MATHEMATICAL COUNTERMEASURES AGAINST IMPLEMENTATION ATTACKS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF APPLIED MATHEMATICS
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

PINAR ÇOMAK

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
IN
CRYPTOGRAPHY

JANUARY 2020

Approval of the thesis:

MATHEMATICAL COUNTERMEASURES AGAINST IMPLEMENTATION ATTACKS

submitted by **PINAR ÇOMAK** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy in Cryptography Department, Middle East Technical University** by,

Prof. Dr. Ömür Uğur
Director, Graduate School of **Applied Mathematics** ____________

Prof. Dr. Ferruh Özbudak
Head of Department, **Cryptography** ____________

Prof. Dr. Ferruh Özbudak
Supervisor, **Dept. of Mathematics, METU** ____________

Prof. Dr. Vincent Rijmen
Co-supervisor, **Dept. of Electrical Engineering,** ____________
KU Leuven Univ.

Examining Committee Members:

Assoc. Prof. Dr. Ali Doğanaksoy
Department of Mathematics, METU ____________

Prof. Dr. Ferruh Özbudak
Department of Mathematics, METU ____________

Assoc. Prof. Dr. Murat Cenk
Cryptography Department, IAM, METU ____________

Assoc. Prof. Dr. Barış Bülent Kırlar
Dept. of Mathematics, Süleyman Demirel University ____________

Assist. Prof. Dr. Eda Tekin
Dept. of Business Administration, Karabük University ____________

Date: ____________

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: PINAR ÇOMAK

Signature :

ABSTRACT

MATHEMATICAL COUNTERMEASURES AGAINST IMPLEMENTATION ATTACKS

Çomak, Pınar
Ph.D., Department of Cryptography
Supervisor : Prof. Dr. Ferruh Özbudak
Co-Supervisor : Prof. Dr. Vincent Rijmen

January 2020, 68 pages

The rapid growth of the usage of connected devices demands strong methods to provide the security. In a black-box model with a trustworthy internal design, usage of secure cryptographic algorithms seems to protect the devices that stores a cryptographic key. However, the physical leakage of these devices has been shown to cause the extraction of the key or other sensitive information, in the so called gray-box model, focusing on the implementation of the cryptographic algorithms in hardware instead of the algorithms themselves. These physical leakages can be any side-channel information such as timing of an execution, power consumption or electromagnetic radiation. If the leakage is related to all or a part of the sensitive information stored in the device, it is called side-channel information. The most successfully exploited side-channels is the power consumption of the cryptosystem which can be analyzed by differential power analysis (DPA). The aim of a DPA attack using the power consumption is to extract the key or any other sensitive information from a device, through measurement and analysis of the different power consumption of different plaintext and key inputs in the cryptographic system. The attacker seeks to find a correlation between the information leaked from the side-channel and the intermediate results of a cryptographic algorithm.

Due to the wide-ranging impact of the attacks over the last two decades, preventing

Side-Channel Analysis (SCA) attacks has been the subject of a large body of research, and many countermeasures have been developed to foil these attacks. Rekeying limits the number of iterations of an algorithm using the same key, hiding changes the behavior of a device rather than the algorithm to make it consume equal amount of power for any values of sensitive information, and masking, which we study, avoids or reduce the correlation between information leaked through a side-channel and sensitive data, on which SCA attacks depends. Masking conceals all sensitive intermediate values by randomizing them with well-known secret sharing schemes. Different masking schemes have been proposed, one of which is Threshold Implementation (TI) [35]. It carries a proof of security against DPA even in the presence of glitches, while it requires smaller area and uses much less randomness compared to the other secure masking methods.

TI is based on secret sharing and multi-party computation in which each bit of the sensitive data is probabilistically divided into shares so that any proper subset of shares is independent of the data itself. TI relies on four properties; uniformity of the unmasked variables, correctness, d^{th}-order non-completeness and uniformity of the shared functions. The early works of TI provide the fact that TI is invariant under the affine transformation.

We present the contributions of this thesis in two-folds. In the first part, we examine the behavior of TI-sharing of S-boxes under a nonlinear transformation and we showed TI is not invariant under inverse transformation except for a subset of classes. In the second part of the thesis, one of the efficient method to share higher degree S-boxes which is the decomposition is studied. We examine when the S-boxes of higher degree can be decomposed into the lower degree ones. In order to find the conditions, we target the decomposition of permutations into quadratic or cubic permutations by considering the power permutations and their parities. We finally give the decomposition results about the finite fields and corresponding lower degree power permutations.

Keywords: Side-channel analysis, countermeasures, masking, the threshold implementations, Boolean functions, permutations, decomposition.

ÖZ

GERÇEKLEME ATAKLARINA KARŞI MATEMATİKSEL TEDBİRLER

Çomak, Pınar
Doktora, Kriptografi Bölümü
Tez Yöneticisi : Prof. Dr. Ferruh Özbudak
Ortak Tez Yöneticisi : Prof. Dr. Vincent Rijmen

Ocak 2020, 68 sayfa

Bağlantılı cihazların kullanımındaki hızlı gelişmeler, güvenliği sağlayan güçlü metotlara ihtiyaç doğurmuştur. Güvenilir bir iç dizayna sahip olan kara kutu modelinde, güvenilir kriptografik algoritmaların kullanımı, kriptografik anahtar depolayan cihazları koruyor gibi görünür. Ancak, bu cihazların fiziksel kaçaklarının, anahtarın ya da diğer hassas bilgilerin sızmasına sebep olduğu gösterilmiştir. Gri kutu denilen bu model, algoritmanın kendisinden ziyade kriptografik algoritmanın donanımdaki gerçeklenmesine odaklanır. Bu fiziksel sızıntılar bir programın çalışma süresi, harcadığı güç ya da yaydığı elektromanyetik radyasyon gibi yan-kanal bilgileri olabilir. Eger bu sızıntı, cihaz içinde saklanan gizli bilginin tamamıyla veya bir parçasıyla iliskiliyse, yan-kanal bilgisi olarak adlandırılır. En başarılı şekilde kullanılan yan-kanallar, diferansiyel güç analizi (DPA) ile çözümlenebilen kriptosisteminin güç tüketimidir. Güç tüketimini kullanan bir DPA saldırısının amacı, kriptografik sistemdeki farklı düz metin ve anahtar girişlerinin farklı güç tüketiminin ölçümü ve analizleriyle, cihazdaki anahtarı veya diğer hassas bilgileri çıkarmaktır. Saldırgan, yan-kanaldan sızan bilgiler ile kriptografik algoritmanın ara sonuçları arasında bir korelasyon bulmaya çalışır.

Son yirmi yılda saldırıların geniş çaplı etkisi nedeniyle, Yan-Kanal Analizi (SCA) saldırılarının önlenmesi büyük bir araştırma konusu olmuştur ve bu saldırıları engellemek için birçok önlem geliştirilmiştir. Yeniden anahtarlama, aynı anahtarı kullanarak bir algoritmanın yineleme sayısını sınırlar; gizleme, herhangi bir hassas bilgiden bağımsız olarak, her girdi için eşit miktarda güç tüketmesini sağlamak amacıyla algoritmadan ziyade aygıtın davranışını değiştirir; ve bizim çalıştığımız tedbir olan maskeleme ise bir yan-kanaldan sızan bilgiler ile hassas veriler arasındaki ilişkiyi önler veya azaltır, ki SCA saldırıları da bu korelasyona bağlıdır. Maskeleme, tüm hassas ara değerleri, iyi bilinen gizli paylaşım şemalarıyla rastgele hale getirerek gizler. Biri Eşik Gerçeklemesi (TI) [35] olan farklı maskeleme şemaları önerilmiştir. Küçük teknik problemlerin varlığında bile DPA'ya karşı güvenli olduğu kanıtlanmıştır, aynı zamanda diğer güvenli maskeleme yöntemlerine kıyasla daha küçük bir alan gerektirir ve çok daha az rastgelelik kullanır.

TI, hassas verilerin her bir bitinin eşit olasılıkla paylaşımlara ayrıldığı gizli paylaşım ve çok partili hesaplamaya dayanır, böylece herhangi bir paylaşım alt kümesi, verilerin kendisinden bağımsızdır. TI dört özelliğe, maskelenmemiş değişkenlerin tekdüzeliğine, doğruluğa, d. seviyeden tamsızlığa ve paylaşılan fonksiyonların tekdüzeliğine dayanır. TI'nin ilk çalışmaları, TI'nin afin dönüşüm altında değişmez olduğunu göstermiştir.

Bu tezin katkılarını iki bölümde sunuyoruz. İlk bölümde, S-kutularının doğrusal olmayan bir dönüşüm altında TI paylaşımının davranışını inceledik ve S-kutularının bazı sınıfları haricinde TI'nin ters dönüşüm altında değişmez olmadığını gösterdik. Tezin ikinci bölümünde, yüksek dereceli S-kutularını paylaşmanın etkili yöntemlerinden biri olan ayrışma yöntemi çalışılmıştır. Yüksek dereceli S-kutularının hangi şartlar altında daha düşük dereceli olanlara ayrıştırılabileceği incelendi. Bu koşulları bulmak için, güç permütasyonları ve pariteleri dikkate alınarak, permütasyonların kuadratik veya kübik permütasyonlara ayrılması hedeflenmiştir. Son olarak, sonlu cisimler ve bunlara karşılık gelen düşük dereceli güç permütasyonları hakkındaki ayrışma sonuçlarını sunuyoruz.

Anahtar Kelimeler: Yan-kanal analizi, tedbirler, maskeleme, alt sınır gerçeklemeleri, Boolean fonksiyonlar, permütasyonlar, ayrışım.

To healthy days, without Coronavirus

ACKNOWLEDGMENTS

Firstly, I would like to express my sincere gratitude to my advisor Ferruh Özbudak for the continuous support of my Ph.D. study and related research for his patience, motivation, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis.

Besides my advisor, I would like to thank the rest of my thesis committee: Ali Doğanaksoy, Murat Cenk, Barış Bülent Kırlar and Eda Tekin, for their insightful suggestions, comments, and encouragement to improve my thesis.

My sincere thanks goes to my co-advisor Vincent Rijmen and Svetla Nikova, who provided me an opportunity to join their team as a visitor. Without their precious support, it would not be possible to conduct this research. I could not have imagined having better mentors for my visit.

The last years have been an excellent chance for me that working in the Mathematics Department, such a friendly and cheerful atmosphere. Thank you for all the fun we have had together. Thank you, my (mostly former) colleagues: Levent and Sezen (my great office mates), Bahar, Kamil, Nisa, Özgür, Sibel, Sidre. I would especially like to thank our secretaries, Güldane, Burcu, Adem, Nejla, and Serkan, for their dedication in administrative works to keep us free of worry.

Aycan, Bahar, Eda, Ekin, Hakan, Mustaf and Nesil, you supported me along the way, thank you for being always with me, thank you all for the shared moments of deep anxiety and big excitement. Bartu, calculating the complexities of the algorithms helped me a lot, thank you. Çağdaş Çalık, thank you for your quick responses to all my questions. Cansu G. and Kübra K., I wish our METU years intersected for a more extended period, thanks for your cheerfulness.

It was also fantastic to spend many memorable days with so many wonderful people in COSIC: Péla Noë, Alan, Begül, Carl, Charlotte, Christina, Danilo, Dušan, Eleftheria, Iraklis, Lauren, Lennert, Liliya, M&M, Sara, Siemen, Thomas, Victor, Yunwen. They all made my stay at COSIC so enjoyable.

It has also been a pleasure to be a part of Ericsson Research. Thanks to all my colleagues for the great environment they provided. Leyli and Zeki, thank you for making me feel safe, hopeful, and lucky.

Burak, you deserve more than whatever I can say. Thanks for all your encouragement.

I owe biggest thanks to Nuran (for also the figures in this thesis) and Duygu for their invaluable friendship, not only during the last years but more than half of my life.

I am grateful to my siblings Mehmet, İlkay, Gülay, Gülcan and Nurcan, and their children, my father Selahattin, and my aunt Gulin, who have provided me with emotional support in my life. Thank you mom, I know you have never left me alone.

Mon Thomas,
Quand j'ai senti que je m'étais éprise de toi,
Quand j'ai réalisé à quel point j'étais fatiguée,
J'ai retrouvé mon calme, ma concentration,
Moi, Pınar, comme la source que je suis,
Coulant doucement, lentement, sûrement,
vers son destin, la mer.

I gratefully acknowledge the financial support of the TUBITAK BIDEB 2214-A Program for ensuring me to visit KU Leuven University, Belgium, during my Ph.D. studies.

The numerical calculations reported in this paper were partially performed at TUBITAK ULAKBIM, High Performance and Grid Computing Center (TRUBA resources).

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

LIST OF ABBREVIATIONS

AES	Advanced Encryption Standard
ANF	Algebraic Normal Form
CPA	Correlation Power Analysis
DES	Data Encryption Standard
DPA	Differential Power Analysis
HW	Hamming Weight
PKC	Public-Key Cryptography
S-Box	Substitution Box
SCA	Side-Channel Analysis
SNR	Signal-to-Noise Ratio
SPA	Simple Power Analysis
SPN	Substitution Permutation Network
TI	Threshold Implementation

CHAPTER 1

INTRODUCTION

The number of Internet-connected devices has been increasing globally. It is expected to be more than 75 billion Internet of Things (IoT) devices worldwide by 2025, so 60 times more connected devices than people!

At the same time, security concerns are becoming more concrete in parallel with the rapid growth of digitization, with the threat of adversaries stealing our data without authorization.

Cybercrime damages are anticipated to cost $6 trillion annually by 2021, according to Cybersecurity Ventures. Knowing that even Google and Facebook lost more than $100 million to phishing attack makes it possible to believe such estimations. For example, a ransomware attack, WannaCry alone costs The National Health Service (NHS), U.K.'s Department of Health and Social Care, totaled over $100 million, and NHS was not the only target. WannaCry hit thousands of organizations in over 150 countries in 2017. It took over infected computers and encrypted their hard drives, decrypted them in return for a payment in Bitcoin.

Mostly the lack of properly implemented cryptographic algorithms has shown to be devastating. However, on the other hand, choosing default usernames and passwords equals to giving gifts for thieves. In 2016, the Mirai botnet, one of the most significant DDoS attacks, nearly brought down the internet along the entire eastern seaboard of the U.S.

Here cryptography gets involved in the game more seriously with the security and privacy needs, although it has a long-standing and also glorious history. The security concerns make the public conscious and aware of how important it is and have raised cryptography on the list of top priorities. Mark Ward, the technology correspondent of BBC News [1], says that encryption makes the modern world go round to emphasize how important the cryptography is for everyone.

1.1 Brief Background of Cryptography

Cryptography takes its name from Greek words "kryptos" meaning hidden secret and "graphein" meaning writing. Cryptography is the practice and study of hiding information. The art or science is encompassing the principles and methods of transforming an understandable message, known as the plaintext, into one that is incomprehensible/meaningless, ciphertext, by a function that takes a key as a variable, and then re-transforming that message back to its original form. It can be shown as $C = E_{K_e}(P)$ to mean that the encryption of the plaintext P using encryption key K_e gives the ciphertext C. The output of the encryption process, the cipher is a character-for-character or bit-for-bit transformation, regardless of the linguistic structure of the message. Similarly, $P = D_{K_d}(C)$ represents the decryption of C using the decryption key K_d to get the plain text again.

The adversary, or anyone other than the intended recipient, is accepted to hear and copy the full ciphertext precisely. Unlike the intended recipient, however, he does not know what the key for decryption is and cannot easily decrypt the ciphertext. Sometimes the adversary can not only read the information from a communication channel (passive adversary) but also record and replay the messages later, inject his own messages, change or delete the original message before transmitting it to the receiver (active adversary).

The growth of computers and communications systems brought with it a demand to protect the information in digital form and to provide security services. At that point, the importance and necessity of cryptography came into our modern-day lives. It enables people to communicate on the Internet, transferring crucial and confidential information securely. Cryptography is essential because it empowers all processes, transactions and communications to be safely performed electronically. It is especially important in communicating personal information that is vulnerable to distortion over any public and private network.

Cryptography, then, not only protects data from enemy or alteration, but can also be used for

- user *authentication*, the process of proving one's identity,
- user *privacy/confidentiality*, keeping information secret from all but those who are intended to see it,
- the *data integrity*, ensuring information has not been altered in any way from the original,
- *non-repudiation*, which is the mechanism to prove that the sender really sent this message.

Of many the information security objectives such as signature, authorization, validation, anonymity and more, these four core objectives provide a framework from which others can be derived, as stated in the Handbook of Applied Cryptography [31] with more details.

1.1.1 Cryptographic Algorithms

Generally, in order to accomplish different aspects of fulfilling the objectives mentioned above, cryptographic schemes fall into a number of basic categories: Symmetric Key Cryptography, Asymmetric Key Cryptography and Hash Functions.

Symmetric/Secret Key Cryptography uses a single key for both encryption and decryption, or it is computationally easy to calculate decryption key d, knowing only encryption key e. The sender uses the key to encrypt the plaintext and sends the ciphertext to the receiver. Since the key must be kept secret from all unintended parties to keep the encrypted message secret, it is called as secret key cryptography. In most practical symmetric key encryption schemes, in order to decrypt the message to recover the plaintext, the receiver uses the same key, which gives it the alternative name "symmetric key cryptography."

Symmetric-key cryptography schemes are commonly categorized as being either *block ciphers* or *stream ciphers*.

