmen, V. (1998). The block cipher rijndael. In *International Conference on Smart Card Research and Advanced Applications*, pages 277–284. Springer.

Daemen, J. and Rijmen, V. (2001). Reijndael: The advanced encryption standard. *Dr. Dobb's Journal: Software Tools for the Professional Programmer*, 26(3):137–139.

De Cnudde, T., Bilgin, B., Reparaz, O., Nikov, V., and Nikova, S. (2015). Higher-order threshold implementation of the aes s-box. In *International conference on smart card research and advanced applications*, pages 259–272. Springer.

Diffie, W. and Hellman, M. (1976). New directions in cryptography. *IEEE transactions on Information Theory*, 22(6):644–654.

Dobraunig, C., Eichlseder, M., Mangard, S., Mendel, F., and Unterluggauer, T. (2017). Isap–towards side-channel secure authenticated encryption. *IACR Transactions on Symmetric Cryptology*, pages 80–105.

Dobraunig, C., Eichlseder, M., Mendel, F., and Schläffer, M. (2016). Ascon v1. 2. *Submission to the CAESAR Competition*.

Dürmuth, M., Oswald, D., and Pastewka, N. (2016). Side-channel attacks on fingerprint matching algorithms. In *Proceedings of the 6th International Workshop on Trustworthy Embedded Devices*, pages 3–13.

Durvaux, F. and Standaert, F.-X. (2016). From improved leakage detection to the detection of points of interests in leakage traces. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 240–262. Springer.

ElGamal, T. (1985). A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE transactions on information theory*, 31(4):469–472.

Fei, Y., Luo, Q., and Ding, A. A. (2012). A statistical model for dpa with novel algorithmic confusion analysis. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 233–250. Springer.

Feistel, H. (1974). Block cipher cryptographic system. US Patent 3,798,359.

Freedman, D., Pisani, R., and Purves, R. (2007). Statistics (international student edition). *Pisani, R. Purves, 4th edn. WW Norton & Company, New York*.

Galbally, J. (2020). A new foe in biometrics: A narrative review of side-channel attacks. *Computers & Security*, 96:101902.

Gkaniatsou, A., Arapinis, M., and Kiayias, A. (2017). Low-level attacks in bitcoin wallets. In *International Conference on Information Security*, pages 233–253. Springer.

Golić, J. D. and Tymen, C. (2002). Multiplicative masking and power analysis of aes. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 198–212. Springer.

Guilley, S., Hoogvorst, P., and Pacalet, R. (2004). Differential power analysis model and some results. In *Smart Card Research and Advanced Applications Vi*, pages 127–142. Springer.

Gupta, N., Jati, A., Sanadhya, S., Chattopadhyay, A., and Chang, D. (2021). Threshold implementations of gift: A trade-off analysis.

Hankerson, D., Hernandez, J. L., and Menezes, A. (2000). Software implementation of elliptic curve cryptography over binary fields. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 1–24. Springer.

Harkanson, R. and Kim, Y. (2017). Applications of elliptic curve cryptography: A light introduction to elliptic curves and a survey of their applications. In *Proceedings of the 12th annual conference on cyber and information security research*, pages 1–7.

Hell, M., Johansson, T., Maximov, A., Willi Meier, F., Sönnerup, S. J., and Yoshida, H. (2019). Grain-128aeadv2-a lightweight aead stream cipher.

Iwata, T., Khairallah, M., Minematsu, K., and Peyrin, T. (2020). Duel of the titans: The romulus and remus families of lightweight aead algorithms. *IACR Transactions on Symmetric Cryptology*, pages 43–120.

Jati, A., Gupta, N., Chattopadhyay, A., Sanadhya, S. K., and Chang, D. (2019). Threshold implementations of GIFT: A trade-off analysis. *IEEE Transactions on Information Forensics and Security*, 15:2110–2120.

Käsper, E. (2011). Fast elliptic curve cryptography in openssl. In *International Conference on Financial Cryptography and Data Security*, pages 27–39. Springer.

