Power Analysis of Stream Ciphers Based on Feedback Shift Registers

by

© Abdulah A. Zadeh, M.Sc., B.Eng.

A thesis submitted to the

School of Graduate Studies

in partial fulfilment of the

requirements for the degree of

Doctor of Philosophy

Faculty of Engineering and Applied Science

Memorial University of Newfoundland

May 2014

St. John's Newfoundland Canada

Acknowledgements

I would like to sincerely thank my supervisor Dr. Howard M. Heys.

Also, I should thank my friends and other professors: Cheng Wang, for many good times we had in lab and working towards our PhD degrees, Dennis Peters, Cheng Li, Lihong Zhang, Paul Gillard, Theodore Norvell, Amr Youssef, Gabriel Lau, and Nolan White.

Abstract

In recent days, many cryptographic devices, such as smart-cards and cell phones, are widely accessible to many people. However, wide access to cryptographic devices makes them vulnerable to side channel analysis (SCA) attack. As such, there is a high demand for research in the field of side channel analysis. Although SCA attacks have been extensively applied to block ciphers, only a limited amount of research is available on the effectiveness of side channel analysis on stream ciphers. In this dissertation, we study SCA attacks on stream ciphers and develop some cryptanalysis methods for applying the attacks effectively on practical realization of stream ciphers.

The proposed power analysis attacks were first theoretically applied to stream ciphers with a linear feedback shift register (LFSR) and nonlinear filtering function, a structure referred to as a filter generator. Since typical stream ciphers include multiple LFSRs and/or nonlinear feedback shift registers (NLFSRs), we first consider the extension of the typical power analysis attack to stream ciphers with multiple LFSRs and a nonlinear combining function, known as a combination generator. Then, the attack is extended to stream ciphers based on nonlinear feedback shift registers (NLFSRs) and stream ciphers with multiple NLFSRs and LFSRs.

In most papers related to applying side channel analysis attacks to stream ciphers, the authors ignore the effect of noise and inaccurate measurements. This limits the applicability of their methods for real applications. This dissertation has developed side channel analysis attacks on feedback shift register (FSR) based stream ciphers with consideration of inaccurate measurement effects. At first, we have developed the attack for stream ciphers based on an individual LFSR and/or NLFSR, while the power measurements are inaccurate and they do not exactly match the theoretical values. Later, considering inaccurate measurements, we have developed power analysis of stream ciphers with multiple LFSRs and NLFSRs.

Finally, we consider combining SCA with some classical attacks on stream ciphers based on mathematical and statistical approaches to recover key or state bits of the stream ciphers. Hence, we have extended the correlation attack, fast correlation attack and algebraic attack, which are mathematical (or classical) attacks, such that they are applicable with side channel analysis. The proposed methods are validated through implementation on a practical cryptographic algorithm, the Grain stream cipher.

The practical investigations in this dissertation are done using simulated ASIC circuits. To simulate the behavior of ASIC circuits, we have implemented them using Cadence Virtuoso Spectre Circuit Simulator version 5.10.41. All the circuits including LFSR, NLFSR and Grain, are prototyped in TSMC 180 nm standard cell CMOS technology. The simulated power consumptions are used to investigate the practical application of the proposed attacks.

This dissertation shows power analysis is a powerful technique to attack stream ciphers and recover state bits and/or the key of the stream ciphers. Furthermore, combining classical methods and measured power data can significantly reduce the complexity of an attack of a stream cipher and countermeasure methods should be considered in hardware implementation of stream ciphers, to make them resistant to side channel analysis.

Contents

Chapter 1

Introduction

Increasing the application of complex systems increases the necessity of security and data obscurity. Many companies require their information and data to be housed securely and transmitted privately on the Internet. *Cryptology* is the science of hiding data [1]. Hence, many new ranges of application in cryptology have been opened in recent years. The demands of transferring these data over different networks such as Internet, mobile systems and wireless networks underscores the necessity of studying modern cryptology.

Nowadays, cryptology's objective is not summarized merely in confidentiality or encryption or hiding data from unauthorized persons. It also includes authenticity (incoming data originating from authorized source), data integrity (data is not changed or maliciously modified before it is received) and digital signature (verification of the authenticity of the message) [2].