- A block cipher is so-called because the scheme breaks up the plaintext into blocks of a fixed length and encrypts one block at a time using the same key on each block. A block cipher encrypts plaintext in fixed-size blocks. For larger messages, the easy approach is to break up the plaintext into fixed-size blocks and encrypt each separately. Block ciphers can operate this encryption in one of several modes for different applications such as Electronic Codebook, Cipher Block Chaining, Cipher Feedback and Output Feedback with varying advantages and disadvantages. Different modes of operations can provide different security guarantees. Further information can be found in [31, 36].
- Stream ciphers can be seen as block ciphers having block length equal to one. They operate on a single bit at a time and implement some form of feedback mechanism so that the key is constantly changing. In contrast to block ciphers, they tend to simultaneously encrypt groups of characters of a plaintext message using a fixed encryption transformation. Further information can be found in [36].

Since there is only one key used in encryption and decryption in Symmetric Key Cryptography, both sender and receiver must have it to communicate. Therefore, the identical key needs to be distributed securely. The concern of exchanging the key leads to public key cryptography because people do not need to transmit their private keys to the others, which reduces cybercriminals' chances of accessing the private key during transmission. This scheme helps with exchanging the secret keys in an encrypted way.

Asymmetric/Public-Key Cryptography (PKC) differs from symmetric key by using one key for encryption and another for decryption, allowing secure communication where there has been no opportunity to safely distribute keys beforehand.

In PKC, each receiver needs to publish a public key for encryption, while keeping the corresponding private key secret for decryption. Trusted third party which certifies that a particular public key belongs to a specific person or entity only is generally needed for assurance of the authenticity. In secure systems, computing the private key given public key should be computationally infeasible though private and public key pair is mathematically related.

Three well-known Public Key Encryption schemes are:

- *RSA Algorithm* is based on the difficulty of factorization problem of a large integer n. So if somebody can factorize the large number, the private key is compromised.
- *El Gamal Encryption Algorithm* is based on the difficulty of finding discrete logarithm in a cyclic group that is even if g^a and g^b are known, computing g^{ab} is difficult, which is so called Diffie-Hellman problem.
- *Elliptic Curve Cryptography* (ECC) is based on difficulty of special versions of the discrete logarithm problem. It provides efficient computation with shorter keys which ease of key management with an equivalent security level.

Hash Function is a mathematical function that converts a numerical input value into another compressed numerical value irreversibly. Hash function takes a data of arbitrary length as input and converts it into a fixed length output. Hash functions provide protection to password storage and data integrity check.

1.1.2 Brief Background of Side-Channel Analysis

A cryptographic device is an electronic device that implements a cryptographic algorithm and stores a cryptographic key. Using that key, it is able to perform cryptographic operations. There are a variety of attacks aiming to extract the key. Attacks can be categorized based on an attacker's knowledge about the cryptosystem and lead to the black-box, the grey-box and the white-box adversary models.

Traditionally, cryptographic implementations were considered as a *black box* with a trustworthy internal design in which an attacker can only observe the behavior of the inputs and outputs. In the late 90s, Kocher, Jaffe, and Jun [27] pointed out real-world attacks circumventing this model by focusing on the implementation of the cryptographic algorithms in hardware instead of the algorithms themselves. This initiated a field of research on so-called Side-Channel Analysis (SCA) attacks where implementations are considered as a *grey box*. In this *grey-box* adversary model, the attacker is able to access additional side-channel information in the form of some physical parameter (e.g., the timing of an operation or its power consumption).

The aim of an SCA attack using the power consumption side channel is to extract the key or any other sensitive information from a device, through measurement and analysis of the different power consumption of different plaintext and key inputs in the cryptographic system. The attacker, as an observant only, seeks to find a correlation between the information leaked from the side channel and the secret key. Detecting and catching such attackers is made difficult by the fact that they generally do not leave traces of their attacks.

Lastly, in the *white-box* adversary model, an attacker is allowed full access of the internal structure of the system [11]. This type of attacker is out of the scope of this thesis.

SCAs are further subdivided depending on which physical parameter, or which side channel is exploited. Timing analysis attacks use the execution time of each operation, power analysis attacks use the dynamic power consumption, electromagnetic analysis attacks exploit the electromagnetic radiation, and acoustic attacks, the sound coming out of the cryptographic device when the operations are being performed.

1.1.2.1 Power Analysis Techniques

One of the most successfully exploited side channels is the power consumption of the cryptosystem, which can be analyzed using simple, differential or correlation power analysis techniques. The target, or the sensitive variable, of the SCA, is the cryptosystem's secret key.

Simple Power Analysis (SPA). To launch an attack, an adversary measures the power consumption of a cryptosystem during a cryptographic operation as a function of time. The resulting set of instantaneous power consumption samples is called a trace. In a Simple Power Analysis attack, an attacker collects only one trace. The following example illustrates how this can be successful. During an execution of an RSA implementation, the device consumes different quantities of power in a multiplication and a squaring operations. In RSA, we decrypt by raising the ciphertext to the power of the decryption key, d. Multiplication and squaring are chosen according to the bits of the binary form of the exponent, which is the secret key, as follows: if the bit equals 1, multiply then square, if the bit equals 0, only take the square. In the power consumption trace, the attacker can distinguish between the power consumption corresponding to a multiplication (often high) and the power consumption corresponding to a squaring (often low). The alternating sequences of high and low power consumption regions in the trace can be easily interpreted and translated to the RSA key. Therefore, when conditional branching operations depend on the sensitive data, SPA can be used to break the cryptographic system.

SPA is less of a threat in symmetric-key cryptography as the type of operations in the algorithm often do not depend on the key. An attacker can, however, collect more than one trace and perform a Differential Power Analysis (DPA) attack [27] or a Correlation Power Analysis (CPA) attack [15].

Differential Power Analysis (DPA). The attacker collects a large number of power consumption traces corresponding to a cryptographic operation under a fixed key and different plaintexts. The traces need to be correctly aligned, meaning fixed points of the trace always correspond to the power consumption of exactly the same sequence of operations [29]. DPA is based on a divide-and-conquer strategy, where smaller subsets of the key are attacked separately. When attacking AES, the key is often attacked byte by byte. For each key byte, a guess is made and the output of its corresponding S-box is calculated using the known plaintext. The S-box output value is then interpreted using a leakage model, or an abstraction of how the device's power consumption behaves. The Hamming Weight (HW) of the S-box output can for instance be chosen and corresponds to the abstraction that a byte of high HW leads to a higher power consumption then a byte with a low HW. The HWs resulting from the key guess and each plaintext is then correlated to the values in the power consumption trace corresponding to that plaintext and the highest correlation leads to the correct key byte. When the Pearson correlation coefficient is used for correlation, this type of attack is referred to as a Correlation Power Analysis (CPA) attack [15].

1.1.2.2 Countermeasures against SCA Attacks

Hardening symmetric-key cryptosystems against the latter class of attacks forms the topic of our research. Due to the wide-ranging impact of these attacks over the last two decades, preventing side-channel attacks has been the subject of a large body of research, and many countermeasures have been developed to foil these attacks. We give a brief overview of successful mitigation strategies below.

Rekeying. Since the attacker needs a large number of power traces if the same key to mount DPA successfully, one can consider to limit the number of iterations of an algorithm using the same key as a countermeasure. Nevertheless, the additional challenges in key management make this method impractical.

Since SCA attacks depend on the relationship between information leaked through a side-channel and sensitive data, a popular approach of countermeasures is to avoid or reduce this dependency. The following two categories are successful examples of this strategy.

Hiding. Hiding aims to break the correlation between the leaked information and the processed data values and operations by changing the behavior of device rather than the algorithm. The hardware on which the algorithm is implemented could be made to consume an equal amount of power for all values, regardless of their Hamming Weight. A drawback of this approach is the high cost of the resulting hardware and a long design time [29].

An alternative class of countermeasures relies on adapting the algorithm rather than the device on which it is implemented. The effort of an attacker to retrieve the key can be increased by randomizing the order of executions of the cryptographic operations or by inserting dummy operations [29]. Recent research has shown that these algorithmic alterations are easily defeated using machine learning, and are therefore not sufficient to protect devices against DPA.

Masking. An effective countermeasure against DPA is found in another data randomization technique known as masking. Masking is a way of securing cryptosystem even if the attacker obtains and analyzes a large number of power traces. Instead of randomizing the order of operations, masking randomizes the intermediate values using well-known secret sharing schemes [42]. Each bit of the sensitive data is probabilistically divided into shares so that any proper subset of shares is independent of the data itself. In our DPA example, an attacker trying to retrieve a key byte now has to make guesses on all shares of that key byte to succeed. The more shares a key is shared with, the harder the attack becomes. This leads us to concept of higher-order DPA which we illustrate informally as follows. An unmasked implementation can be broken using a guess on a *single* intermediate value, which is what we referred to as DPA earlier, and which we here generalize as *first-order* DPA. To thwart first-order DPA, we can share the key over *two* shares. An attacker guessing a single intermediate value can not retrieve the secret key anymore. However, if the attacker makes a guess on the *two* shares, or when the attacker launches a *second-order* DPA, the key can be retrieved. To thwart this second-order DPA, the key can be shared over three shares, requiring an attacker to make a guess on the three shares. In a d$^{\text{th}}$-order masking scheme, d^{th}*-order* DPA attacks are thwarted by using at least $d+1$ randomized shares of data. Higher-order DPA attacks become exponentially more difficult to mount with each order putting a practical limit on the number of shares needed for a secure implementation. For a rigorous study of (higher-order) DPA we refer the reader to the work of Mangard et al. [29].

The generic principle of masking consists in randomly splitting all intermediate variables processed during the execution of the cipher into s shares such that $X = X_1 \odot X_2 \odot \cdots \odot X_{s-1} \odot X_s$. The operators corresponds to a group operation in classical examples are Boolean masking based on the exclusive-or (XOR) and arithmetic masking based on modular addition. One of the most common masking techniques is the Boolean masking which randomizes key-dependent intermediate values of an algorithm. The first masked implementations of DES and RSA were given in [24], and independently, Chari et al. [20] proposed a generic masking technique to create provably resistant implementations against DPA.

The designer of a cryptosystem has to tailor the masking scheme to the implementation target. When the algorithm is to be executed on a general-purpose processor, for instance a Central Processing Unit (CPU), a software-based masking scheme is used. Several software masking schemes have been proposed and implemented [3, 24, 23], and have shown to offer provable security in a leakage model [39, 40, 41]. The increasing complexity of many processors and their common lack of transparency has made it harder for programmers to approximate the correct leakage model and therefore harder to correctly mask the algorithm. Additionally, masking an algorithm in software results in very high overheads in terms of program memory and execution time.

In many cryptosystems, a dedicated co-processor is responsible for the cryptographic operations. This co-processor is specifically designed in hardware for high security and its implemented algorithms are therefore secured with hardware masking schemes. Similar to software masking schemes, it is important for a designer to be able to approximate the leakage model of the hardware correctly. The first generation of hardware masking schemes [26] required a strict adherence to the execution order of intermediate values, which in hardware was not trivial to achieve due to the presence of so-called glitches. Glitches are temporary values that are present in the circuit due to different arrival times of input signals to various transistors. As glitches consume power, they can reveal additional side-channel information and can degrade the level of order of the implemented masking scheme [30]. Enforcing the correct execution order on hardware masking schemes has shown to increase the execution time, the power consumption, the size and the cost of the resulting implementation. As designing countermeasures with limited impact on these parameters is key for their adoption in cryptosystems, a new masking scheme called Threshold Implementations (TI) was introduced in 2006 [35]. It carried a proof of security against first-order DPA needing minimal assumptions on the hardware, which directly lead to its security in the presence of glitches at lower implementation costs. In 2014, TI was generalized to be secure against higher-order DPA [6].

1.2 Research Questions

In 1949, for a secure cryptographic algorithm, Claude Shannon [43] gave two properties that a secure cryptosystem should have: confusion and diffusion. Confusion means that the key does not relate to the ciphertext in a simple way while diffusion means that one bit change in plaintext should spread over the entire ciphertext. The simplest way to achieve both confusion and diffusion is a substitution-permutation network (SPN) on which many modern block ciphers are constructed based. Confusion is achieved by Substitution Boxes (S-box), generally the only nonlinear part of cryptographic algorithms while Permutation Boxes (P-box) mixes the sub-blocks linearly and provide diffusion of the system.

Unlike the TI-sharing of any linear/affine function, TI of an S-box can be challenging [5]. Fortunately, though, an affine transformation of any S-box can be shared easily by affine transformation of the TI-sharing of the original S-box. Using the equivalence classes listed by De Canniere [16], several efficient methods to have first-order TI-sharings of small S-boxes (3×3 and 4×4) with different number of shares have been provided in [8, 9]. Then in [6] provable security of TI is extended from resisting first-order DPA to higher-order DPA.

The fact that TI is invariant under the affine transformation raises our first question addressed by the thesis research:

Research Question 1: *Can we do the same with a nonlinear transformation? For example can we take the inverse of a TI-sharing of an S-box and obtain the sharing of the inverse of the original S-box? Or is there a relation between them?*

This question has spawned an exploratory search. TI-sharing with different number of shares (3, and partially 4 and 5) is applied not only to the representative S-box (3-bit, 4-bit and partially 5-bit) of each equivalence class but also to each S-box in every class in order to find the answer of this question.

It is showed that the number of shares is increasing with the degree of function together with the order of the TI [5]. One of the efficient method to share a higher degree S-box is the decomposition of it into lower degree ones. The (cubic) S-box of the block cipher PRESENT [10] is decomposed into two quadratic S-boxes in [37].

In order to limit the area increase of the protected implementations, it is important to keep the number of shares as low as possible. This observation brings the following research question:

Research Question 2: *Under which conditions can we decompose the S-boxes of* *higher degree into the lower degree S-boxes?*

In this thesis, we target the decomposition of permutations into quadratic or cubic permutations to be able to use fewer shares. Stafford's Theorem [44] has made us consider the power permutations and their parities in order to investigate when a permutation over a finite field can be decomposed into permutations of lower degree.

1.3 Organization of the Thesis

This thesis is made up of four chapters. Below is a brief description of each chapter's content and the contribution within each. They are divided according to the research question they answered.

Chapter 1. The first chapter, i.e., this chapter, briefly introduces the importance of cryptography in modern life with the increasing number of devices and the issues about their security. It is mentioned shortly that security objectives that are needed to protect the information in digital form and to provide security. In order to fulfill these objectives, different cryptographic systems are summed up. Then some background for SCA attacks and TI masking scheme is given. This chapter ends with our research questions together with this section, which provides an overview of the content.

We present our answers across two chapters.

Chapter 2. In the second chapter called "Threshold Implementation", firstly, the related notations used during the work are given. Then the properties of TI masking and some methods to have an efficient one are provided with varying examples for each. One of them, the decomposition method, is given more detail since it is one of our focus in the thesis. The concept of Substitution boxes is described, then behavior of TI-sharing of S-boxes under affine and inverse transformation are examined. The chapter is concluded with our results.

Chapter 3. In the third chapter called "Decomposition of Permutation", firstly the related notations used during the work and some well-known theorems are mentioned. How the parity of a power permutation is related to its decomposition is established. A special case of a power permutation x^k over a finite field $\mathbb{F}_q$ is stated with its proof.

For a more general acquisition, we delve into the cycle decomposition of permutations and we explore the relations between its disjoint cycles. Then, we count them and give the results in a lemma before giving a toy example to be given to clarify. Finally, the main theorem is presented and applied for varying power permutations over different finite fields. The comparison with a previous method is given. The chapter ends with our experimental and remarkable results.

Chapter 4. In the fourth and last chapter, we sum up what we did during this work. We conclude this thesis by listing open questions for future research.

CHAPTER 2

THRESHOLD IMPLEMENTATION

In this chapter, the properties of a Threshold Implementation (TI) of any function which is used to prevent any device that leaks the intermediate values against d$^{\text{th}}$-order Differential Power Analysis (DPA) are given.

In the following we will give some basic background information, notations which are needed.

In (s, n) secret sharing schemes, a secret X is distributed partly among n players such that at least s players of all n, are needed in order to recover the secret uniquely. Therefore, in a perfect (s, n) secret sharing scheme, knowledge of up to $s - 1$ shares does not give any additional information on the secret value.

We use the upper-case characters to represent elements in $GF(2^n)$ or $\mathbb{F}_2^n$ for some positive integer n, where $\mathbb{F}_2$ is the finite field with two elements. In order to specify each bit of an element X, we will use the n-tuple form with lower-case characters $X = (x, y, z, w, \dots) \in \mathbb{F}_2^n$, where $x, y, z, \dots \in \mathbb{F}_2$, as the input of a multi-output, or more generally vectorial, Boolean function F which is defined from $\mathbb{F}_2^n$ to $\mathbb{F}_2^m$. In block cipher designs, those functions are used in the substitution layer, as Substitution boxes, shortly S-boxes. Similarly, to specify each coordinate function, called single-output Boolean functions corresponding to the case $m = 1$, of the vectorial Boolean function F, and we will use the m-tuple of Boolean functions $(f(X), g(X), h(X), r(X), \dots)$, which are called the coordinate functions of F, where $f(X), g(X), \dots$ are defined from $\mathbb{F}_2^n$ to $\mathbb{F}_2$.

There are several different ways to represent a vectorial function, the one which is the most commonly used in cryptography is the Algebraic Normal Form (ANF) representation.

Each coordinate Boolean functions $f, g, h, \ldots$ of a vectorial Boolean function F is uniquely written as a polynomial depending on n variables, say $x, y, z, w, \ldots$, whose degrees are at most one, and with coefficients from $\mathbb{F}_2$. Similarly F is uniquely written in the same form but with coefficients a_I from $\mathbb{F}_2^m$ such as

$$F(X) = \bigoplus_{I \in \mathcal{P}(N)} a_I X^I$$

where $\mathcal{P}(N)$ denotes the power set of $N = \{1, \ldots, n\}$. This polynomial is called the algebraic normal form of F. Here one can observe that

$$a_I = \bigoplus_{X \in \mathbb{F}_2^n / supp(X) \subseteq I} F(X)$$

and conversely

$$F(X) = \bigoplus_{I \subseteq supp(X)} a_I$$

where $supp(X)$, the support of the binary vector X, is the set of coordinate positions in which X has nonzero entries, i.e., for $X = (1, 0, 0, 1)$ we have $supp(X) = \{1, 4\}$.