Kim, S., Hong, D., Sung, J., and Hong, S. (2020). Classification of 4-bit s-boxes for bogi permutation. *IEEE Access*, 8:210935–210949.

Knudsen, L. and Wagner, D. (2002). Integral cryptanalysis. In *International Workshop on Fast Software Encryption*, pages 112–127. Springer.

Koblitz, N. (1987). Elliptic curve cryptosystems. *Mathematics of computation*, 48(177):203–209.

Kocabas, O., Soyata, T., and Aktas, M. K. (2016). Emerging security mechanisms for medical cyber physical systems. *IEEE/ACM transactions on computational biology and bioinformatics*, 13(3):401–416.

Kocher, P., Jaffe, J., and Jun, B. (1999). Differential power analysis. In *Annual international cryptology conference*, pages 388–397. Springer.

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

Lamping, U. and Warnicke, E. (2004). Wireshark user's guide. *Interface*, 4(6):1.

Lauter, K. (2004). The advantages of elliptic curve cryptography for wireless security. *IEEE Wireless communications*, 11(1):62–67.

Li, H., Zhou, Y., Ming, J., Yang, G., and Jin, C. (2020). The notion of transparency order, revisited. *The Computer Journal*, 63(12):1915–1938.

Lo, O., Buchanan, W. J., and Carson, D. (2017). Power analysis attacks on the aes-128 s-box using differential power analysis (dpa) and correlation power analysis (cpa). *Journal of Cyber Security Technology*, 1(2):88–107.

López, J. and Dahab, R. (1998). Improved algorithms for elliptic curve arithmetic in gf (2 n). In *International Workshop on Selected Areas in Cryptography*, pages 201–212. Springer.

Mancillas-López, C., Martínez-Herrera, A. F., and Bernal-Gutiérrez, J. A. (2020). Fpga implementation of some second round nist lightweight cryptography candidates. *Electronics*, 9(11):1940.

Massey, J. L. (1994). Guessing and entropy. In *Proceedings of 1994 IEEE International Symposium on Information Theory*, page 204. IEEE.

Matsui, M. (1993). Linear cryptanalysis method for des cipher. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 386–397. Springer.

Menezes, A. J. and Vanstone, S. A. (1993). Elliptic curve cryptosystems and their implementation. *Journal of cryptology*, 6(4):209–224.

Mennink, B. (2021). Elephant v2.

Moabalobelo, T., Nelwamondo, F. V., and Tsague, H. D. (2012). Survey on the cryptanalysis of wireless sensor networks using side-channel analysis.

Mohajerani, K., Haeussler, R., Nagpal, R., Farahmand, F., Abdulgadir, A., Kaps, J.-P., and Gaj, K. (2020). Fpga benchmarking of round 2 candidates in the nist lightweight cryptography standardization process: Methodology, metrics, tools, and results. *IACR Cryptol. ePrint Arch.*, 2020:1207.

Montgomery, P. L. (1987). Speeding the pollard and elliptic curve methods of factorization. *Mathematics of computation*, 48(177):243–264.

Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. *Decentralized Business Review*, page 21260.

Oswald, E., Mangard, S., Pramstaller, N., and Rijmen, V. (2005). A side-channel analysis resistant description of the aes s-box. In *International workshop on fast software encryption*, pages 413–423. Springer.

O'Flynn, C. and Chen, Z. D. (2014). Chipwhisperer: An open-source platform for hardware embedded security research. In *International Workshop on Constructive Side-Channel Analysis and Secure Design*, pages 243–260. Springer.

Picek, S., Batina, L., Jakobović, D., Ege, B., and Golub, M. (2014a). S-box, set, match: a toolbox for s-box analysis. In *IFIP International Workshop on Information Security Theory and Practice*, pages 140–149. Springer.

Picek, S., Papagiannopoulos, K., Ege, B., Batina, L., and Jakobovic, D. (2014b). Confused by confusion: Systematic evaluation of dpa resistance of various s-boxes. In *International Conference on Cryptology in India*, pages 374–390. Springer.