Cryptology is divided into two dependent fields, *cryptography* and *cryptanalysis* [2, 3]. The aim of cryptography is to design secure systems and/or protocols of transferring data. It includes offering algorithms and methods of obscuring data called cryptographic algorithms or ciphers. However, cryptanalysis is the study of methods for obtaining the meaning of encrypted information without access to the secret information. Typically, in cryptanalysis, scientists try to find the key. Applying a cryptanalysis method is called an *attack*.

Cryptographic algorithms are divided into two classes, *public key* and *private key*. In private key (also known as symmetric key), both parties involved share the same key. This key is kept secret. Anybody who obtains it, can decrypt the ciphertext and recover the original data (plaintext). Thus, a secure channel should be established to transfer the private key to the other party. In public key cryptography, each party has its own unique key and access to a shared public key. Since, each party uses its own key to encrypt or decrypt, a secure channel is not necessary [4].

In comparison public key cryptography requires more computations. Hence, the main application of public key cryptography is key exchanging. In order to avoid high computational process of public key cryptography, applications usually also make use of private key cryptography. Public key cryptography are used to generate a shared key for both parties of the communications. Then, private key cryptography, with lower computational process is used to encrypt and decrypt the data.

Private key cryptography is also divided into two categories, *block ciphers* and *stream ciphers*. In block ciphers, the input is divided into blocks of fixed size and encryption and/or decryption is performed on the whole block. In a stream cipher, input is a continuous stream of bits. At each step (or clock), a random bit is generated by the stream cipher. A bit wise Xor of the input and the generated bit makes the output of the encryption or decryption.

Among different kinds of cryptographic methods, stream ciphers typically consume less power and occupy smaller area on the chip. Due to this fact, stream ciphers are attractive in many low power designs. Notable examples of low power circuits are used in RFID-tags, smart-cards, and wireless sensor networks. Because of these characteristics of stream ciphers, the improvement of security in stream ciphers has been intensively researched in recent years.

A prominent example for a stream cipher is the A5/1 cipher, which is part of the GSM mobile phone standard and is used for voice encryption [5]. Another notable application of stream ciphers is the E0 cipher used in bluetooth data transferring [6]. Although, stream ciphers are sometimes also used for encrypting Internet traffic, especially the stream cipher RC4, in practice, block ciphers are used more than stream ciphers for Internet communications [7].

Stream ciphers are basically random bit generator state machines. The output of the state machine is called the *keystream*. At each clock, encryption or decryption is achieved by adding (Xoring) a bit from a keystream to a plaintext bit. Hence, both parties of the communication should be synchronized. The process is shown in Figure 1.1.

In order to generate random bit values, linear feedback shift registers (LFSRs) are widely used in stream ciphers. LFSRs as a building block of many stream ciphers, can generate a good pseudo-random sequence. The other advantage of using LFSRs in stream ciphers is straightforward implementation in hardware. One general architecture to generate keystream in stream ciphers is called a combination generator and uses multiple LFSRs or nonlinear feedback shift registers (NLFSRs) combining their output bit values to generate the stream cipher's keystream. Notable examples

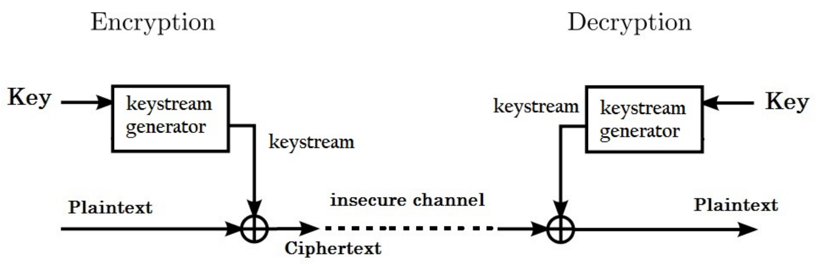


Figure 1.1: General architecture of stream ciphers

of this architecture are A5/1, A5/2 [5], E0 [6], Grain [8] and Trivium [9].