The algebraic degree of the function is the global degree of its ANF by $d^oF = max\{|I||a_I \neq (0, \ldots, 0), I \in \mathcal{P}(N)\}$ as defined in [17]. Therefore, it is the maximal algebraic degree of the coordinate functions of F, usually denoted by t. Note that if the function F is a permutation of $\mathbb{F}_{2^n}$, then $deg(F) \leq n - 1$.

In block ciphers design, one can see that they consist two different building blocks; small non-linear components, called S-boxes, which are typically used to obscure the relationship between the key and the ciphertext to satisfy Shannon's confusion property [43], and large linear diffusion layers, called Permutation boxes, which diffuse this non-linearity over the complete block, which yields the substitution-permutation network (SPN).

In the following, although the properties of TI are considered for the functions defined as $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$ where n not necessarily equal to m, in this thesis we consider the TI of the permutation function $F : \mathbb{F}_2^n \to \mathbb{F}_2^n$, so n-bit permutations, some of which define $n \times n$ invertible Substitution boxes (S-boxes) that are used in cryptographic algorithms. They can be seen as (n, n) vectorial Boolean functions defined over a vector space $\mathbb{F}_2^n$.

2.1 Properties of Threshold Implementation

There are four properties of a TI. Two of these, *uniform masking* and *correctness*, which means that all input shares are chosen from an independent and identically distributed uniform random source and the output of an unshared function is equal to XOR of the output of the shared function, respectively, are common in all the masking schemes algorithms to compute on masked data. The third property is *d*th*-order non-completeness*, where d is the order of side-channel resistance. In a dth-order non-complete sharing, any combination of d component functions must be independent of at least one input share. The last property, *uniform sharing* states that the output shares of the component functions should be uniform.

2.1.1 Uniform Masking

First we split a variable $X \in \mathbb{F}_2^n$ into s shares, we will use s_{in} to distinguish it is an input share, $X_i \in \mathbb{F}_2^n$ such that $X = \bigoplus_i X_i$ which satisfying (s_{in}, s_{in}) secret sharing schemes, i.e., all s_{in} shares are required to recover the secret. We denote the shared vector of the s_{in} shares X_i by the bold character as $\mathbf{X} = (X_1, X_2, \cdots, X_{s_{in}})$, where $\mathbf{X} \in \mathbb{F}_2^{ns_{in}}$, *the masking or the sharing of the unmasked variable* X. Without loss of generality the shares $X_1, \cdots, X_{s_{in}-1}$ are randomly chosen variables from a *uniform* distribution and the last share $X_{s_{in}}$ is calculated such that $X = \bigoplus_{i=1}^{s_{in}} X_i$ holds. Hence, the knowledge of up to $s_{in} - 1$ shares does not reveal any information on X.

Let $Sh(X)$ denote the set of valid share vectors $\mathbf{X} \in \mathbb{F}_2^{ns_{in}}$ for $X \in \mathbb{F}_2^n$ such that

$$Sh(X) = \{\mathbf{X} \mid X_1 \oplus X_2 \oplus \cdots X_{s_{in}} = X\}.$$

Property 1: A masking **X** is *uniform* if and only if for all X the corresponding vectors $\mathbf{X} \in Sh(X)$ with masked values occur with the same probability, actually with the probability $p = |\mathbb{F}|^{n(1-s_{in})}$.

Uniformity of a masking implies the independence of the combination of any $s_{in} - 1$ shares from the unmasked value hence, satisfying an (s_{in}, s_{in}) secret sharing scheme [5]. In [17], it is defined that the sharing of F is balanced. In other words, a vectorial Boolean function $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$ is balanced if each $A \in \mathbb{F}_2^m$ has 2^{n-m} preimages.

Moreover, if the unshared function is a permutation, the shared function should also be a permutation.

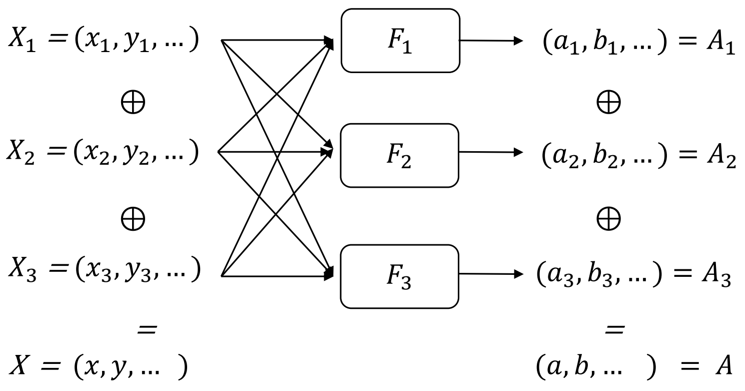
2.1.2 Correctness

In order to implement a function $A = F(X)$ defined from $\mathbb{F}_2^n$ to $\mathbb{F}_2^m$, the TI method requires a set of s_{out} functions F_i which together compute the output(s) of F.

The function $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$, that takes X as an input and gives A as an output, is also split into s shares; we will use s_{out} to distinguish it is an output share, as $F_i : \mathbb{F}_2^{ns_{in}} \to \mathbb{F}_2^{ms_{out}}$, which are called component functions. We denote the shared function of the s_{out} shares F_i by the bold character again as $\mathbf{F} = (F_1, F_2, \cdots, F_{s_{out}})$. The sharing **F** must satisfy the following property for a correct implementation.

Property 2: A sharing **F** is *correct* if and only if for all $A \in \mathbb{F}_2^m$, $A = F(X)$ implies that $A = \bigoplus_{i=1}^{s_{out}} A_i = \bigoplus_{i=1}^{s_{out}} F_i(\mathbf{X})$ for all **X** satisfying $X = \bigoplus_{i=1}^{s_{in}} X_i$ and $X \in \mathbb{F}_2^n$.

Hence the output of an unshared function equals the sum of shared output of a shared function. In the figure 2.1, we give an example with $s_{in} = 3$ and $s_{out} = 3$ to be correct.



$X_1 = (x_1, y_1, \ldots)$
$\oplus$
$X_2 = (x_2, y_2, \ldots)$
$\oplus$
$X_3 = (x_3, y_3, \ldots)$
$=$
$X = (x, y, \ldots)$
F_1
F_2
F_3
$(a_1, b_1, \ldots) = A_1$
$\oplus$
$(a_2, b_2, \ldots) = A_2$
$\oplus$
$(a_3, b_3, \ldots) = A_3$
$=$
$(a, b, \ldots) = A$

Figure 2.1: Correctness of a masking

The uniform masking of the input and correctness of a sharing are common properties for all masking schemes. For higher-order DPA security on hardware in the presence of glitches, the sharing needs two other properties given in the following sections.

2.1.3 dth-order non-completeness

Most of the masking schemes, all s_{in} input shares are being used in at least one of their component functions. It implies when an attacker probes the corresponding wire, he can observe all the information about sensitive data, which implies solving the (s_{in}, s_{in}) secret sharing. Therefore, it is desired to have the component functions implemented in such a way that their leakages are independent of each other.

Unlike other masking schemes, the threshold implementation provides d wires which are depended on at most $s_{in} - 1$ shares, which are independent of the sensitive information. Hence, probing d wires can only give the information from at most $s_{in} - 1$ shares, i.e., for the first-order non-complete TI, i^{th} output share F_i is independent of one input share, without loss of generality, say the i^{th} input share X_i, then it implies that the power consumption of F_i is also independent of X_i. This property together with uniform masking makes the d^{th}-order DPA infeasible.

Property 3: A sharing is called *d^{th}-order non-complete* if any combination of up to d component functions F_i of **F** must be independent of at least one input share.

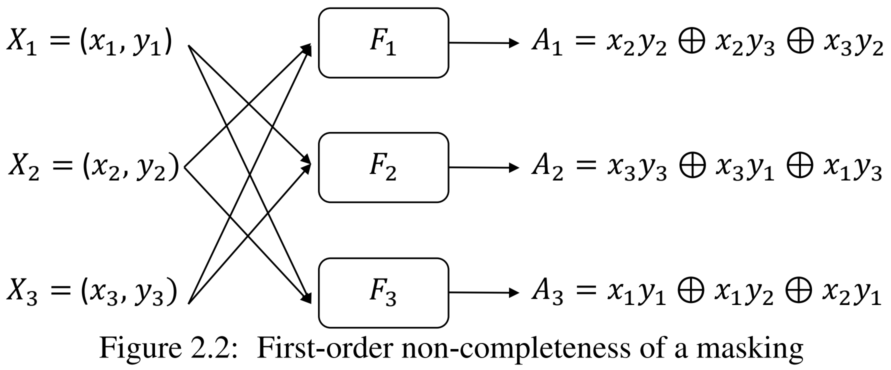

$X_1 = (x_1, y_1)$
$X_2 = (x_2, y_2)$
$X_3 = (x_3, y_3)$
F_1
F_2
F_3
$A_1 = x_2y_2 \oplus x_2y_3 \oplus x_3y_2$
$A_2 = x_3y_3 \oplus x_3y_1 \oplus x_1y_3$
$A_3 = x_1y_1 \oplus x_1y_2 \oplus x_2y_1$

Figure 2.2: First-order non-completeness of a masking

This property of TI provides the combination of leakages resulting from the process of d component functions to be independent of the sensitive variable X given a uniform sharing X.

In [5], it is proved that for a shared function **F**, if the input masking is uniform and **F** satisfies the correctness and the d^th^-order non-completeness, then probing d or less wires of the circuit implementing **F**, one can not reveal the unmasked input value X, at least one independent input share of the input, even in the presence of glitches or delayed inputs. Therefore, this enables security on demanding non-ideal circuits.

Hence, this implies that a TI of a function F, which satisfies uniform masking, correctness and non-completeness properties is secure against d^{th}-order DPA, so there is no leakage of information in this circuit.

2.1.4 Uniform sharing of a function

In 2.1.1, we see that a sharing **F** requires the input **X** to be uniform. However, the uniformity of the input **X** does not imply that the output a is also uniform. In the case that the outputs are not uniform, and if the output is used as the input of a second function, let say G, used in the following rounds, then it might leak information since the second masking **A** is not uniform, and **A** should also be uniform for the second one to be secure, where $\mathbf{A} = (A_1, A_2, \cdots, A_{s_{out}})$.

Therefore, in order to guarantee the uniformity of the input sharing on the following rounds, we need to make sure that the input of the second sharing **G**, is also uniform, as in first property 2.1.1. This is equivalent to say that the first sharing **F** should be a uniform sharing of the function F.

Property 4: The *d^{th}-order sharing* $\mathbf{F} = (F_1, \cdots, F_{s_{out}})$ is uniform if and only if $\forall X \in \mathbb{F}_2^n$, $\forall A \in \mathbb{F}_2^m$ such that $F(X) = A$, $\forall \mathbf{A} \in Sh(A)$ and $s_{out} \geq d+1$,

$$|\{\mathbf{X} \in Sh(X)|\mathbf{F(X)} = \mathbf{A}\}| = \frac{|\mathbb{F}|^{n(s_{in}-1)}}{|\mathbb{F}|^{m(s_{out}-1)}}$$

where $\mathbf{F(X)} = (F_1(\mathbf{X}), \cdots, F_{s_{out}}(\mathbf{X})) = (A_1, \cdots, A_{s_{out}}) = \mathbf{A}$.

In the case $s_{in} = s_{out} = s$, we have, for all $(A_1, A_2, \cdots, A_s)$ satisfying $\bigoplus A_i = A$, the number of valid sharing $\mathbf{X} \in \mathbb{F}_2^{ns_{in}}$ for which $F_j(\mathbf{X}) = A_j$, $1 \leq j \leq s$, is equal to $2^{(s-1)(n-m)}$ times the number of $X \in \mathbb{F}_2^n$ for which $A = F(X)$.

It follows, in the case $s_{in} = s_{out} = s$ and $n = m$, there is one valid sharing and also in

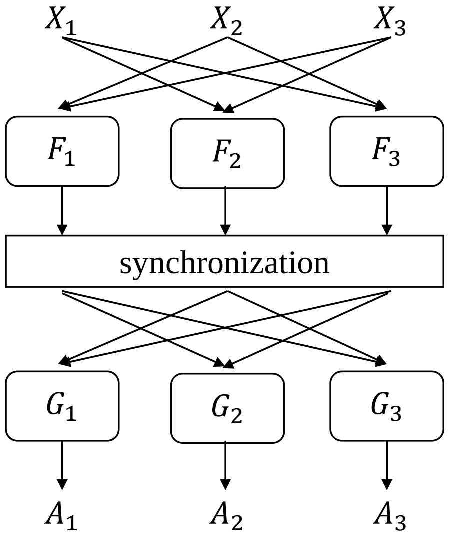


Figure 2.3: Uniform Sharing

this case a uniform sharing **F** is a permutation on $\mathbb{F}_2^{ns}$ if and only if F is a permutation on $\mathbb{F}_2^{n}$.

To preserve d^{th}-order security we require the outputs of F's component functions to be stable at their final value before forwarding them to G's component functions. We impose this by using a synchronization layer between the component functions of F and G [38] which can be seen in the Figure 2.3. In many implementations this layer takes the form of registers.

This property is the least trivial one to be achieved. In order to achieve uniformity, several methods are given in [8, 9]. However, the sharings which do not satisfy this property can also be used in TI. In the case one does not have a uniform sharing, it can be used the re-masking, also known as adding fresh randomness, proposed in [4, 7, 32], one can increase the number of shares or the function can be decomposed, some of them will be detailed in the end of the following section.

2.2 Sharing Examples with Different Number of Shares

It has been shown in [6], that there always exists a d$^{\text{th}}$-order non-complete sharing of a degree t function f with $s_{in} \geq td + 1$ input shares. This implies that given a security order together with the degree of the function increases the required number of shares. Since more shares cause an increase in required resources such as area, we need to keep the number of shares as low as possible. Fortunately, any permutation of high degree can be written as a combination of quadratic or cubic functions, with details in the Chapter 3.

In the same work, a method for generating the component functions with $s_{in} = td+1$ input and $s_{out} = \binom{s_{in}}{t}$ is provided. However, a TI using these number of shares is not the only possible sharing and also it does not guarantee that s_{out} or the number $s_{in} + s_{out}$ is minimum.

For an adequate explanation of various methods for the TI-sharings of functions with various degrees, we have provided the following examples taken from the work [5].

In the light of the above findings, *an affine function* $F(X) = A$ can be implemented with $s \geq d+1$ component functions to thwart d^th^-order DPA.

1. As an example, for the affine function $F(X) = 1 + X$, we can implement as follows: $F_1(X_1) = 1 + X_1$, $F_2(X_2) = X_2, \cdots, F_s(X_s) = X_s$ with s input and output shares. Clearly if X_i's are chosen randomly where $\bigoplus_{i=1}^{s} X_i = X$, uniform masking property 2.1.1 is satisfied, since $\bigoplus_{i=1}^{s} F_i = F$, correctness property 2.1.2 is satisfied and also the non-completeness property 2.1.3 is satisfied with $s = d+1$, since every component function depends on only one share. X_i's are uniform, so the outputs of the component functions. therefore it has also the fourth property 2.1.4.

The fact that higher degree of the function causes the higher number of shares can be seen in the following examples, for the function $F : \mathbb{F}_2^3 \to \mathbb{F}_2$ and the variables $x, y, z \in \mathbb{F}_2$.

2. As a quadratic example, consider $A = F(X) = F(x, y, z) = 1 \oplus y \oplus xz$. The following sharing is a first-order TI with $s_{in} = 2$ and $s_{out} = 4$, if the inputs x, y and z are independent of each other, since each function gives information from at most one share of each input.

$$A_1 = 1 \oplus y_1 \oplus x_1 z_1$$

$$A_2 = y_2 \oplus x_1z_2$$
$$A_3 = x_2z_1$$
$$A_4 = x_2z_2.$$

3. As another quadratic example, consider $A = F(x, y, z) = 1 \oplus x \oplus xy \oplus xz$.

   (a) The first order sharing of this function will be similar with $s_{in} = 2$ and $s_{out} = 4$ in the following

   $$\begin{aligned} A_1 &= 1 \oplus x_1 \oplus x_1y_1 \oplus x_1z_1 \\ A_2 &= x_1y_2 \oplus x_1z_2 \\ A_3 &= x_2 \oplus x_2y_1 \oplus x_2z_1 \\ A_4 &= x_2y_2 \oplus x_2z_2. \end{aligned}$$

   (b) The second order sharing of the function with $s_{in} = 3$ and $s_{out} = 9$ is as follows

   $$\begin{aligned} A_1 &=1 \oplus x_1 \oplus x_1y_1 \oplus y_1z_1 & A_4 &=x_2z_1 \oplus y_2z_1 & A_7 &=x_3z_1 \oplus y_3z_1 \\ A_2 &=x_1z_2 \oplus y_1z_2 & A_5 &=x_2 \oplus x_2z_2 \oplus y_2z_2 & A_8 &=x_3z_2 \oplus y_3z_2 \\ A_3 &=x_1z_3 \oplus y_1z_3 & A_6 &=x_2z_3 \oplus y_2z_3 & A_9 &=x_3 \oplus x_3z_3 \oplus y_3z_3. \end{aligned}$$

4. The following examples show the ways that one can share the same function with different number of input and output shares. Consider the quadratic function $A = F(x, y, z) = 1 \oplus y \oplus xy \oplus xz \oplus yz$.