Pongaliur, K., Abraham, Z., Liu, A. X., Xiao, L., and Kempel, L. (2008). Securing sensor nodes against side channel attacks. In *2008 11th IEEE High Assurance Systems Engineering Symposium*, pages 353–361. IEEE.

Popp, T., Mangard, S., and Oswald, E. (2007). Power analysis attacks and countermeasures. *IEEE Design & test of Computers*, 24(6):535–543.

Preneel, B. and Wyseur, B. (2008). White-box cryptography. In *Dagstuhl Workshop on Security Hardware in Theory and Practice-A Marriage of Convenience, Date: 2008/06/18-2008/06/20, Location: Dagstuhl Germany.*

Prouff, E. (2005). Dpa attacks and s-boxes. In *International Workshop on Fast Software Encryption*, pages 424–441. Springer.

Pycroft, L. and Aziz, T. Z. (2018). Security of implantable medical devices with wireless connections: The dangers of cyber-attacks. *Expert Review of Medical Devices*, 15(6):403–406.

Rivest, R. L. (1994). The rc5 encryption algorithm. In *International Workshop on Fast Software Encryption*, pages 86–96. Springer.

Saarinen, M.-J. O. (2011). Cryptographic analysis of all 4× 4-bit s-boxes. In *International workshop on selected areas in cryptography*, pages 118–133. Springer.

San Pedro, M., Servant, V., and Guillemet, C. (2019). Side-channel assessment of open source hardware wallets. *IACR Cryptol. Eprint Arch.*, 2019:401.

Sasdrich, P., Bock, R., and Moradi, A. (2018). Threshold implementation in software. In *International Workshop on Constructive Side-Channel Analysis and Secure Design*, pages 227–244. Springer.

Sasdrich, P., Moradi, A., and Güneysu, T. (2016). White-box cryptography in the gray box. In *International Conference on Fast Software Encryption*, pages 185–203. Springer.

Satheesh, V. and Shanmugam, D. (2018). Secure realization of lightweight block cipher: A case study using gift. In *International Conference on Security, Privacy, and Applied Cryptography Engineering*, pages 85–103. Springer.

Schneier, B. (1993). Description of a new variable-length key, 64-bit block cipher (blowfish). In *International Workshop on Fast Software Encryption*, pages 191–204. Springer.

Selvam, R., Shanmugam, D., and Annadurai, S. (2015). Vulnerability analysis of prince and rectangle using cpa. In *Proceedings of the 1st ACM Workshop on Cyber-Physical System Security*, pages 81–87.

Shanmugam, D., Selvam, R., and Annadurai, S. (2014). Differential power analysis attack on simon and led block ciphers. In *International Conference on Security, Privacy, and Applied Cryptography Engineering*, pages 110–125. Springer.

Shannon, C. E. (1948). A mathematical theory of communication. In *Bell System Technical Journal*, volume 27, pages 379–423.

Shannon, C. E. (1949). Communication theory of secrecy systems. *The Bell system technical journal*, 28(4):656–715.

Shibutani, K., Isobe, T., Hiwatari, H., Mitsuda, A., Akishita, T., and Shirai, T. (2011). Piccolo: an ultra-lightweight blockcipher. In *International workshop on cryptographic hardware and embedded systems*, pages 342–357. Springer.

Standaert, F.-X., Malkin, T. G., and Yung, M. (2009). A unified framework for the analysis of side-channel key recovery attacks. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 443–461. Springer.

Standaert, F.-X., Peeters, E., and Quisquater, J.-J. (2005). On the masking countermeasure and higher-order power analysis attacks. In *International Conference on Information Technology: Coding and Computing (ITCC'05)-Volume II*, volume 1, pages 562–567. IEEE.

Sundaresan, V., Rammohan, S., and Vemuri, R. (2008). Defense against side-channel power analysis attacks on microelectronic systems. In *2008 IEEE National Aerospace and Electronics Conference*, pages 144–150. IEEE.

Tan, C. and Ji, Q. (2016). An approach to identifying cryptographic algorithm from ciphertext. In *2016 8th IEEE International Conference on Communication Software and Networks (ICCSN)*, pages 19–23. IEEE.