Cryptanalysis is divided into two classes. The first class is referred to as *mathematical attack* (or classical cryptanalysis) and is based on a combination of algebraic, statistical and numerical techniques. Well-known examples of mathematical attacks on stream ciphers are the algebraic attack, correlation attack, fast correlation attack and distinguishing attack. The second class is called *side channel analysis* (SCA) which is based on gaining information from physical implementation of cryptosystem, such as power consumption, timing information and electromagnetic leaks.

For an SCA attack, the attacker should have physical access to the hardware implementation of the cryptographic circuit and the ability to measure the physical characteristics of the hardware. Wide access to hardware systems such as wireless sensor nodes, RFID-tags and smart-cards make many applications vulnerable to side channel analysis attack. Also, advancements in technology provides many measuring devices with high accuracy at low cost for small labs and personal purpose. The physical access to the target devices and accurate measuring devices increase the necessity of studying side channel analysis.

In this dissertation, we will investigate the application of side channel analysis, in particular power analysis, to stream ciphers constructed using feedback shift registers.

Chapter 2

Background

This chapter presents some basic concepts of stream ciphers and side channel analysis attacks. As well, it introduces the notion of the linear feedback shift register (LFSR) and the nonlinear feedback shift register (NLFSR) as basic components of many stream ciphers. Grain, as a test bench for our cryptanalysis techniques, is described and some preliminaries of classical attacks applied to stream ciphers and side channel analysis are presented.

2.1 Linear and Nonlinear Feedback Shift Registers

Linear feedback shift registers are widely used as a basic component of a keystream generator in many proposed stream ciphers [2], due to their simple hardware structure and the good pseudo-random properties of the generated sequence. A right-shifting LFSR of size L consists of L bits and the output of each step (i.e., as the result of a triggering clock edge in synchronous sequential digital logic hardware) is the rightmost bit. The bit values are shifted to the right at each step and a new bit is injected into the leftmost bit of the register after being produced as a linear combination of bits currently stored in the register. It is well known that if the feedback is chosen as a *primitive polynomial*, the LFSR makes a sequence of bits with a maximal period of $2^L - 1$ [10, 11, 12]. Using the feedback coefficients, we can give a compact description of an LFSR through its feedback polynomial.

The value of the i-th register bit at time t is represented as $s_t(i)$. The content of the register at time t is $S_t = (s_t(L-1), s_t(L-2), s_t(L-3), ..., s_t(0))$. This is called the *state* of the LFSR at time t. The first L bit values of the LFSR, $S_0 = (s_0(L-1), s_0(L-2), s_0(L-3), ..., s_0(0))$, are loaded into the register at the start, and is denoted as the *initial state* of the LFSR.

LFSRs are used as a building block in many applications. Although they are designed for hardware, they can be efficiently implemented in software. In software, a finite field corresponding with the word size can be used efficiently to implement LFSRs. For instance, a 64-bit LFSR can be implemented easily on a 64 bit processor.

Since the register bit values and resulting outputs are generated from the linear combination of the previous L bit values, the register value of the LFSR at a particular point in time can easily be derived from any previous or following sequence of L consecutive bits of output. Hence, in order to increase the security, some stream ciphers use nonlinear feedback shift registers, NLFSRs. In an NLFSR, the feedback is a nonlinear combination of bit values. Although the nonlinear feedback makes the analysis of the output stream more difficult, it also reduces the output sequence period below the maximal value of $2^L - 1$ [11, 12].

The general structure of an LFSR or NLFSR is shown in Figure 2.1, where each square represents a register bit or D flip-flop.

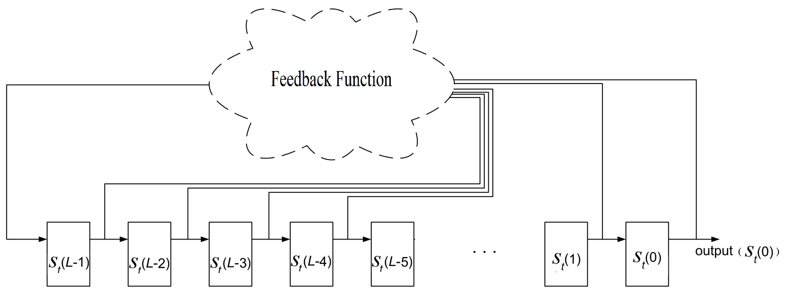


Figure 2.1: Overall architecture of LFSR or NLFSR

2.2 Grain

In 2004, the European Network of Excellence in Cryptology, ECRYPT, launched a call for stream cipher proposals named eSTREAM [13]. The candidate stream ciphers were submitted in May 2005. The candidates were divided into software oriented and hardware oriented stream ciphers.