   (a) with $s_{in} = 2$ and $s_{out} = 6$ is

   $$\begin{aligned} A_1 &= 1 \oplus y_1 \oplus x_1y_1 \oplus x_1z_1 \oplus y_1z_1 & A_4 &= x_2y_2 \oplus x_2z_2 \oplus y_2z_2 \\ A_2 &= x_1z_2 \oplus y_1z_2 & A_5 &= x_1y_2 \\ A_3 &= y_2 \oplus x_2z_1 \oplus y_2z_1 & A_6 &= x_2y_1. \end{aligned}$$

   Since this sharing causes an undesired increase in the number of output shares, one can increase the number of input shares to have less number of output shares which can be seen in the following.

(b) with $s_{in} = s_{out} = 3$, which satisfy $s_{in} = td + 1$ input and $s_{out} = \binom{s_{in}}{t}$ output shares, the first order sharing is as follows

$$\begin{aligned}
A_1 =& 1 \oplus y_2 \oplus (x_2y_2 \oplus x_2y_3 \oplus x_3y_2) \oplus (x_2z_2 \oplus x_2z_3 \oplus x_3z_2) \\
& \oplus (y_2z_2 \oplus y_2z_3 \oplus y_3z_2) \\
A_2 =& y_3 \oplus (x_3y_3 \oplus x_3y_1 \oplus x_1y_3) \oplus (x_3z_3 \oplus x_3z_1 \oplus x_1z_3) \\
& \oplus (y_3z_3 \oplus y_3z_1 \oplus y_1z_3) \\
A_3 =& y_1 \oplus (x_1y_1 \oplus x_1y_2 \oplus x_2y_1) \oplus (x_1z_1 \oplus x_1z_2 \oplus x_2z_1) \\
& \oplus (y_1z_1 \oplus y_1z_2 \oplus y_2z_1).
\end{aligned}$$

This sharing is an example for *first-order direct sharing* with three shares for a quadratic function. The linear terms including the indices i, i.e., y_i, the quadratic terms including the indices i and $i + 1$ together, i.e., x_iy_{i+1}, and finally the ones including the indices only i, i.e., x_iy_i, appear in the component function F_{i-1} in a cyclic manner.

(c) The previous sharing of the function $A = F(x, y, z) = 1 \oplus y \oplus xy \oplus xz \oplus yz$ does not satisfy the uniform sharing property 2.1.4, while the following sharing gives the TI-sharing with same number of shares using the *Correction Terms* which is detailed in 2.2.1.

$$\begin{aligned}
A_1 =& 1 \oplus y_2 \oplus (x_2y_2 \oplus x_2y_3 \oplus x_3y_2) \oplus (x_2z_2 \oplus x_2z_3 \oplus x_3z_2) \\
& \oplus (y_2z_2 \oplus y_2z_3 \oplus y_3z_2) \oplus \boldsymbol{y_2} \boldsymbol{\oplus} \boldsymbol{y_3} \\
=& 1 \oplus \boldsymbol{y_3} \oplus (x_2y_2 \oplus x_2y_3 \oplus x_3y_2) \oplus (x_2z_2 \oplus x_2z_3 \oplus x_3z_2) \\
& \oplus (y_2z_2 \oplus y_2z_3 \oplus y_3z_2) \\
A_2 =& y_3 \oplus (x_3y_3 \oplus x_3y_1 \oplus x_1y_3) \oplus (x_3z_3 \oplus x_3z_1 \oplus x_1z_3) \\
& \oplus (y_3z_3 \oplus y_3z_1 \oplus y_1z_3) \oplus \boldsymbol{y_3} \boldsymbol{\oplus} \boldsymbol{y_1} \\
=& \boldsymbol{y_1} \oplus (x_3y_3 \oplus x_3y_1 \oplus x_1y_3) \oplus (x_3z_3 \oplus x_3z_1 \oplus x_1z_3) \\
& \oplus (y_3z_3 \oplus y_3z_1 \oplus y_1z_3) \\
A_3 =& y_1 \oplus (x_1y_1 \oplus x_1y_2 \oplus x_2y_1) \oplus (x_1z_1 \oplus x_1z_2 \oplus x_2z_1) \\
& \oplus (y_1z_1 \oplus y_1z_2 \oplus y_2z_1) \oplus \boldsymbol{y_1} \boldsymbol{\oplus} \boldsymbol{y_2} \\
=& \boldsymbol{y_2} \oplus (x_1y_1 \oplus x_1y_2 \oplus x_2y_1) \oplus (x_1z_1 \oplus x_1z_2 \oplus x_2z_1) \\
& \oplus (y_1z_1 \oplus y_1z_2 \oplus y_2z_1).
\end{aligned}$$

5. The following is another first-order direct sharing example with four shares of the cubic function $A = F(x, y, z) = 1 \oplus x \oplus xy \oplus xyz$ with $s_{in} = s_{out} = 4$

$$\begin{aligned}
A_1 =& 1 \oplus x_2 \oplus (x_2y_2 \oplus x_2y_3 \oplus x_2y_4 \oplus x_4y_3) \oplus (x_2y_2z_2 \oplus x_2y_3z_2 \oplus x_2y_2z_3 \\
&\oplus x_2y_3z_4 \oplus x_2y_4z_3 \oplus x_2y_2z_4 \oplus x_2y_4z_2 \oplus x_2y_4z_4 \oplus x_2y_3z_3 \oplus x_4y_3z_2 \\
&\oplus x_3y_4z_2 \oplus x_4y_2z_3 \oplus x_3y_2z_4 \oplus x_4y_3z_3 \oplus x_4y_4z_3 \oplus x_4y_3z_4) \\
A_2 =& x_3 \oplus (x_3y_3 \oplus x_3y_4 \oplus x_3y_1 \oplus x_1y_4) \oplus (x_3y_3z_3 \oplus x_3y_4z_3 \oplus x_3y_3z_4 \\
&\oplus x_3y_4z_1 \oplus x_3y_1z_4 \oplus x_3y_3z_1 \oplus x_3y_1z_3 \oplus x_3y_1z_1 \oplus x_3y_4z_4 \oplus x_1y_4z_3 \\
&\oplus x_4y_1z_3 \oplus x_1y_3z_4 \oplus x_4y_3z_1 \oplus x_1y_4z_4 \oplus x_1y_1z_4 \oplus x_1y_4z_1) \\
A_3 =& x_4 \oplus (x_4y_4 \oplus x_4y_1 \oplus x_4y_2 \oplus x_2y_1) \oplus (x_4y_4z_4 \oplus x_4y_1z_4 \oplus x_4y_4z_1 \\
&\oplus x_4y_1z_2 \oplus x_4y_2z_1 \oplus x_4y_4z_2 \oplus x_4y_2z_4 \oplus x_4y_2z_2 \oplus x_4y_1z_1 \oplus x_2y_1z_4 \\
&\oplus x_1y_2z_4 \oplus x_2y_4z_1 \oplus x_1y_4z_2 \oplus x_2y_1z_1 \oplus x_2y_2z_1 \oplus x_2y_1z_2) \\
A_4 =& x_1 \oplus (x_1y_1 \oplus x_1y_2 \oplus x_1y_3 \oplus x_3y_2) \oplus (x_1y_1z_1 \oplus x_1y_2z_1 \oplus x_1y_1z_2 \\
&\oplus x_1y_2z_3 \oplus x_1y_3z_2 \oplus x_1y_1z_3 \oplus x_1y_3z_1 \oplus x_1y_3z_3 \oplus x_1y_2z_2 \oplus x_3y_2z_1 \\
&\oplus x_2y_3z_1 \oplus x_3y_1z_2 \oplus x_2y_1z_3 \oplus x_3y_2z_2 \oplus x_3y_3z_2 \oplus x_3y_2z_3).
\end{aligned}$$

6. Consider the function $A = F(x, y) = xy$.

   (a) The first-order direct TI-sharing with $s_{in} = s_{out} = 3$ is as follows.

$$\begin{aligned}
A_1 =& x_2y_2 \oplus x_2y_3 \oplus x_3y_2 \\
A_2 =& x_3y_3 \oplus x_1y_3 \oplus x_3y_1 \\
A_3 =& x_1y_1 \oplus x_1y_2 \oplus x_2y_1.
\end{aligned}$$

   Note that the direct sharing method does not guarantee that it satisfies the property 2.1.4 which can be seen in the following table for this function. It is a relatively easy example to form the table here for the function with less shares and variables.

		(A_1, A_2, A_3)							
		(0,0,0)	(0,1,1)	(1,0,1)	(1,1,0)	(0,0,1)	(0,1,0)	(1,0,0)	(1,1,1)
(x, y)	(0,0)	7	3	3	3	0	0	0	0
	(0,1)	7	3	3	3	0	0	0	0
	(1,0)	7	3	3	3	0	0	0	0
	(1,1)	0	0	0	0	5	5	5	1

Table 2.1: Number of times that a sharing (A_1, A_2, A_3) occurs for a given input (x, y)

If the input masking (x, y) is uniform, then the sharing of F is distributed as shown in Table 2.1. For example, for the pair $(x, y) = (1, 0)$, there are 16 uniform 3-sharings; and seven of these sharings gives the value $(0, 0, 0)$, the values $(0, 1, 1)$, $(1, 0, 1)$ and $(1, 1, 0)$ occur three times for each, and the others does not occur under the calculation with the direct sharing of F stated above. However, as stated in 2.1.4, in order to have a uniform sharing of **F** for a given A, the number of sharing **X** must be 4, i.e., $|\{\mathbf{X} \in Sh(X)|$ **F(X)** $= A\}| = 2^{2(3-1)}/2^{1(3-1)} = 4$, which is not satisfied for this sharing, since the table contains elements different from 0 and 4. If the outputs of this function is used as the input of a second circuit, it might leak the sensitive information because now the input masking is not uniform.

(b) If one *increase the number of shares*, we have a first order TI-sharing satisfying all the properties which can be seen in the following.

$$\begin{aligned}
A_1 =&(x_3 \oplus x_4)(y_2 \oplus y_3) \oplus y_2 \oplus y_3 \oplus y_4 \oplus x_2 \oplus x_3 \oplus x_4 \\
A_2 =&(x_1 \oplus x_3)(y_1 \oplus y_4) \oplus y_1 \oplus y_3 \oplus y_4 \oplus x_1 \oplus x_3 \oplus x_4 \\
A_3 =&(x_2 \oplus x_4)(y_1 \oplus y_4) \oplus y_2 \oplus x_2 \\
A_4 =&(x_1 \oplus x_2)(y_2 \oplus y_3) \oplus y_1 \oplus x_1.
\end{aligned}$$

(c) Also by *decreasing the number of input shares*, one can have a uniform sharing with $s_{in} = 2$ and $s_{out} = 4$ as follows.

$$\begin{aligned}
A_1 &= x_1 y_1 & A_3 &= x_2 y_1 \\
A_2 &= x_1 y_2 & A_4 &= x_2 y_2.
\end{aligned}$$

(d) Increasing the number of input variables of the function F, i.e., adding extra variable z to F without influencing the output of the function as a *virtual variable*, may make a sharing uniform. The shares of this variable are called as *virtual shares*. The sharing with the variable z which is also uniform, one can re-define the sharing of F as if the function is now $F = F(X) = F(x, y, z)$.

$$A_1 = x_2y_2 \oplus x_2y_3 \oplus x_3y_2 \oplus x_2z_2 \oplus x_3z_3 \oplus y_2z_2 \oplus y_3z_3$$
$$A_2 = x_3y_3 \oplus x_1y_3 \oplus x_3y_1 \oplus x_3z_3 \oplus x_1z_1 \oplus y_3z_3 \oplus y_1z_1$$
$$A_3 = x_1y_1 \oplus x_1y_2 \oplus x_2y_1 \oplus x_1z_1 \oplus x_2z_2 \oplus y_1z_1 \oplus y_2z_2.$$

(e) We have a uniform sharing in this example by *varying the number of shares*. The previous examples have less number of input shares than the outputs. It is also possible to have a sharing with $s_{in} > s_{out}$, satisfying the all TI properties.

$$A_1 = (x_2 \oplus x_3 \oplus x_4)(y_2 \oplus y_3) \oplus y_4$$
$$A_2 = (x_1 \oplus x_3)(y_1 \oplus y_4) \oplus x_1y_3 \oplus x_4$$
$$A_3 = (x_2 \oplus x_4)(y_1 \oplus y_4) \oplus x_1y_2 \oplus x_4 \oplus y_4.$$

2.2.1 Achieving Uniform Sharing

There is no straightforward method to share a function so that the uniformity property holds. Hence, each d$^{\text{th}}$-order non-complete sharing, should be checked explicitly in order to assure the uniformity property 2.1.4. Fortunately, the sharings which do not satisfy this property can also be used in TI.

In order to achieve uniformity, several methods are given in [8, 9]. However, the sharings which do not satisfy this property can also be used in TI.

In the case one does not have a uniform sharing by these methods, it can be used the correction terms, re-masking, also known as adding fresh randomness, proposed in [4, 7, 32], one can vary the number of shares or the function can be decomposed.

- One can use the *correction terms* to have a uniform sharing which can be seen in the example 4c. The correction terms can be added in pairs to more than one share in a way that they still satisfy the non-completeness property. Since the terms in a pair cancel each other, the sharing does not break the correctness property. However, to find a term that corrects the sharing is challenging because of the huge amount of possible choices of sharing with different correction terms.

- One can *increase the number of shares* which can be seen in the example 6b. However, since the area requirements of an implementation of an algorithm increase with the number of shares, this method should not be considered as an efficient way. Also note that this is an example that the boundaries for s_{in} and s_{out}, i.e., $s_{in} = td + 1$ and $s_{out} = \binom{s_{in}}{t}$, are not the optimal ones.

- *Decreasing the number of input shares* makes the sharing uniform which can be seen in the example 6c. However, although it has 4 output shares, it is secure against only first-order DPA.

  When $d > 1$, it can be seen that the number of output shares is greater than the number of input shares. Since this causes an undesired increase in area, we need to decrease the number of shares. One can find some methods given in [6] for this aim.

- In the example 6e, we can see that *varying the number of shares* makes the sharing uniform. It does not decrease the number of elements used, but it has other advantages [5].

- To make a nonuniform sharing of a function uniform, one can use *re-masking* by introducing extra fresh randomness in the circuit with details in [32].

  If the output $A = (A_1, A_2, A_3)$ of the nonuniform sharing F is followed by the re-masking operation r that uses two random masks m_1, m_2, then the output sharing $A' = (A_1', A_2', A_3')$ is uniform without breaking the other properties of TI.

$$\begin{aligned} A_1' &= r_1(A_1, m_1) = A_1 \oplus m_1 \\ A_2' &= r_2(A_2, m_2) = A_2 \oplus m_2 \\ A_3' &= r_3(A_3, m_1, m_2) = A_3 \oplus m_1 \oplus m_2. \end{aligned}$$

However, since finding good masks may not be easy, and also due to the undesired cost increased of high throughput fresh random number generation, it should not be considered as straightforward method to make a nonuniform sharing uniform, [4].

This technique is used in the first and second order TI of the Advanced Encryption Standard's (AES) S-box in order to guarantee the uniformity in [21].

Another technique called *using virtual variable and virtual shares* as in the example 6d makes the sharing uniform. Since it can be seen as a kind of adding randomness to the shared function, we can place this technique under this section.

- Together with having a uniform sharing, we also need to keep the number of shares as low as possible. Since the numbers are increasing with the degree of the function and security order, we can *decompose* the nonlinear functions into lower degree ones having a uniform TI-sharing with less shares. The following example is more detailed since we focus on this method in the Chapter 3. It can be considered as a background information for the next chapter.

  An example of the decomposing functions for the lightweight encryption algorithm PRESENT S-box is given in [37]. In the substitution layer of PRESENT, the S-box, denoted by $S(X)$, which is composed of four Boolean functions, three of which are cubic and one is quadratic, with the following truth table is used:

X	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
S[x]	C	5	6	B	9	0	A	D	3	E	F	8	4	7	1	2

Table 2.2: The truth table of the PRESENT S-box $S(X)$.

The block cipher PRESENT is described in [10] with its design and the analyses of its security and detailed performance.

The ANF of $S(X) : \mathbb{F}_2^4 \rightarrow \mathbb{F}_2^4$ is as follows. Here $X = (x, y, z, w)$ is the input of $S(X)$ and $S = (s, t, u, v)$ is the output, where s, t, u and v are Boolean functions.

$$s = 1 + x + z + w + yz + xyw + xzw + yzw,$$

$$t = 1 + x + y + xz + xw + zw + xyw + xzw,$$
$$u = x + z + xy + xz + xyw + xzw + yzw,$$
$$v = x + y + w + yz.$$

It can be seen that the algebraic degree of the S-box $S(X)$ is three, i.e., the maximum number of variables in any term that appears in the component functions is three, for example f contains the term xyw of three variables. It is decomposed into two S-boxes $F(X)$ and $F'(X)$ with degree two such that $S(X) = F(F'(X))$, where $S, F, F' : \mathbb{F}_2^4 \to \mathbb{F}_2^4$, and $F'(0)$ is considered as 0 in order to speed up the search. Note that since S is a bijection, so permutation, both F and F' are also bijections.

The quadratic S-boxes $F(X)$ and $F'(X)$ for a decomposition satisfying $S(X) = F(F'(X))$, which are used in [37] for the experiment, have the following truth table:

X	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
F[x]	0	8	B	7	A	3	1	C	4	6	F	9	E	D	5	2
F'[x]	7	E	9	2	B	0	4	D	5	C	A	1	8	3	6	F

Table 2.3: Truth tables of the S-boxes $F(X)$ and $F'(X)$.