Tsalis, N., Stergiopoulos, G., Bitsikas, E., Gritzalis, D., and Apostolopoulos, T. K. (2018). Side channel attacks over encrypted tcp/ip modbus reveal functionality leaks. In *ICETE (2)*, pages 219–229.

Tsalis, N., Vasilellis, E., Mentzelioti, D., and Apostolopoulos, T. (2019). A taxonomy of side channel attacks on critical infrastructures and relevant systems. In *Critical Infrastructure Security and Resilience*, pages 283–313. Springer.

Turan, M. S., McKay, K. A., Çalik, Ç., Chang, D., Bassham, L., et al. (2019). Status report on the first round of the nist lightweight cryptography standardization

process. *National Institute of Standards and Technology, Gaithersburg, MD, NIST Interagency/Internal Rep.(NISTIR)*.

Unger, W., Babinkostova, L., Borowczak, M., and Erbes, R. (2021). Side-channel leakage assessment metrics: A case study of gift block ciphers. In *2021 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pages 236–241. IEEE.

Weatherley, R. (2021). Lightweight cryptography primitives - performance on arm cortex m3.

Wu, H. and Huang, T. (2019). Tinyjambu: A family of lightweight authenticated encryption algorithms. *Submission to the NIST Lightweight Cryptography Competition, available online at https://csrc. nist. gov/CSRC/media/Projects/Lightweight-Cryptography/documents/round-1/spec-doc/TinyJAMBU-spec. pdf*.

Xiao, Y., Hao, Q., and Yao, D. D. (2019). Neural cryptanalysis: Metrics, methodology, and applications in cps ciphers. In *2019 IEEE Conference on Dependable and Secure Computing (DSC)*, pages 1–8. IEEE.

Zhang, J., Li, L., Li, Q., Zhao, J., and Liang, X. (2021). Power analysis attack on a lightweight block cipher gift. In *Proceedings of the 9th International Conference on Computer Engineering and Networks*, pages 565–574. Springer.

Zhang, M., Raghunathan, A., and Jha, N. K. (2014). Trustworthiness of medical devices and body area networks. *Proceedings of the IEEE*, 102(8):1174–1188.

Ziemer, R. E. and Tranter, W. H. (2014). *Principles of communications*. John Wiley & Sons.

APPENDIX A:

BLACK BOX APPENDIX

A.1 Example Computations

This this section of the appendix we will display some computational examples used in our black box case study.

A.1.1 Look Up Tables

One of the pieces at the heart of our experiment was the look up tables that we used to approximate the XOR pad used in our case study. In this example we will show the first 20 look up table values on the 3^{rd} of the 4 bytes that are used in the look up tables.

0000000000000000 1cd0ab8503154f80 38a0560b072a9e01 547001910a3fed82

7040ac170e553d03 8c10579d116a8c84 a8e00223157fdb05 c4b0ada918952a86

e080582f1caa7a07 fc5003b51fbfc987 1821ae3b23d51808 34f159c126ea6789

50c104472affb70a 6c91afcd2d15068b 88615a53312a550c a43105d9343fa48d

c001b05f3854f40e dcd15be53b6a438e f8a1066b3f7f920f 1472b1f14294e190

This listing denotes the first 20 values for the third byte and they correspond with the inputs 0,1,2,...,19.

A.1.2 Addition and OTP Prediction Example

This example will how an example decryption using the Version 1 software in which the plaintext and predicted XOR pad are the same.