One can calculate the algebraic normal forms of $F(X)$ and $F'(X)$ which are given by

$$F(x,y,z,w) = (f,g,h,r), \qquad F'(x,y,z,w) = (f',g',h',r')$$
$$f = y + z + w + xw \qquad f' = y + z + w$$
$$g = x + zw \qquad g' = 1 + y + z$$
$$h = y + z + xw \qquad h' = 1 + x + z + yw + zw$$
$$r = z + yw. \qquad r' = 1 + w + xy + xz + yz.$$

For the first order TI-sharings of the functions F and F', the input is split into three shares such that $X = X_1 + X_2 + X_3$ where $X, X_i \in \mathbb{F}_2^4$, where $i = 1, 2, 3$ and the output is also split into three shares as $F(X_1+X_2+X_3) = F_1(X_2, X_3) + F_2(X_1, X_3) + F_3(X_1, X_2)$ where $F_i : \mathbb{F}_2^8 \to \mathbb{F}_2^4$. The same sharing, which can be seen in the following figure, can be observed for the other function F':

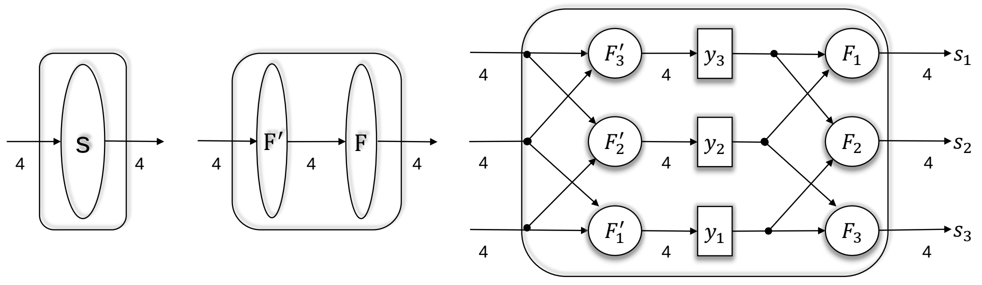


Figure 2.4: Decomposition of PRESENT S-box

It is a correct and non-complete sharing, i.e., $F = F_1 + F_2 + F_3$ and the shared function F_i is independent from the variable X_i for all i, respectively. In order to have a uniform masking and uniform sharing, we can say that $(X_1, X_2, X_3) \mapsto (F_1(X_2, X_3), F_2(X_1, X_3), F_3(X_1, X_2))$ is a 12-bit permutation. If the input X and its shares X_i's are denoted by the 4-bit vectors (x, y, z, w) and (x_i, y_i, z_i, w_i) for $i = 1, 2, 3$, respectively, we have the sharing as follows

$$\begin{aligned}
&(x_1, x_2, x_3, x_4, y_1, y_2, y_3, y_4, z_1, z_2, z_3, z_4, w_1, w_2, w_3, w_4) \mapsto \\
&\qquad (F_1(x_2, y_2, z_2, w_2, x_3, y_3, z_3, w_3), F_2(x_1, y_1, z_1, w_1, x_3, y_3, z_3, w_3), \\
&\qquad\qquad\qquad\qquad\qquad\qquad F_3(x_1, y_1, z_1, w_1, x_2, y_2, z_2, w_2)
\end{aligned}$$

Since the output of the nonlinear function F' is used as an input to the nonlinear function F, the output of F' is stored in the registers. The authors of [37] choose the decomposition that gives the most efficient hardware implementation in terms of area, among 907,200 possible decompositions of $S(X)$ that satisfy all properties of TI.

They shared the functions according to the *direct sharing method* stated in the example 4b. This sharing satisfy the uniformity condition, so they do not need to use any methods given in the section 2.2.1, such as re-masking or correction terms, in order to make it uniform. The sharings with the ANFs of the shared functions are as follows:

$$\begin{aligned}
&F_1(x_2, y_2, z_2, w_2, x_3, y_3, z_3, w_3) = (f_1, g_1, h_1, r_1) \\
&\quad f_1 = y_2 + z_2 + w_2 + x_2w_2 + x_2w_3 + x_3w_2, \\
&\quad g_1 = x_2 + z_2w_2 + z_2w_3 + z_3w_2, \\
&\quad h_1 = y_2 + z_2 + x_2w_2 + x_2w_3 + x_3w_2,
\end{aligned}$$

$r_1 = z_2 + y_2w_2 + y_2w_3 + y_3w_2.$

$F_2(x_1, y_1, z_1, w_1, x_3, y_3, z_3, w_3) = (f_2, g_2, h_2, r_2)$

$f_2 = y_3 + z_3 + w_3 + x_3w_3 + x_1w_3 + x_3w_1,$

$g_2 = x_3 + z_3w_3 + z_1w_3 + z_3w_1,$

$h_2 = y_3 + z_3 + x_3w_3 + x_1w_3 + x_3w_1,$

$r_2 = z_3 + y_3w_3 + y_1w_3 + y_3w_1.$

$F_3(x_1, y_1, z_1, w_1, x_2, y_2, z_2, w_2) = (f_3, g_3, h_3, r_3)$

$f_3 = y_1 + z_1 + w_1 + x_1w_1 + x_1w_2 + x_2w_1,$

$g_3 = x_1 + z_1w_1 + z_1w_2 + z_2w_1,$

$h_3 = y_1 + z_1 + x_1w_1 + x_1w_2 + x_2w_1,$

$r_3 = z_1 + y_1w_1 + y_1w_2 + y_2w_1.$

$F'_1(x_2, y_2, z_2, w_2, x_3, y_3, z_3, w_3) = (f'_1, g'_1, h'_1, r'_1)$

$f'_1 = y_2 + z_2 + w_2,$

$g'_1 = 1 + y_2 + z_2,$

$h'_1 = 1 + x_2 + z_2 + y_2w_2 + y_2w_3 + y_3w_2 + z_2w_2 + z_2w_3 + z_3w_2,$

$r'_1 = 1 + w_2 + x_2y_2 + x_2y_3 + x_3y_2 + x_2z_2 + x_2z_3 + x_3z_2 + y_2z_2 + y_2z_3 + y_3z_2.$

$F'_2(x_1, y_1, z_1, w_1, x_3, y_3, z_3, w_3) = (f'_2, g'_2, h'_2, r'_2)$

$f'_2 = y_3 + z_3 + w_3,$

$g'_2 = y_3 + z_3,$

$h'_2 = x_3 + z_3 + y_3w_3 + y_1w_3 + y_3w_1 + z_3w_3 + z_1w_3 + z_3w_1,$

$r'_2 = w_3 + x_3y_3 + x_1y_3 + x_3y_1 + x_3z_3 + x_1z_3 + x_3z_1 + y_3z_3 + y_1z_3 + y_3z_1.$

$F'_3(x_1, y_1, z_1, w_1, x_2, y_2, z_2, w_2) = (f'_3, g'_3, h'_3, r'_3)$

$f'_3 = y_1 + z_1 + w_1,$

$$g'_3 = y_1 + z_1,$$
$$h'_3 = x_1 + z_1 + y_1w_1 + y_1w_2 + y_2w_1 + z_1w_1 + z_1w_2 + z_2w_1,$$
$$r'_3 = w_1 + x_1y_1 + x_1y_2 + x_2y_1 + x_1z_1 + x_1z_2 + x_2z_1 + y_1z_1 + y_1z_2 + y_2z_1.$$

One can observe that the sharing of one of the component functions of F, without loss of generality consider the first component Boolean function $f = y + z + w + xw$, is:

$$f_1 = y_2 + z_2 + w_2 + x_2w_2 + x_2w_3 + x_3w_2$$
$$f_2 = y_3 + z_3 + w_3 + x_3w_3 + x_1w_3 + x_3w_1$$
$$f_3 = y_1 + z_1 + w_1 + x_1w_1 + x_1w_2 + x_2w_1.$$

In [37], one can find the details about hardware architectures of five different profiles that are used to attack, yielding different levels of side-channel resistance, and the corresponding implementation results. To achieve the perfect resistance against first order DPA, they combine the data masking with the key masking.

So far, several methods are provided to generate a sharing that satisfies the uniform masking, correctness, non-completeness and uniform sharing properties together, of which the last one is not trivial. Then some methods are given to obviate this problem. More TI-sharing examples can be found in [5] with different number of input and output shares for the functions of different degrees and with different security orders with more details.

2.3 On Substitution Boxes

In this thesis we consider the TI of the permutation function $F : \mathbb{F}_2^n \to \mathbb{F}_2^n$, so n-bit permutations or $n \times n$ invertible S-boxes. In order to find the TI-sharing of the permutations, we need to classify all permutations; however, since classifying all n-bit permutations is a challenging problem even for small values of n and it is getting harder exponentially, because for example the number Boolean functions of n variables is 2^{2^n}, an equivalence relation is needed.

2.3.1 Affine Equivalence of Permutations

The need to avoid examining all permutations brings with it the need to find an equivalence relation in which the relevant properties are invariant. Therefore, we will consider the concept of *affine equivalence* of the permutations in the following.

Definition 2.3.2. *Consider two $n \times n$ permutations $S_1(X)$ and $S_2(X)$. If there exists a pair of affine mappings $A_l(X)$ and $A_r(X)$ such that $S_1 = A_l \circ S_2 \circ A_r$, the S-boxes S_1 and S_2 are said to be affine equivalent.*

Note that the algebraic degree, parity (odd/even number of transpositions to represent the permutation), linearity and differential uniformity are invariant within the equivalence class.

By the definition, since every affine permutation $A(X)$ can be written as $A \times X \oplus a$ where a is an n-bit constant and A is an $n \times n$ invertible matrix over $\mathbb{F}_2$, one can write $S_1(X) = A_l \times S_2(A_r \times X \oplus a_r) \oplus a_l$ where $X \in \mathbb{F}_2^n$. Therefore, it can be seen that the number of affine permutations is 2^n times the number of nonsingular Boolean $n \times n$ matrices; hence, it increases exponentially with the increase of n.

A classification based on the affine equivalence of permutations is described in [16] to reduce the search area. According to the Canniere's work, the numbers of equivalence classes for the corresponding permutations with the given values of n are given.

Since all 2-bit permutations are affine, there is only one class, and it contains the affine permutations defined in $\mathbb{F}_{2^2}$. For 3-bit permutations, there are 4 equivalence classes with an affine class and 3 classes with quadratic permutations. For 4-bit permutations, there are 302 equivalence classes containing 1 class of affine, 6 classes of quadratic and the remaining 295 classes of cubic permutations. Then in [14], for 5-bit quadratic permutations it is showed that there exist 75 affine equivalence classes.

After these classifications, a technique to find TI-sharing of small S-boxes are provided in [8].

Theorem 2.3.3. *[5] If a uniform d^{th}-order TI-sharing of a permutation F which is a representative of an affine equivalence class is known, a uniform d^{th}-order TI-sharing of any permutation from the same equivalence class can be derived.*

Therefore, the TI-sharing of an affine transformation of an S-box can be derived easily by affine transformation of the TI-sharing of the original S-box. With the technique provided in [5] and the help of the theorem 2.3.3, TI-sharing of all 3×3 and 4×4 invertible S-boxes, with 2,3 and 4 shares, and TI-sharing of 5×5 invertible S-boxes, with 3 shares, are provided.

This theorem raised a question: "Can a sharing of a nonlinear transformation of an S-box satisfy the properties of TI-sharing". As an example, we consider the inverse operation in our work. Hence now the question is: "can we take the inverse of a TI-sharing of an S-box and obtain the sharing of the inverse of the original S-box".

2.3.4 Inverse S-boxes

The aim of this chapter is to examine the relation between inverse of TI-sharings of S-boxes and TI-sharings of inverses of the corresponding S-boxes. Note that S^{-1}, the inverse of an S-box S, and the S-box itself are not necessarily in the same equivalence class, which may imply that they do not have the same algebraic degrees. With the help of the following lemma, we can provide some information about their degrees.

Lemma 2.3.5. *[13] Let F be a permutation in $\mathbb{F}_{2^n}$, then $deg(F^{-1}) = n - 1$ if and only if $deg(F) = n - 1$.*

The inverse of an affine permutation is also affine for any n. It implies that the permutation and its inverse are in the same class since there is only one affine class for permutations in $\mathbb{F}_{2^n}$ for any n. So, the affine permutations are affine equivalent to their inverses. In the case $n = 3$, by the Lemma 2.3.5, the inverse of a quadratic permutation is also quadratic. Furthermore, each permutation belongs to the class of its inverse for 3×3 S-boxes. Such permutations are called self-inverse, or involution. Again by the lemma above, for $n = 4$, the inverse of a cubic and a quadratic permutation is also cubic and quadratic, respectively, but not necessarily self-inverse. In this case, 172 classes are self-inverse while the other 130 classes are not, and these 65 pairs of inverse classes, which consist an S-box in the first class and its inverse in the second class, are listed in [8]. Many examples that a function and its inverse function have different algebraic degrees can be given for $n = 5$.

For a given TI-shared S-box, its inverse sharing exists because of the uniformity, however it might not be a TI-sharing since non-completeness might not be satisfied. In other words the TI inverse might not be a TI-sharing of the inverse.

In this study, we have focused on the question that investigate the conditions which guarantee that the inverse of the TI-shared S-box is still a TI-sharing.

We are trying to determine when the inverse of the TI-shared S-box is a TI-sharing of an inverse S-box. We implement our algorithm using C++. We used the CONDOR system provided by COSIC/KU Leuven and TRUBA system by TÜBİTAK because we can conclude our search only if we use a system which gives us opportunity to execute our program in parallel.

We are using the following algorithm:

loop over a set of chosen S-boxes, denoted by S

i. Find the TI uniform sharing with k shares of $n \times n$ S-box S, denoted by $kn \times kn$ SS with same algebraic degree t.

ii. Then we check for uniformity of SS. Note one way is to use the "inverse" and check whether it is a permutation. Namely using the fact: SS^{-1} is a permutation if and only if the TI-sharing of SS is uniform. If it is not uniform we go back to the step i. Otherwise continue to iii.

iii. Find the inverse SS^{-1} and inverse of S, i.e., S^{-1}. Note that SS^{-1} should be still uniform and correct sharing of S^{-1}.

iv. Compute the algebraic degree t_1 of S^{-1} and t_2 of SS^{-1}. If $t_1 = t_2$ continue to the step v below, else stop since SS^{-1} cannot be a TI sharing of S^{-1}. Main problem is how to check the non-completeness of SS^{-1} and whether it is correct masking of S^{-1}.

v. A way to do this is to use the ANF of SS^{-1}, i.e., get the coefficients of the ANF. For each coordinate function (there are kn such Boolean functions) verify on how many variables it depends. Record/remember which coordinate function from which variables it depends. Note that if the non-completeness is satisfied then each such function should depend of max $(k-1)n$ variables. If there is a coordinate function which depends on more than $(k-1)n$ variables then stop SS^{-1} cannot be a TI-sharing of S^{-1}, otherwise go to vi. Note that the check is not finished yet.

vi. The question is how to choose the tuples of k variables of SS^{-1} which will be the shares of 1 variable in S^{-1}. And at the same time how to choose the tuples of k coordinate functions of SS^{-1} which will be the shares of 1 coordinate function in S^{-1}. The remembered/recorded in step v relations between coordinate functions and variables come at help to reduce the exhaustive search. Namely there should be at least k coordinate functions of SS^{-1} which do not depend of a particular tuple of at least k variables. These set of at least k coordinate functions are the candidates, as well as the corresponding tuple of at least k variables. In other words the set of kn coordinate functions and the set of kn variables are split on such subsets among which we look for exactly k tuples.

This search is exhaustive, i.e., we choose n tuples of size k for the coordinate functions and at the same time we choose n tuples of size k for the variables.

For each such choice we obtain the pre-shared version of SS^{-1} denoted by S_1 then verify whether $S_1 = S^{-1}$. Record/remember those solutions (if any) which result in $S_1 = S^{-1}$ and finish the exhaustive search step.

vii. If there are no solutions, finish. In other words, SS^{-1} can't be a TI-sharing of S^{-1}.

viii. For each of the found "solution" we are now sure we have both "correctness" and "non-completeness". So, collect the cases when this happens.

end loop

The algorithm above is for a single S-box. Then we extend this by applying it for each S-box in a given affine class. When applied inside a class, we know how the proper sharings can be obtained from the first one in the class using the affine permutations.

2.4 Research results

The search of 3-bits S-boxes with 3,4 and 5 shares is completed in approximately 1, 3 and 10 hours respectively while the search of 4-bits S-boxes with 3 shares seems to be completed more than 4 months. We have stored the results which have taken a couple hundred gigabytes in space.

We searched for all possible 3-bits S-boxes, we shared them with 3,4 and 5 shares, and all possible 4-bits S-boxes are shared with 3-shares.

Note that 3-bit S-boxes have 4 different classes, the affine class, named Class 0, has 1344 S-boxes; and the classes including quadratic S-boxes, Class 1 has 9408 S-boxes, Class 2 has 18816 S-boxes and Class 3 has 10752 S-boxes. 4-bit S-boxes have an affine class, Class 0 has 329280 S-boxes, the classes including quadratic S-boxes Class 4 has 3628800, Class 12 has 270950400, Class 293 has 270950400 S-boxes, Class 294 has 203212800 and the Class 299 has 232243200 S-boxes.

Our experimental results are as follows.

i. According to the search about TI-sharing of S-boxes with 3-shares, the inverse of TI-sharing of 3-bits S-box is also a TI-sharing of an inverse S-box if and only if it lies in Class 0 or Class 1. The inverses of the TI-sharings of the S-boxes in Class 2 are not TI-sharing. The TI-sharing of the S-boxes in Class 3 do not exist with 3-shares.

ii. The search about TI sharing of 3-bits S-boxes with 4 and 5-shares shows that the inverse of TI sharing of 3-bits S-box is also a TI sharing of an inverse S-box if and only if it lies in Class 0 or Class 1. The inverses of the TI sharings of the S-boxes in Class 2 and in Class 3 are not TI sharing.

iii. According to the search about TI-sharing of 4-bits S-boxes with 3-shares, TI-sharings of the S-boxes in the Classes 0, 4, 12, 293, 294 and 299 exist. The inverse of TI-sharing of an S-box is also a TI-sharing of an inverse S-box if and only if it lies in Class 0 or Class 4. The inverses of the TI-sharings of the S-boxes in Class 12, in Class 293, in Class 294 and in the Class 299 are not TI-sharing.

These results shows that TI-sharing is not invariant under the inverse transformation except for a subset of classes.

CHAPTER 3

DECOMPOSITION OF PERMUTATION

The goal of this chapter is to analyze the nonlinear components of symmetric cryptographic algorithms. As mentioned in previous chapters 1, 2, it is crucial to minimize the area of the protected implementation of mostly symmetric key cryptographic algorithms. It is shown that the decrease in number of shares has a direct impact in the area requirements. Achieving this goal motivates us to investigate the conditions to decompose the substitutions boxes, with focus being on Boolean permutations i.e. invertible S-boxes, of high algebraic degree into the ones of lower degree. The parities of power permutations help us to determine whether the higher degree permutations are decomposable into those power permutations or not.

In this chapter, we focus on the decomposition method 2.2.1, one of those to have an efficient TI-sharing described in [5]. With the aim of decomposition of high degree permutations into the lower ones, in [34], it is determined the conditions to obtain quadratic and cubic permutations over the finite fields $\mathbb{F}_{2^n}$ for values of n between 3 and 16 using Carlitz's Theorem. Then in [19] it is investigated the decomposition of permutations in $Sym(\mathbb{F}_{2^n})$ for $3 \leq n \leq 31$ into quadratic or cubic permutation using Stafford's Theorem, which is stated below, in the Section 3.1. Also, the decomposition process of permutation is reduced to a modular arithmetic problem as well as this work. In order to investigate when a permutation over a finite field can be decomposed into permutations of lower degree, we studied the parities of power permutations. In the Section 3.3, we provide many lemmas and corollaries that we presented our techniques leading us to find the cycle structures and parities of permutations. The cycle structure of power permutation was also studied in [2]. We give the results in a different point of view, which is much better in computational complexity than the previous ones.

3.1 Preliminaries

Let $\mathbb{F}_q$ be the finite field $GF(q)$ with $q = 2^n$ elements and $Sym(\mathbb{F}_q)$ denote its symmetric group. We find the values of n such that permutations in $Sym(\mathbb{F}_q)$ can be written as a composition of permutations of lower algebraic degree. For the definition of algebraic degree and for detailed study on symmetric groups, we refer the reader to the works [17] and [25], respectively.

A polynomial $f \in \mathbb{F}_q[x]$ is called a *permutation polynomial* of $\mathbb{F}_q$ if the mapping $f : \mathbb{F}_q \to \mathbb{F}_q$ given by $c \mapsto f(c)$ is a permutation, i.e f is 1-1 and onto. Given a permutation ψ in $Sym(\mathbb{F}_q)$, there exists the unique permutation polynomial representing ψ, which can be seen in the following lemma. We refer the reader [17], [28] and [33] for a rigorous information.

Lemma 3.1.1. *[28] For any function $\psi : \mathbb{F}_q \to \mathbb{F}_q$ there exists a unique polynomial $f \in \mathbb{F}_q[x]$ of degree at most $q-1$ such that the associated polynomial function $f : c \mapsto f(c)$ satisfies $\psi(c) = f(c)$ for all $c \in \mathbb{F}_q$.*

Proof. $f(x)$ can be written by the Lagrange and then by the Carlitz interpolation formulas as following:

$$f(x) = \sum_{c \in \mathbb{F}_q} \psi(c) \prod_{\substack{c_i \in \mathbb{F}_q \\ c_i \neq c}} \left(\frac{x - c_i}{c - c_i} \right) = \sum_{c \in \mathbb{F}_q} \psi(c)(1 - (x-c)^{q-1}).$$

For uniqueness, suppose that f and g are two different polynomials in $\mathbb{F}_q[x]$ of degrees at most $q-1$, satisfying $f(c) = g(c)$ for all $c \in \mathbb{F}_q$. Since $f \neq g$, it follows that their difference $f - g$ is a nonzero polynomial that vanishes at all q elements of $\mathbb{F}_q$. But $deg(f-g) \leq q-1$, so $f - g$ can have at most $q-1$ roots in $\mathbb{F}_q$, which is a contradiction. $\square$

Consequently, all permutations considered in this work are of degrees $\leq q-1$. We recall the following well-known theorem.

Theorem 3.1.2. *[28] The monomial x^k is a permutation polynomial of $\mathbb{F}_q$ if and only if $gcd(k, q-1) = 1$.*

Proof. (2) Since $0^n = 0$, the monomial x^n is onto if and only if the function

$$\begin{aligned} f : \mathbb{F}_q^* &\to \mathbb{F}_q^* \\ x &\mapsto x^n \end{aligned}$$

is onto. Let g be a primitive element of the cyclic group $\mathbb{F}_q^*$. Then the image of $\mathbb{F}_q^*$ under f is the cyclic subgroup generated by g^n, which equals $\mathbb{F}_q^*$ if and only if g^n is a primitive element. This is equivalent to the statement $gcd(n, q-1) = 1$. □

We will refer to permutations induced by monomials x^k as power permutations. The (algebraic) degree of a power permutation x^k is defined to be equal to $wt(k)$, where $wt(k)$ denotes the Hamming weight of the n-bit vector corresponding to the binary expansion of k in [17], or equivalently $2 - adic$ notation of the number k.

Any permutation can be represented as a composition of disjoint cycles. A cycle is a set of elements in a permutation which switch an element with one another. A cycle with 2 elements is called a *transposition*. Any permutation can be written as a product of such transpositions. There is no unique way to express a permutation using transpositions; however, the number of them is either always odd or always even, depending on the permutation. This number corresponds the parity or the sign of the permutation.

Recall that Euler's totient function $\phi(q-1)$ which counts the number of positive integers up to $q-1$ that are relatively prime to $q-1$.

3.1.3 Composition of permutations

The permutations $\tau_{a,b}$ defined by $x \mapsto ax + b$ for $a \in \mathbb{F}_q^*$ and $b \in \mathbb{F}_q$ are called affine permutations. The set $Aff(\mathbb{F}_q) = \{\tau_{a,b} \mid a \in \mathbb{F}_q^*, b \in \mathbb{F}_q\}$ is clearly closed under composition and inversion, hence it is a subgroup of $Sym(\mathbb{F}_q)$.

If there exists a permutation φ, such that φ and $Aff(\mathbb{F}_q)$ together generate $Sym(\mathbb{F}_q)$, then every element ψ of $Sym(\mathbb{F}_q)$ is of the form

$$\psi = \tau_1 \circ \varphi \circ \tau_2 \cdots \circ \varphi \circ \tau_k$$

for some affine permutations $\tau_1, \tau_2, \ldots, \tau_k$. If φ can be decomposed as

$$\varphi = Q_1 \circ Q_2 \circ \ldots Q_m,$$

where Q_i's are permutations of degree d, then ψ is a composition of permutations of degree d

$$\psi = (\tau_1 \circ Q_1) \circ Q_2 \circ \ldots Q_m \circ (\tau_2 \circ Q_1) \circ Q_2 \circ \ldots Q_m \circ \cdots \circ (\tau_{k-1} \circ Q_1) \circ Q_2 \circ \ldots (Q_m \circ \tau_k).$$

Thus, in order to show that every permutation in $Sym(\mathbb{F}_q)$ can be decomposed into permutations of degree d, it is sufficient to show

i. that there exists a permutation φ, which can be decomposed into permutations of degree d, and

ii. that φ generates $Sym(\mathbb{F}_q)$ together with $Aff(\mathbb{F}_q)$.

It was shown that every permutation could be written as a composition of affine permutations and the power permutation x^{q-2}, for $q = 5$ by Betti and for $q = 7$ by Dickson [22]. Later on, Carlitz proved that, for any q, every transposition (0α) can be generated by affine polynomials and the monomial x^{q-2}, where α denotes a fixed non-zero number in $\mathbb{F}_q$, by considering the polynomial:

$$g(x) = -\alpha^2\Big(\big((x-\alpha)^{q-2} + \frac{1}{\alpha}\big)^{q-2} - \alpha\Big)^{q-2}$$

where $g(0) = \alpha$, $g(\alpha) = 0$ and $g(\beta) = \beta$ for $\beta \neq 0, \beta \neq \alpha$. Explanations how the polynomial is constructed are given in [45]. Since every permutation can be written as a composition of transpositions, we have the following.

Theorem 3.1.4. *[18] The group $Sym(\mathbb{F}_q)$ is generated by the affine permutations and the power permutation x^{q-2}.*

In [34], the authors investigated when the power permutation $x^{q-2} = x^{-1}$ for $3 \leq n \leq 16$ can be decomposed into quadratic (or cubic) permutations and found those with a minimum decomposition length. The authors proved that every permutation in $Sym(\mathbb{F}_q)$ can be decomposed into quadratic permutations whenever n is not divisible by 4 and into cubic permutations when n is divisible by 4.

In this thesis, we extend this result for larger n due to the following generalization of Carlitz's result. In [44], Stafford generalized the previous result to all power maps with the following result. Namely, instead of using power permutation x^{q-2} (i.e. inverse map), it suffices to use any power permutation x^k under some conditions.

Theorem 3.1.5. *[44] Let $\mathbb{F}_q$ be the finite field where $q = 2^n$ and let $1 < k < q-2$ be an integer relatively prime to $q-1$. If k is not a power of 2 and the power permutation x^k is an odd permutation, then $Sym(\mathbb{F})$ is generated by the affine permutations $Aff(\mathbb{F}_q)$ and the power permutation x^k.*

If we can write a power permutation x^k, which satisfies Stafford's conditions, as a composition of quadratic (or cubic) permutations, then every permutation in $Sym(\mathbb{F}_q)$ can be written as a composition of quadratic (or cubic) permutations.

Our aim is to find low degree odd permutations x^k over a finite field $\mathbb{F}_{2^n}$ using Stafford's result.

3.2 Parity

Recall that a transposition is a cycle of length 2. A transposition is odd and so is any cycle of even length, as it can be written as a product of odd number of transpositions.

3.2.1 Analytic Approach

In this section, we show how to determine analytically the parity of a power permutation. Let α be a primitive element of the finite field $\mathbb{F}_q$. Then we can write

$$\mathbb{F}_q = \{0, \alpha, \alpha^2, \alpha^3, \ldots, \alpha^{q-2}, \alpha^{q-1} = 1\} = \{0\} \cup <\alpha>$$

Consider the power permutation x^k in $Sym(\mathbb{F}_q)$. That is,

$$x^k : \begin{pmatrix} 0 & \alpha^1 & \alpha^2 & \alpha^3 & \dots & \alpha^{q-2} & 1 \\ 0 & \alpha^k & \alpha^{2k} & \alpha^{3k} & \dots & \alpha^{k(q-2)} & 1 \end{pmatrix}$$

In order to determine whether or not the power permutation x^k is odd, it is sufficient to determine its cycle structure. Notice that the elements 0 and 1 are fixed points of x^k and we discard them. We begin writing x^k as a composition of disjoint cycles. The first cycle is of the form

$$[\alpha] = (\alpha^1, \alpha^k, \alpha^{k^2}, \dots, \alpha^{k^{N_1-1}})$$

where the length of cycle decomposition N_1 is the least positive integer such that

$$k^{N_1} \equiv 1 \ (mod\, q-1).$$

That is, N_1 is the order of k in the multiplicative group $\mathbb{Z}^*_{q-1}$. For the second cycle, if exists, we take the first α^j not included in this cycle and consider $(\alpha^j, \alpha^{kj}, \alpha^{k^2j}, \dots)$.

We repeat this procedure until we exhaust all elements.

Since a cycle is even if and only if its length is odd, we should count how many disjoint cycles there are of even length to determine if x^k is odd. Notice that this idea reduces the problem of checking the parity of x^k to a problem in modular arithmetic.

3.2.2 Special case

The idea in 3.2.1 reveals a direct theorem below:

Theorem 3.2.3. *Let x^k be a power permutation in $Sym(\mathbb{F}_q)$ of degree d. Assume that $q-1$ is an odd prime number and k is a primitive root of the multiplicative group $\mathbb{Z}^*_{q-1}$. Then every permutation in $Sym(\mathbb{F}_q)$ can be decomposed into permutations of degree d.*

Proof. Since k is a primitive root of $\mathbb{Z}^*_{q-1}$, the least positive integer i such that $k^i \equiv 1 \ (mod\, q-1)$ is $q-2$. Therefore, the cycle decomposition of x^k is

$$(\alpha^1, \alpha^k, \alpha^{k^2}, \dots, \alpha^{k^{q-3}})$$

Since the length of this cycle is even, the permutation x^k is odd and hence it generates $Sym(\mathbb{F}_q)$ together with $Aff(\mathbb{F}_q)$ by Theorem 3.1.5. Consequently, every permutation in $Sym(\mathbb{F}_q)$ can be decomposed into permutations of degree d.

□

The specific instances of this theorem can be seen for the permutations defined over the finite fields $\mathbb{F}_q$, where $q = 2^n$ and $n = 3, 5, 7, 13, 17, 19, \ldots$ (i.e. the exponents of some Mersenne primes) with $k = 3$.

3.3 Cycle Decomposition of Permutations

Assume that, using the exhaustive procedure described previously in the Section 3.2.1, the permutation x^k can be written in cycle decomposition notation, with disjoint cycles as

$$\underbrace{(\alpha^1, \alpha^k, \alpha^{k^2}, \ldots, \alpha^{k^{N_1-1}})}_{N_1\text{-many elements}} \ldots \underbrace{(\alpha^m, \alpha^{mk}, \ldots, \alpha^{mk^{N_m-1}})}_{N_m\text{-many elements}} \ldots$$

under the assumption that α^m is not included in the previous cycles.

Notation 3.3.1. *We shall denote the length of the cycle $[\alpha^m]$ by N_m. Equivalently, N_m is the minimum positive integer such that $mk^{N_m} \equiv 1 \mod (q-1)$. In addition, in the case m is a proper divisor of $q-1$, N_m is the order of k in the multiplicative group $(\mathbb{Z}_{q-1/m})^\times$. Throughout the chapter, the subscripts are from $\{1, 2, \cdots, q-2\}$ and unless otherwise indicated, 'divisor' is used instead of 'proper divisor' in these cases.*

3.3.2 On the length of the cycles

In this section, to show some relations between the lengths of certain cycles of some elements in $\mathbb{F}_q$ in cycle decomposition of the power permutation x^k, we have a useful lemma:

Lemma 3.3.3. *$N_{ms}|N_s$ for all m, s.*

Proof. Recall that N_s is the minimum positive integer satisfying $\alpha^{sk^{N_s}} = \alpha^s$. So, if we take the m^{th}-power of both sides, we get that

$$\alpha^{msk^{N_s}} = \alpha^{ms}$$

On the other hand, α^{ms} is in the cycle $(\alpha^{ms}, \alpha^{msk}, \alpha^{msk^2}, \ldots, \alpha^{msk^{N_{ms}-1}})$ and so

$$\alpha^{msk^{N_{ms}}} = \alpha^{ms}$$

where N_{ms} is the minimum positive integer satisfying this. Let $N_s = qN_{ms} + r$ for some integers q, r with $0 \leq r < N_{ms}$. Assume that $r \neq 0$. Then we have that

$$\begin{aligned}
\alpha^{msk^{N_s}} &= \alpha^{ms} \\
\alpha^{msk^{qN_{ms}+r}} &= \alpha^{ms} \\
\alpha^{msk^{qN_{ms}}k^r} &= \alpha^{ms} \\
\left(\alpha^{msk^{qN_{ms}}}\right)^{k^r} &= \alpha^{ms} \\
\left(\left(\cdots\left(\alpha^{msk^{N_{ms}}}\right)^{k^{N_{ms}}}\cdots\right)^{k^{N_{ms}}}\right)^{k^r} &= \alpha^{ms}
\end{aligned}$$

where we iteratively exponentiate q times to the power $k^{N_{ms}}$. However, since we have $\alpha^{msk^{N_{ms}}} = \alpha^{ms}$, one obtains that $\alpha^{msk^r} = \alpha^{ms}$, which contradicts the minimality of N_{ms}. Thus $r = 0$ and so $N_{ms}|N_s$. □

The following corollary follows immediately from Lemma 3.3.3.

Corollary 3.3.4. $N_m|N_1$ *for all* m.

The next lemma gives us a stronger result under some conditions.

Lemma 3.3.5. *Let* ρ *be a divisor of* $q-1$ *and suppose that* $\gcd\left(t, \frac{q-1}{\rho}\right) = 1$. *Then* $N_{\rho t} = N_\rho$.

Proof. Recall that, by the definition of N_ρ, we have that $\alpha^{\rho k^{N_\rho}} = \alpha^\rho$. It follows that $\alpha^{\rho(k^{N_\rho}-1)} = 1$ in $\mathbb{F}_q$. As the order of α in the multiplicative group $\mathbb{F}_q^*$ is $q-1$, we have that

$$q - 1|\rho(k^{N_\rho} - 1).$$

Moreover, N_ρ is the least positive integer satisfying this relation. Similarly, we know that $\alpha^{\rho t k^{N_{\rho t}}} = \alpha^{\rho t}$ and so $\alpha^{\rho t(k^{N_{\rho t}}-1)} = 1$ in $\mathbb{F}_q$. As before, we have that $q-1|\rho t(k^{N_{\rho t}}-1)$ and so $\frac{q-1}{\rho}|t(k^{N_{\rho t}}-1)$. Since $\gcd\left(t, \frac{q-1}{\rho}\right) = 1$, one can see

$$q-1|\rho(k^{N_{\rho t}}-1).$$

By the minimality of N_ρ, we have that $N_\rho|N_{\rho t}$. Otherwise, after applying the division algorithm to $N_{\rho t}$ and N_ρ as before, and we would obtain $0 < r < N_\rho$ such that $\alpha^{\rho k^r} = \alpha^\rho$ which is equivalent to $q-1|\rho(k^r-1)$. By Lemma 3.3.3, we also know that $N_{\rho t}|N_\rho$. Hence

$$N_{\rho t} = N_\rho.$$

□

Again we present an immediate corollary which follows from the Lemma 3.3.5.