Consider the ciphertext with the first 8 bytes being zero and the last 4 bytes being 1,2,3 and 4 in that order. The 4 values for the look up table are as follows:

$$\text{Cipher} = 0\text{x } 00\ 00\ 00\ 00\ 00\ 00\ 00\ 00\ B_1\ B_2\ B_3\ B_4$$

$$B_1 = 0x01,\ B_2 = 0x02,\ B_3 = 0x03,\ B_4 = 0x04$$

$$Table_1(B_1) = 00\ 00\ f9\ 80\ 00\ 40\ 70\ 00$$

$$Table_2(B_2) = 00\ 00\ 00\ 00\ 00\ e0\ 00\ 00$$

$$Table_3(B_3) = 54\ 70\ 01\ 91\ 0a\ 3f\ ed\ 82$$

$$Table_4(B_4) = 00\ 70\ 2f\ f2\ 00\ 3d\ 01\ 80$$

When preforming addition in the without carry mode each byte adds the 4 input byte and takes the output mod 256. That will produce a value in the set { 0,1,2,...,255}

54 e0 29 03 0a 9c 5e 02

In the case of addition with carry we have a carry over value from bytes 3,4,6,7 and 8 resulting in:

54 e1 2b 03 0b 9d 5f 02

In practice we chose to use addition without carry. This was chosen because in the case of addition without carry the error was always higher meaning the prediction of each byte was correct or smaller that the actual value. In the case of addition with carry it has cases where is could be lower or higher making it harder to preform the prediction and error computations.

A.2 XOR Pad Values

In our computations we sampled many of the 2^{32} options for the 4 bytes that created the XOR pad used in our prediction software. A sample of some of the XOR pad values shored numerically is shown below in FigureA.1:

On observation we note that at least the first byte in not totally random as the byte value is always divisible by 4. We know that not all 2^{8*8} value are possible be

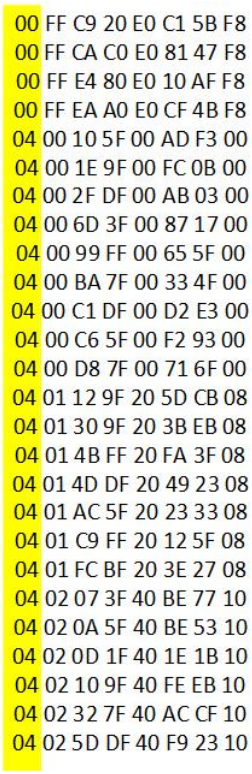

Figure A.1: Sample XOR pads sorted by value

possible because there are only 2^{4*8} values for the 4 bytes that generate the XOR pad.

A.3 Data Collection Examples

In this section we will describe the data collection process in more detail and with screenshots of the process.

To capture the actual true plaintext values we use a packet sniffer program called Wireshark (WU add citation) which is well known in the realm of computer networks. The plaintext data that is received is using the Telnet protocol and we have configure Wireshark with a filter that restricts the source IP address, port, and the length of the message. We create these restriction so we only have listed plaintext values because

we know the IP address of the server RTAC, the known port for Telnet is 23, and because we are sending a 12-byte ciphertext it will in turn be a 8-byte plaintext. A screenshot of the Wireshark filter is shown in Figure A.2 After the filter is applied one

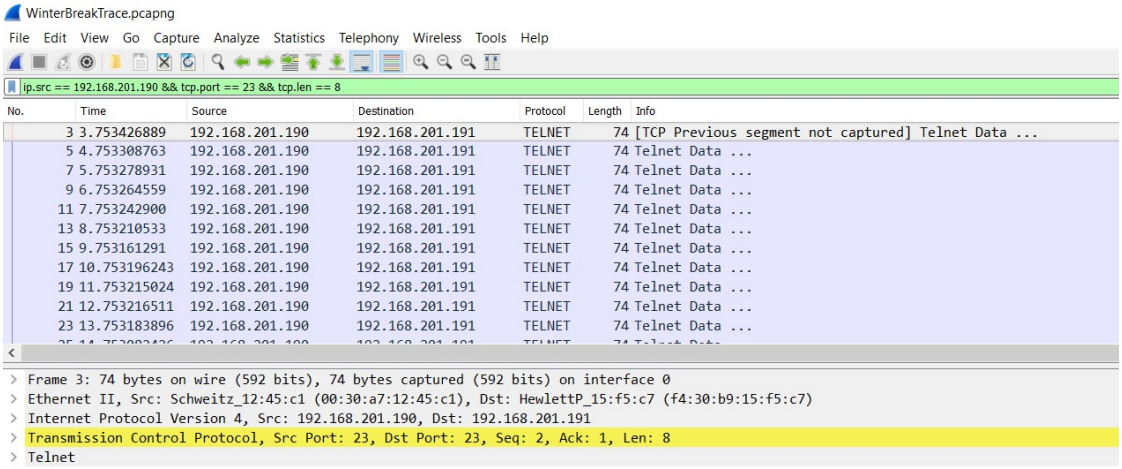

Figure A.2: Applying a Wireshark filter

needs to export the packets to a text file in preparation to be used by our prediction software. That process within the Wireshark program in shown in Figure A.3 and Figure A.4 so that only the packet bytes are exported.

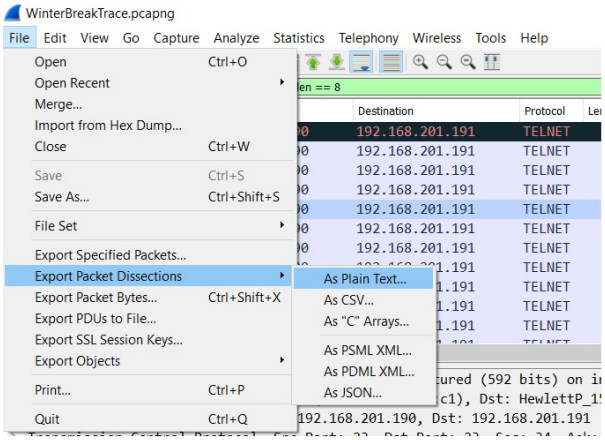

Figure A.3: Starting the Wireshark export process

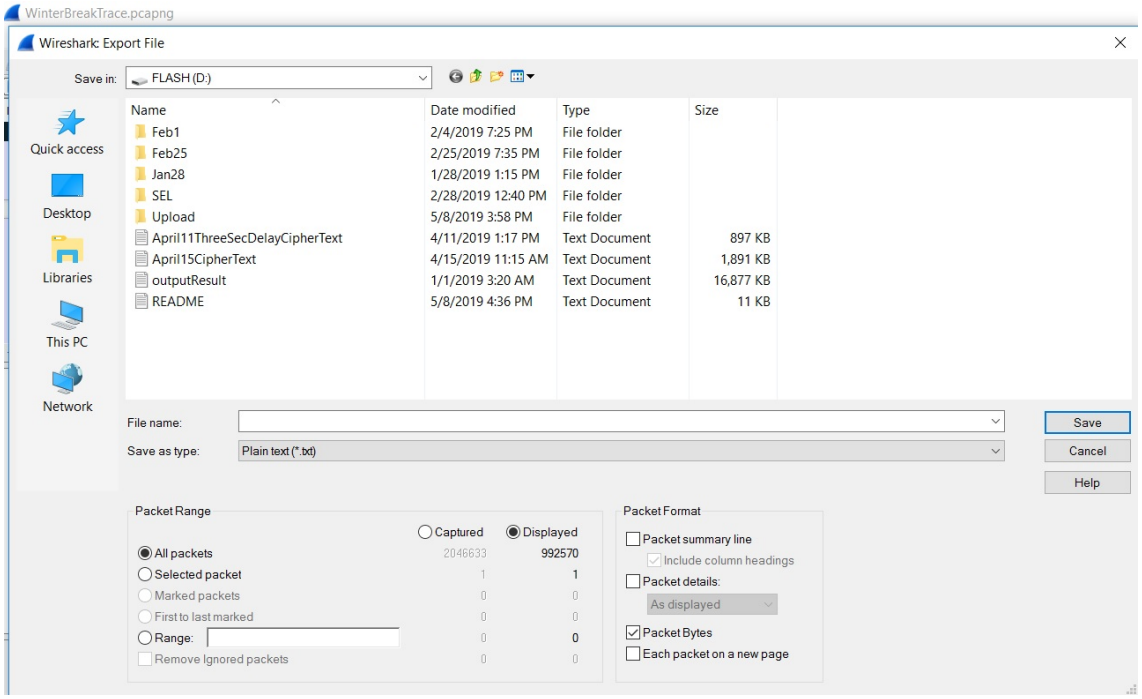

Figure A.4: Final step in exporting

After exporting to a text file the consents of the file are split up into columns containing byte numbers, hex content, and ASCII content. An Example of 3 exported packets are shown below:

0000 f4 30 b9 15 f5 c7 00 30 a7 12 45 c1 08 00 45 00 .0.....0..E...E.