Corollary 3.3.6. *If* $\gcd(t, q-1) = 1$, *then* $N_t = N_1$.

As one can deduce from the Lemma 3.3.5, the Euler totient function defined above in the Section 3.1, is needed in order to be able to count the number of a part of elements within the cycle of the same length.

3.3.7 The number of distinct cycles

Let ρ be a divisor of $q-1$. In this subsection, it will be proven that for a given divisor, the corresponding cycles are all distinct and the counting of elements appearing in the cycle decomposition of a permutation is completely done.

Notation 3.3.8. *Set* $K_\rho = \phi\left(\frac{q-1}{\rho}\right)$. *Let* W_ρ *denote the set of as* $W_\rho = \{t : \gcd\left(t, \frac{q-1}{\rho}\right) = 1$ *and* $1 \le t < \frac{q-1}{\rho}\}$. *Note that* $|W_\rho| = \phi\left(\frac{q-1}{\rho}\right) = K_\rho$. *We enumerate the elements of* W_ρ *as* $W_\rho = \{t_1, t_2, \ldots, t_{K_\rho}\}$. *Then, for a divisor* ρ *of* $q-1$, *we define the list* L_ρ *of cycles as the following list:*

$$[\alpha^{\rho t_1}] = (\alpha^{\rho t_1}, \alpha^{\rho t_1 k}, \alpha^{\rho t_1 k^2}, \ldots, \alpha^{\rho t_1 k^{N_\rho - 1}})$$
$$[\alpha^{\rho t_2}] = (\alpha^{\rho t_2}, \alpha^{\rho t_2 k}, \alpha^{\rho t_2 k^2}, \ldots, \alpha^{\rho t_2 k^{N_\rho - 1}})$$

$$\vdots \qquad\qquad (\star)$$
$$[\alpha^{\rho t_{K_\rho}}] = (\alpha^{\rho t_{K_\rho}}, \alpha^{\rho t_{K_\rho} k}, \alpha^{\rho t_{K_\rho} k^2}, \ldots, \alpha^{\rho t_{K_\rho} k^{N_\rho - 1}})$$

Observe that each of these cycles has length N_ρ, since $\gcd(t, \frac{q-1}{\rho}) = 1$ implies that the length of $[\alpha^{\rho t}]$ is the same as the length of $[\alpha^\rho]$, as stated in the Lemma 3.3.5.

Some of the cycles in ($\star$) may be identical. Let U_ρ denote the number of *distinct* cycles in this list. In the following lemma we determine U_ρ.

Lemma 3.3.9. *Let ρ be a divisor of $q-1$ and U_ρ denote the number of distinct cycles in the list ($\star$), which is the list determined by ρ as explained above. We have*

$$U_\rho = \frac{K_\rho}{N_\rho}$$

where K_ρ and N_ρ is defined in Notation 3.3.8 and 3.3.1, respectively.

Proof. Since $\gcd(k, q-1) = 1$ and $\gcd\left(t_i, \frac{q-1}{\rho}\right) = 1$, following from $\forall t_i \in W_\rho$, we have that $\gcd\left(t_i k, \frac{q-1}{\rho}\right) = 1$, which implies that $t_i k \mod (\frac{q-1}{\rho})$ is also in the set W_ρ. Hence, one can see that $\alpha^{\rho t_i}$ is counted in at least N_ρ different cycles in the list $\star$. As this holds for any $t \in W_\rho$, we conclude that $N_\rho | K_\rho$ and

$$U_\rho \leq \frac{K_\rho}{N_\rho}$$

We claim that $U_\rho \geq \frac{K_\rho}{N_\rho}$ for every divisor ρ of $q-1$. Assume to the contrary that, for some divisor ρ of $q-1$, we have that $U_\rho < \frac{K_\rho}{N_\rho}$. Then, since every element of $\mathbb{F}_q^*$ is contained in some cycle of this form for some divisor ρ of $q-1$, we would have

$$|\mathbb{F}_q^*| \leq \sum_{\rho | q-1} U_\rho N_\rho < \sum_{\rho | q-1} \frac{K_\rho}{N_\rho} N_\rho = \sum_{\rho | q-1} \phi\left(\frac{q-1}{\rho}\right) = q-1$$

which is a contradiction. Hence, $U_\rho = \frac{K_\rho}{N_\rho}$ for every divisor ρ of $q-1$.

□

Remark 3.3.10. *Note that it is possible to have two distinct divisors ρ_1 and ρ_2 of $q-1$ such that $N_{\rho_1} = N_{\rho_2}$. Therefore, U_{ρ_1} might be strictly less than the number of all cycles of length N_{ρ_1}.*

In order to make it more precise, the following toy example is given for a power permutation defined in a finite field $\mathbb{F}_{2^n}$ with a small value of n because of the cycles being easy to compute.

Example 3.3.11. Consider the power permutation x^5 over $\mathbb{F}_{2^6}$. So $q-1 = 63 = 3^2 7$. Let $\rho_1 = 1$, $\rho_2 = 3$, $\rho_3 = 7$, $\rho_4 = 9$ and $\rho_5 = 21$, the proper divisors of 63.

– For $\rho_1 = 1, W_1 = \{1, 2, 4, 5, 8, 10, 11, 13, 16, 17, 19, 20, 22, 23, 25, 26, 29, 31, 32, 34, 37, 38, 40, 41, 43, 44, 46, 47, 50, 52, 53, 55, 58, 59, 61, 62\}$, i.e. the numbers coprime to 63. The distinct cycles in L_1 by computing the cycles of α^{t_i} where $t_i \in W_1$ are:

  - $[\alpha] = (\alpha, \alpha^5, \alpha^{25}, \alpha^{62}, \alpha^{58}, \alpha^{38})$
  - $[\alpha^2] = (\alpha^2, \alpha^{10}, \alpha^{50}, \alpha^{61}, \alpha^{53}, \alpha^{13})$
  - $[\alpha^4] = (\alpha^4, \alpha^{20}, \alpha^{37}, \alpha^{59}, \alpha^{43}, \alpha^{26})$
  - $[\alpha^8] = (\alpha^8, \alpha^{40}, \alpha^{11}, \alpha^{55}, \alpha^{23}, \alpha^{52})$
  - $[\alpha^{16}] = (\alpha^{16}, \alpha^{17}, \alpha^{22}, \alpha^{47}, \alpha^{46}, \alpha^{41})$
  - $[\alpha^{19}] = (\alpha^{19}, \alpha^{32}, \alpha^{34}, \alpha^{44}, \alpha^{31}, \alpha^{29})$

  Note that it is not necessary to compute the cycle $[\alpha^5]$ separately since it is nothing but $[\alpha]$. Clearly it is seen that all elements α^{t_i} satisfying $t_i \in W_1$ are spanned above. 36 elements of $\mathbb{F}_{2^6} \setminus \{0, 1\}$ are located in a cycle. There are 6 different cycles, which confirms with Lemma 3.3.9 as the values of the total number of elements, $K_1 = \phi(63) = 36$, the number of elements in one cycle $N_1 = 6$ and the number of distinct cycles $U_1 = 6$ in the list L_1.

– For $\rho_2 = 3$, $W_3 = \{1, 2, 4, 5, 8, 10, 11, 13, 16, 17, 19, 20\}$. Check the cycles of α^{3t_i} where $t_i \in W_3$.

  - $[\alpha^3] = (\alpha^3, \alpha^{15}, \alpha^{12}, \alpha^{60}, \alpha^{48}, \alpha^{51})$
  - $[\alpha^6] = (\alpha^6, \alpha^{30}, \alpha^{24}, \alpha^{57}, \alpha^{33}, \alpha^{39})$

  Again all elements α^{t_i} satisfying $t_i \in W_3$ are spanned. Another 12 elements of $\mathbb{F}_{2^6} \setminus \{0, 1\}$ are located in a cycle.

We continue to apply the same procedure to the rest of divisors with less details.

- For $\rho_3 = 7$, $W_7 = \{1, 2, 4, 5, 7, 8\}$ and the only cycle is:
  - $[\alpha^7] = (\alpha^7, \alpha^{35}, \alpha^{49}, \alpha^{56}, \alpha^{28}, \alpha^{14})$
- For $\rho_4 = 9$, $W_9 = \{1, 2, 3, 4, 5, 6\}$ and there is again one cycle:
  - $[\alpha^9] = (\alpha^9, \alpha^{45}, \alpha^{36}, \alpha^{54}, \alpha^{18}, \alpha^{27})$
- For $\rho_5 = 21$, $W_{21} = \{1, 2\}$ and the cycle:
  - $[\alpha^{21}] = (\alpha^{21}, \alpha^{42})$

One can see that each element of $\mathbb{F}_{2^6}$ appears exactly once in the distinct cycles listed above. The values of K_ρ, N_ρ and U_ρ are given together with the divisors ρ, in the following table:

	$K_\rho = \phi\left(\frac{q-1}{\rho}\right)$	N_ρ, order of k in $(\mathbb{Z}_{q-1/\rho})^\times$	$U_\rho = \frac{K_\rho}{N_\rho}$
$\rho = 1$	36	6	6
$\rho = 3$	12	6	2
$\rho = 7$	6	6	1
$\rho = 9$	6	6	1
$\rho = 21$	2	2	1

Table 3.1: The K_ρ, N_ρ and U_ρ values for the given divisors ρ

The cycle structure of the power permutation x^5 over $\mathbb{F}_{2^6}$ can be represented as $[<6, 10>, <2, 1>]$, which means there are 10 different cycles of length 6 and there is 1 cycle of length 2. The cycles of the elements 0 and 1 are not listed in this notation. In brief, we can give a summary of lemmas and some corresponding examples specific to x^5 defined over $\mathbb{F}_{2^6}$. By the Lemma 3.3.3, $N_9 | N_3$ and $N_{21} | N_7$ etc, where $N_9 = 6$, $N_3 = N_7 = 6$, $N_{21} = 2$. By the Corollary 3.3.4, $N_\rho | N_1 = 6$ for all divisors ρ. By the Lemma 3.3.5, $N_{18} = N_9$ since $\gcd(2, 7) = 1$, i.e. $\gcd\left(t, \frac{q-1}{\rho}\right) = 1$. By the Corollary 3.3.6, $N_2 = N_1$ since $\gcd(2, 63) = 1$. Also as mentioned in the Remark 3.3.10, one can see that there are some cycles of same length for the elements placed in different lists, for example $N_3 = N_7$. By the Lemma 3.3.9, we cover all elements in $\mathbb{F}_{2^6}$.
From its cycle structure, one can say that x^5 defined over $\mathbb{F}_{2^6}$ has odd parity since there are odd many cycles of even length. Afterwards, we will decide the parity of a permutation without computing the cycle structure of it.

Corollary 3.3.12. *If N_1 is odd, then permutation is even.*

Proof. N_1 is odd implies N_m's are all odd, for any $1 \leq m \leq q-2$, since $N_m | N_1$ by Corollary 3.3.4. Hence regardless of their number of cycles, it forms a even permutation. $\square$

Therefore using Lemma 3.3.5 and the Lemma 3.3.9, we arrive at our main Theorem.

Theorem 3.3.13. *Let k be a positive integer less than $q-2$ and relatively prime to $q-1$. Let d be the algebraic degree of k, where x^k is a power permutation in $Sym(\mathbb{F}_q)$. For a divisor m of $q-1$, let N_m be the order of k in $(\mathbb{Z}_{\frac{q-1}{m}})^\times$, where $(\mathbb{Z}_{\frac{q-1}{m}})^\times$ is the multiplicative group consisting of invertible elements of $\mathbb{Z}_{\frac{q-1}{m}}$. Then, x^k is odd if and only if N_1 is even and $|S|$ is odd where $S := \{m \mid N_m$ is even$\}$.*

By the help of this theorem we can determine the parity of a given power permutation as well as its cycle structure. In the literature, the cycle structure of power permutation ψ_k was also given by Ahmad, in [2] as follows:

Theorem 3.3.14. *Let m be any positive integer. Then x^k has a cycle of length m if and only if $q-1$ has a divisor t such that k belongs to the exponent m modulo t. The exact number T_m of such cycles is*

$$T_m = \sum_{e \in C_m} \phi(e)$$

where $C_m = \{t : t|d_m$ and k belongs to m modulo t, where $d_m = gcd(k^m-1, q-1)\}$ and ϕ is Euler's totient function.

By using our method, in a different point of view than the one in [2], which is described in Section 3.3 and which will be described in the following algorithms in details, one can obtain the cycle structure of a power permutation with a better computational complexity. Moreover, our method give information also about the parity of the permutation faster.

3.4 Algorithms

In order to determine that a permutation is even or not, we provide the following algorithm:

```
Result: The parity of a power permutation $x^k$
Inputs: $k, q-1$
$N_1 \leftarrow ord_{\mathbb{Z}_{q-1}^{\times}}(k)$
if $N_1$ is odd then
    Print "The permutation is even"
    Stop
else
    while $\rho$ is a proper divisor of $q-1$ do
        $N_\rho \leftarrow ord_{(\mathbb{Z}_{q-1/\rho})^{\times}}(k)$
        if $N_\rho$ is even then
            $U_\rho \leftarrow \phi(\frac{q-1}{\rho})/N_\rho$
            if $U_\rho$ is odd then
                count ← count + 1
            end
        end
    end
end
```

Algorithm 1: The parity of a power permutation x^k

By this algorithm, we can determine that the permutation is odd if the count is odd, otherwise it is even.

In addition, in order to write also the cycle structure of a monomial using our method we provide the following algorithm:

```
Result: The cycle structure of a power permutation $x^k$
Inputs: $k, q-1$
List[a][2] ← 0                                  ▷ where a is the number of divisors
count ← 0
while $\rho$ is a divisor of $q-1$ do
    $N_\rho \leftarrow ord_{\mathbb{Z}_{q-1}^{\times}/\rho}(k)$
    $U_\rho \leftarrow \phi(\frac{q-1}{\rho})/N_\rho$
    for $i \leftarrow 0$ to $count$ do
        if $N_\rho$ in List[i][0] then
            List[i][1] ← List[i][1]+$U_\rho$
            break $i$
        else
            count ← count + 1
            List[count][0] ← $N_\rho$
            List[count][1] ← $U_\rho$
        end
    end
end
for $i \leftarrow 0$ to $count$ do
    for $j \leftarrow 0$ to 1 do
        Print List[i][j]
    end
end
```

Algorithm 2: The cycle structure of a power permutation x^k

3.4.1 Complexity Comparison

In this section, it is given the complexities of our method and the one of Ahmad [2] in detail.

Since $q = 2^n$, the number of digits of q, equally $\log(q)$ or $\log(q-1)$ when required, equals to n. First we need to find the divisors of the number $q-1$. This can be done by factorizing having the complexity $\mathcal{O}(exp((64/9)^{1/3}n^{1/3}(\ln(n))^{2/3}))$ with the Generalized Number Field Sieve. The number of divisors are bounded from above with $n^{1/3}$. Then, it will be computed the order of k in $\mathbb{Z}^{\times}_{q-1/\rho}$, having the complexity $\mathcal{O}(\sqrt{N_\rho}) < \mathcal{O}(2^{n/2})$ with Pollard's Rho algorithm. To calculate the Euler totient value of a number, we can use the factorization which is previously computed, therefore there is no additional complexity for these steps, and dividing it by N_ρ for each divisor ρ has relatively small complexity.

In total, both Algorithm 1 and 2 have the complexity:

$$\mathcal{O}(exp((64/9)^{1/3}n^{1/3}(\ln(n))^{2/3}))+\mathcal{O}(n^{1/3}2^{n/2}) \approx \mathcal{O}(exp((64/9)^{1/3}n^{1/3}(\ln(n))^{2/3})).$$

In Ahmad's method, there are $q-1$, i.e. 2^n-1, many choices in the beginning. The method firstly calculates the greatest common divisor of k^m-1 and $q-1$ which has complexity $\mathcal{O}(\log(n))$ and finds a divisor of that number with the same complexity as integer factorization which is $\mathcal{O}(exp((64/9)^{1/3}n^{1/3}(\ln(n))^{2/3}))$. Then finds the order of k in $\mathbb{Z}^{\times}_t$ which has $\mathcal{O}(2^{n/2})$ complexity and finally calculates Euler totient r times, where r is the number of elements of C_m with no additional complexity as stated above and the sum which has relatively small complexity. In overall it has the following complexity:

$$\mathcal{O}\left((2^n-1)(exp((64/9)^{1/3}n^{1/3}(\ln(n))^{2/3}) + \log(n) + 2^{n/2})\right)$$

which approximately equals to

$$\mathcal{O}((2^n-1)exp((64/9)^{1/3}n^{1/3}(\ln(n))^{2/3})).$$

Hence, Ahmad's method is (2^n-1) times more complex than ours.

3.5 Experimental Results

In this section, before stating our experimental result, it is given in [34] that the following lemma which helps us in omitting the search for quadratic power permutations in some finite fields.