0010 00 3c 73 2d 40 00 40 06 b2 bf c0 a8 c9 be c0 a8 .¡s-@.@.........

0020 c9 bf 00 17 00 17 41 85 0c 28 f8 21 63 fd 80 18 ......A..(.!c...

0030 03 91 55 26 00 00 01 01 08 0a d9 29 ed 29 9c 63 ..U& ....).).c

0040 af 99 68 9b 80 4c 6d 17 f6 db ..h..Lm...

0000 f4 30 b9 15 f5 c7 00 30 a7 12 45 c1 08 00 45 00 .0.....0..E...E.

0010 00 3c 73 2e 40 00 40 06 b2 be c0 a8 c9 be c0 a8 .¡s.@.@.........

```
0020 c9 bf 00 17 00 17 41 85 0c 30 f8 21 63 fd 80 18 ......A..0.!c...
0030 03 91 b2 d3 00 00 01 01 08 0a d9 29 f1 11 9c 63 ...........)...c
0040 be 42 e4 c6 0f a7 dc ef 0b 37                   .B.......7

0000 f4 30 b9 15 f5 c7 00 30 a7 12 45 c1 08 00 45 00 .0.....0..E...E.
0010 00 3c 73 2f 40 00 40 06 b2 bd c0 a8 c9 be c0 a8 .¡s/@.@.........
0020 c9 bf 00 17 00 17 41 85 0c 38 f8 21 63 fd 80 18 ......A..8.!c...
0030 03 91 32 cf 00 00 01 01 08 0a d9 29 f4 f9 9c 63 ..2........)...c
0040 c2 2a a0 e9 80 02 34 87 ff 4d                   .*....4..M

...
```

Our program takes in this input and with knowledge of the following structure extracts the 8 bytes that are the plaintext data part of the Telnet packet and ignores the rest of the packet. This process is the same for all 3 Versions of our prediction software.

A.4 Ciphertext Files

In our case study we generate ciphertexts and send those chosen encrypted values to the RTAC Server for decryption. The creation of the payloads are computed in java and send but we also have to keep track of the ciphertexts sent for use in our prediction program. Our program randomly selects the bytes and send them to the RTAC device but also keeps track of the ciphertexts sent in a text file

For the Version 1 content we only keep track of the last 4 bytes as the first 8 bytes are

all zeros. A sample from our Version 1 ciphertext files is shown below in which each line has 4 decimal represented bytes separated by spaces and each line represents a ciphertext. 153 110 214 15

177 127 63 60

189 44 88 40

247 138 10 162

110 21 70 231

163 168 214 36

252 64 249 3

Note that there are two new line characters in between every set of data. The extra new lines were a design choice and can be easily edited in the code.
For Versions 2 and 3 the design is similar but the file contains ciphertexts represented as 12 bytes as all 12 are random during the production of ciphertexts. An example of the ciphertext file for Versions 2 and 3 are shown below:
99 72 42 22 123 64 8 98 111 77 152 49

145 22 104 154 208 35 54 17 143 225 249 135

177 70 161 103 241 106 111 237 216 172 64 207

122 128 158 35 28 161 55 140 116 50 192 16

211 50 38 138 177 43 100 158 131 105 178 238

46 45 22 184 127 15 105 47 205 39 134 226

128 99 8 6 231 6 107 195 12 211 51 18

NOTE: Certain parts of the code use these value as int values (32-bit) and other parts use them as byte values (8-bit). In the java code an int value could have a value of (255) but that same value in the byte form would be (-1).