Lemma 3.5.1. *No quadratic power permutations exist for $n=2^m$ in $\mathbb{F}_q$ with $q=2^n$.*

Proof. For a quadratic power permutation x^k, k should be of the form $k = 2^{j+i} + 2^j$ for some integers i, j with $i > 0$, i.e. the binary representation of quadratic k is $(0 \cdots 010 \cdots 010 \cdots 0)$ where 1's are in the $(j+i)^{th}$ and j^{th} positions from right. Clearly $k = 2^j(2^i + 1)$. By Theorem 3.1.2, x^k being a permutation is equivalent to $gcd(k, q-1) = 1$. So we check whether $gcd(2^j(2^i+1), 2^{2^m} - 1) = 1$. It is easily seen that $gcd(2^j(2^i+1), 2^{2^m}-1) = gcd(2^i+1, 2^{2^m}-1)$ and moreover, we have that

- If $gcd(2^i - 1, 2^{2^m} - 1) = 1$, then

$$gcd(2^i+1, 2^{2^m}-1) = \gcd(2^{2i}-1, 2^{2^m}-1) = 2^{gcd(2i,2^m)} - 1 \neq 1.$$

- If $gcd(2^i - 1, 2^{2^m} - 1) \neq 1$, then

$$\begin{aligned} gcd(2^i+1, 2^{2^m}-1) = \frac{\gcd(2^{2i}-1, 2^{2^m}-1)}{gcd(2^i-1, 2^{2^m}-1)} &= \frac{2^{gcd(2i,2^m)}-1}{2^{gcd(i,2^m)}-1} \\ &= 2^{gcd(i,2^m)} + 1 \neq 1. \end{aligned}$$

In both cases, since $gcd(k, q-1) \neq 1$, no quadratic power permutations exists in $\mathbb{F}_q$ with $q = 2^{2^m}$. □

In this thesis, we performed a search for quadratic and cubic power permutations for the values $3 \leq n \leq 141$ using the Algorithm 1 which follows from the Theorem 3.3.13, and also by the help of the Lemma 3.5.1.

We now state our experimental results.

Theorem 3.5.2. *Let n be an integer such that $3 \leq n \leq 141$.*

– *If n is not divisible by 4, then every permutation in $Sym(\mathbb{F}_{2^n})$ can be written as a composition of quadratic and affine permutations.*

– *If n is divisible by 4, then every permutation in $Sym(\mathbb{F}_{2^n})$ can be written as a composition of cubic and affine permutations.*

  - *Moreover, if n is a power of 2, then every permutation in $Sym(\mathbb{F}_{2^n})$ can be written as a composition of x^{13} and affine permutations.*
  - *$Sym(\mathbb{F}_{2^n})$ cannot be generated by the quadratic power permutations and the affine permutations.*

One can find these decomposition results to any permutation in $GF(2^n)$ in [34] for the values $3 \leq n \leq 16$, and in [19] for the values $3 \leq n \leq 31$.

We performed a search for quadratic and cubic power permutations for various values of n, using C and MAGMA [12]. Based on the computational evidences, we conjecture the following:

Conjecture 3.5.3.

– *For all integers $n \geq 1$, the power permutation x^3 is odd in $Sym(\mathbb{F}_{2^{2n+1}})$.*

– *For all integers $n \geq 1$, the power permutation x^5 is odd in $Sym(\mathbb{F}_{2^{4n+2}})$ and $Sym(\mathbb{F}_{2^{4n+3}})$.*

– *For all integers n which is a multiple of 4 and not a power of 2, all quadratic permutations of $\mathbb{F}_{2^n}$ are even.*

For some small values of n multiple of 4, the odd but cubic permutations are provided:

- For $n = 4$, there exists no quadratic power permutation and the cubic power permutation x^7 is odd, in total there are 4 cubic permutations with odd parity.
- For $n = 8$, there exists no quadratic power permutation and the cubic power permutation x^{13} is odd, in total there are 8 cubic permutations with odd parity.
- For $n = 12$, the all 12 quadratic power permutations are even and the cubic power permutation x^{11}, one of the 36, is odd.
- For $n = 16$, there exists no quadratic power permutation and the cubic power permutation x^7 is odd, in total there are 160 cubic permutations with odd parity.
- For $n = 20$, the all 40 quadratic power permutations are even and the cubic power permutation x^{13}, one of the 280, is odd.
- For $n = 24$, the only quadratic power permutation is x^{257}, which is even; and the cubic power permutation x^{11} is odd.
- For $n = 28$, the only quadratic power permutations are x^{17}, x^{257} and x^{4097}, which are even; and the cubic power permutation x^7 is odd. with odd parity.

We note that the permutations in the list above are not necessarily the unique ones. For example, for $n = 4$, the power permutation x^{13} is also odd and hence, generates $Sym(\mathbb{F}_{2^n})$ together with affine permutations.

CHAPTER 4

CONCLUSION

Throughout this thesis, we elaborated the process that led us to our main results/ conclusion.

The thesis contains four chapters. Below is a brief summary of each chapter and the contribution within each. Then we propose directions for future research.

We introduced the importance of cryptography in the modern life in the Introduction Chapter. We briefly mentioned the cryptographic systems, Side-Channel Analysis and countermeasures, and attempted to draw the path of reaching Threshold Implementation and the decomposition method. Then we stated our research questions.

We presented our answers across two chapters. In the second chapter, we recalled the properties of Threshold Implementation and gave various examples. To be able to give an answer to our first question, we examined the behavior of TI-sharing of S-boxes under inverse transformation and we showed that the answer of our first research question proves to be negative except for a subset of classes.

The third chapter was the most theoretical one. We presented several lemmas and corollaries to arrive at our main theorem. Finally, we apply our method for various finite fields and we obtained some remarkable experimental results. We gave an answer to our second question.

We left a conjecture 3.5.3 containing “3 different open questions” to be solved.

Under the assumption that the third item of the conjecture holds, “Can we find a power permutation over $\mathbb{F}_{2^n}$ such that it is cubic and odd for all n multiple of 4.”

In this dissertation, we focused on only the power permutations. We can suggest another future investigation: “Can we find a permutation, rather than power permutations, that is used to generate $Sym(\mathbb{F}_{2^n})$ for n multiple of 4?”

REFERENCES

[1] How the modern world depends on encryption, https://www.bbc.com/news/technology-24667834, accessed: 2020-01-05.

[2] S. Ahmad, Cycle structure of automorphisms of finite cyclic groups, Journal of Combinatorial Theory, 6(4), pp. 370 – 374, 1969, ISSN 0021-9800.

[3] M.-L. Akkar and C. Giraud, An implementation of des and aes, secure against some attacks, in Ç. K. Koç, D. Naccache, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems — CHES 2001*, pp. 309–318, Springer Berlin Heidelberg, Berlin, Heidelberg, 2001, ISBN 978-3-540-44709-2.

[4] T. Beyne and B. Bilgin, Uniform first-order threshold implementations, Cryptology ePrint Archive, Report 2016/715, 2016, https://eprint.iacr.org/2016/715.

[5] B. Bilgin, Threshold implementations: as countermeasure against higher-order differential power analysis, 2015.

[6] B. Bilgin, B. Gierlichs, S. Nikova, V. Nikov, and V. Rijmen, Higher-order threshold implementations, IACR Cryptology ePrint Archive, 2014, p. 751, 2014.

[7] B. Bilgin, B. Gierlichs, S. Nikova, V. Nikov, and V. Rijmen, A more efficient aes threshold implementation, in D. Pointcheval and D. Vergnaud, editors, *Progress in Cryptology - AFRICACRYPT 2014*, volume 8469 of *Lecture Notes in Computer Science*, pp. 267–284, Springer International Publishing, 2014, ISBN 978-3-319-06733-9.

[8] B. Bilgin, S. Nikova, V. Nikov, V. Rijmen, and G. Stütz, Threshold implementations of all 3x3 and 4x4 s-boxes, in *International Workshop on Cryptographic Hardware and Embedded Systems*, pp. 76–91, Springer, 2012.

[9] B. Bilgin, S. Nikova, V. Nikov, V. Rijmen, N. Tokareva, and V. Vitkup, Threshold implementations of small s-boxes, Cryptography and Communications, pp. 1–31, 2014, ISSN 1936-2447.

[10] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. B. Robshaw, Y. Seurin, and C. Vikkelsoe, Present: An ultra-lightweight block cipher, in P. Paillier and I. Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems - CHES 2007*, pp. 450–466, Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[11] J. W. Bos, C. Hubain, W. Michiels, C. Mune, E. S. Gonzalez, and P. Teuwen, White-box cryptography: Don't forget about grey-box attacks, Journal of Cryptology, pp. 1–49, 2017.

[12] W. Bosma, J. Cannon, and C. Playoust, The Magma algebra system. I. The user language, J. Symbolic Comput., 24(3-4), pp. 235–265, 1997, ISSN 0747-7171, computational algebra and number theory (London, 1993).

[13] C. Boura and A. Canteaut, On the influence of the algebraic degree of f^{-1} on the algebraic degree of $g \circ f$, IEEE Transactions on Information Theory, 59(1), pp. 691–702, 2012.

[14] D. Bozilov, B. Bilgin, and H. A. Sahin, A note on 5-bit quadratic permutations' classification, IACR Trans. Symmetric Cryptol., 2017(1), pp. 398–404, 2017.

[15] E. Brier, C. Clavier, and F. Olivier, Correlation power analysis with a leakage model, in M. Joye and J.-J. Quisquater, editors, *Cryptographic Hardware and Embedded Systems - CHES 2004*, pp. 16–29, Springer Berlin Heidelberg, Berlin, Heidelberg, 2004, ISBN 978-3-540-28632-5.

[16] C. D. Cannière, Analysis and design of symmetric encryption algorithms (analyse en ontwerp van symmetrische encryptie-algoritmen), 2007.

[17] C. Carlet, Vectorial boolean functions for cryptography, Boolean models and methods in mathematics, computer science, and engineering, 134, pp. 398–469, 2010.

[18] L. Carlitz, Permutations in a finite field, Proceedings of the American Mathematical Society, 4(4), p. 538, 1953.

[19] P. Çomak, S. Nikova, and V. Rijmen, On decomposition of permutations, in *3rd International Conference on Cryptography and Information Security in the Balkans*, 2018.

[20] S. Chari, C. S. Jutla, J. R. Rao, and P. Rohatgi, Towards sound approaches to counteract power-analysis attacks, in M. Wiener, editor, *Advances in Cryptology — CRYPTO' 99*, pp. 398–412, Springer Berlin Heidelberg, Berlin, Heidelberg, 1999, ISBN 978-3-540-48405-9.

[21] T. D. Cnudde, B. Bilgin, O. Reparaz, V. Nikov, and S. Nikova, Higher-order threshold implementation of the AES s-box, in *Smart Card Research and Advanced Applications - 14th International Conference, CARDIS 2015, Bochum, Germany, November 4-6, 2015. Revised Selected Papers*, pp. 259–272, 2015.

[22] L. E. Dickson, The analytic representation of substitutions on a power of a prime number of letters with a discussion of the linear group., The Annals of Mathematics, 11(1/6), pp. 65–120, 1896.

[23] J. D. Golić and C. Tymen, Multiplicative masking and power analysis of aes, in B. S. Kaliski, ç. K. Koç, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002*, pp. 198–212, Springer Berlin Heidelberg, Berlin, Heidelberg, 2003, ISBN 978-3-540-36400-9.

[24] L. Goubin and J. Patarin, Des and differential power analysis the “duplication” method, in Ç. K. Koç and C. Paar, editors, *Cryptographic Hardware and Embedded Systems*, pp. 158–172, Springer Berlin Heidelberg, Berlin, Heidelberg, 1999, ISBN 978-3-540-48059-4.

[25] I. N. Herstein, *Abstract algebra*, Prentice Hall, 1996.

[26] Y. Ishai, A. Sahai, and D. A. Wagner, Private circuits: Securing hardware against probing attacks, in *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, pp. 463–481, 2003.

[27] P. Kocher, J. Jaffe, and B. Jun, Differential power analysis, in M. Wiener, editor, *Advances in Cryptology — CRYPTO’ 99*, pp. 388–397, Springer Berlin Heidelberg, Berlin, Heidelberg, 1999, ISBN 978-3-540-48405-9.

[28] R. Lidl and H. Niederreiter, Finite fields: Encyclopedia of mathematics and its applications., Computers & Mathematics with Applications, 33(7), pp. 136–136, 1997.

[29] S. Mangard, E. Oswald, and T. Popp, *Power Analysis Attacks: Revealing the Secrets of Smart Cards (Advances in Information Security)*, Springer-Verlag, Berlin, Heidelberg, 2007, ISBN 0387308571.

[30] S. Mangard, T. Popp, and B. M. Gammel, Side-channel leakage of masked cmos gates, in *Proceedings of the 2005 International Conference on Topics in Cryptology*, CT-RSA’05, p. 351–365, Springer-Verlag, Berlin, Heidelberg, 2005, ISBN 3540243992.

[31] A. J. Menezes, S. A. Vanstone, and P. C. V. Oorschot, *Handbook of Applied Cryptography*, CRC Press, Inc., USA, 1st edition, 1996, ISBN 0849385237.

[32] A. Moradi, A. Poschmann, S. Ling, C. Paar, and H. Wang, Pushing the limits: A very compact and a threshold implementation of aes, in K. G. Paterson, editor, *Advances in Cryptology – EUROCRYPT 2011*, pp. 69–88, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[33] G. L. Mullen and D. Panario, *Handbook of finite fields*, CRC Press, 2013.

[34] S. Nikova, V. Nikov, and V. Rijmen, Decomposition of permutations in a finite field, Cryptography and Communications, 11(3), pp. 379–384, May 2019, ISSN 1936-2455.

[35] S. Nikova, C. Rechberger, and V. Rijmen, Threshold implementations against side-channel attacks and glitches, in *International Conference on Information and Communications Security*, pp. 529–545, Springer, 2006.

[36] C. Paar and J. Pelzl, *Understanding Cryptography - A Textbook for Students and Practitioners.*, Springer, 2010, ISBN 978-3-642-04100-6.

[37] A. Poschmann, A. Moradi, K. Khoo, C.-W. Lim, H. Wang, and S. Ling, Side-channel resistant crypto for less than 2,300 ge, Journal of Cryptology, 24, pp. 322–345, 2010.

[38] O. Reparaz, B. Bilgin, S. Nikova, B. Gierlichs, and I. Verbauwhede, Consolidating masking schemes, in R. Gennaro and M. Robshaw, editors, *Advances in Cryptology – CRYPTO 2015*, pp. 764–783, Springer Berlin Heidelberg, Berlin, Heidelberg, 2015, ISBN 978-3-662-47989-6.

[39] M. Rivain, E. Dottax, and E. Prouff, Block ciphers implementations provably secure against second order side channel analysis, in K. Nyberg, editor, *Fast Software Encryption*, pp. 127–143, Springer Berlin Heidelberg, Berlin, Heidelberg, 2008, ISBN 978-3-540-71039-4.

[40] M. Rivain and E. Prouff, Provably secure higher-order masking of aes, in S. Mangard and F.-X. Standaert, editors, *Cryptographic Hardware and Embedded Systems, CHES 2010*, pp. 413–427, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, ISBN 978-3-642-15031-9.

[41] K. Schramm and C. Paar, Higher order masking of the aes, in D. Pointcheval, editor, *Topics in Cryptology – CT-RSA 2006*, pp. 208–225, Springer Berlin Heidelberg, Berlin, Heidelberg, 2006, ISBN 978-3-540-32648-9.

[42] A. Shamir, How to share a secret, Commun. ACM, 22(11), p. 612–613, November 1979, ISSN 0001-0782.

[43] C. E. Shannon, Communication theory of secrecy systems, Bell system technical journal, 28(4), pp. 656–715, 1949.

[44] R. M. Stafford, Groups of permutation polynomials over finite fields, Finite Fields and Their Applications, 4(4), pp. 450–452, 1998.

[45] M. E. Zieve, On a theorem of carlitz, Journal of Group Theory, 17(4), pp. 667–669, 2014.

CURRICULUM VITAE

PERSONAL INFORMATION

Surname, Name: Çomak, Pınar
Nationality: Turkish
Date and Place of Birth: 09.02.1987, Ankara
E-mail address: pnarcomak@gmail.com
Phone: +90 555 624 86 99

EDUCATION

Degree	Institution	Year of Graduation
M.S.	Middle East Technical University, Cryptography Department	2013
B.S.	Middle East Technical University, Department of Mathematics	2010
High School	Nermin Mehmet Çekiç Anadolu Lisesi	2005

PROFESSIONAL EXPERIENCE

Year	Place	Enrollment
Oct 2011- Sep 2019	Middle East Technical University	Research Assistant
May 2013- July 2013	Sogang University - South Korea	Visiting Researcher
Oct 2017- June 2018	KU Leuven University - Belgium	Visiting Researcher
Sep 2019-current	Ericsson Research - TR	Security Researcher

PUBLICATIONS

International Journal Publications

- Pınar Çomak, Ferruh Özbudak, “On the Parity of Power Permutations", (to appear).
- Pınar Çomak, Svetla Nikova, and Vincent Rijmen, “On Decomposition of Permutations", (to appear in Springer’s CCIS series).
- Pınar Çomak, Jon-Lark Kim, Ferruh Özbudak, “New cubic self-dual codes of length 54, 60 and 66". Appl. Algebra Engrg. Comm. Comput. 29(4), 303–312 (2018). `https://doi.org/10.1007/s00200-017-0343-x`

International Conference Publications

- Pınar Çomak, Svetla Nikova, and Vincent Rijmen, “On Decomposition of Permutations", Balkan CryptSec, September 20-21, 2018, Iasi, Romania.
- Pınar Çomak, Jon-Lark Kim, Ferruh Özbudak, “Some new quasi-cyclic self-dual codes", 15th International Workshop on Algebraic and Combinatorial Coding Theory (ACCT), June 18-24, 2016, Albena, Bulgaria.