Grain is a light-weight stream cipher, first proposed by M. Hell, T. Johansson and W. Meier to eSTREAM. The original Grain [14] (now referred to as Grain version 0 or Grain-v0) uses an 80-bit key and a 64-bit *initialization vector* (IV). The IV is a publicly known value and used along with the secret key, to fill the internal state or register bits of the stream cipher. Grain has 160 bits of internal state including an 80 bit LFSR and an 80 bit NLFSR. The generated keystream bit at each clock pulse is a nonlinear combination of some LFSR and NLFSR bits. A slightly modified version (with small changes to the output function and the nonlinear feedback function), referred to as Grain version 1 or Grain-v1 [8] has been selected for the hardware portfolio by the eSTREAM project. In addition to Grain-v0 and Grain-v1, a version of Grain with 128 bit key proposed in [15] is called Grain-128. It includes a 128

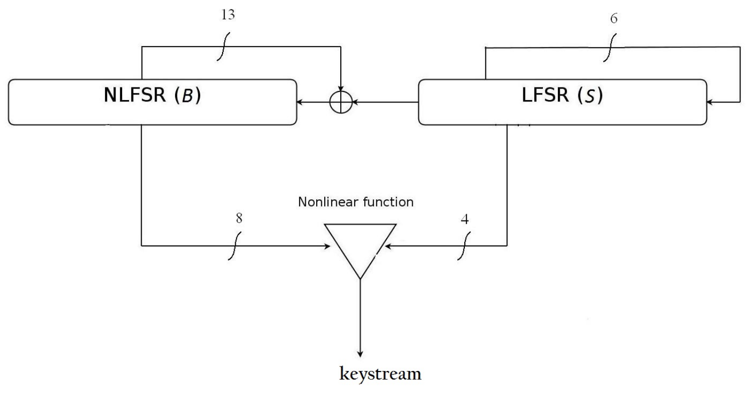


Figure 2.2: Architecture of Grain stream cipher

bit LFSR and a 128 bit NLFSR and nonlinear combination function to generate keystream bits. In this dissertation, we only study Grain-v0 and Grain-v1, however the proposed methods are applicable to Grain-128. The overall architecture of Grain-v1 is shown in Figure 2.2. Let S_t and B_t denote the 80-bit LFSR and NLFSR states, respectively, and $s_t(i)$ and $b_t(i)$, $0 \leq i < 80$, represents the value of bit i of S_t and B_t at time t.

The primitive polynomial of the LFSR for both Grain-v0 and Grain-v1 is

$$x^{80} + x^{67} + x^{57} + x^{42} + x^{29} + x^{18} + 1 = 0 \tag{2.1}$$

and the update function or feedback of the LFSR is

$$s_t(80) = s_t(62) \oplus s_t(51) \oplus s_t(38) \oplus s_t(23) \oplus s_t(13) \oplus s_t(0) \tag{2.2}$$

where $\oplus$ represents Xor operation. The expression of the feedback function for the

NLFSR of Grain-v0 is given by

$$\begin{aligned}
b_t(80) \quad = \quad & s_t(0) \oplus b_t(62) \oplus b_t(60) \oplus b_t(52) \oplus b_t(45) \oplus b_t(37) \oplus b_t(33) \oplus b_t(28) \\
& \oplus b_t(21) \oplus b_t(15) \oplus b_t(9) \oplus b_t(0) \oplus b_t(60) \cdot b_t(63) \oplus b_t(37) \\
& \cdot b_t(33) \oplus b_t(9) \cdot b_t(15) \oplus b_t(45) \cdot b_t(52) \cdot b_t(60) \oplus b_t(33) \cdot b_t(28) \\
& \cdot b_t(21) \oplus b_t(9) \cdot b_t(28) \cdot b_t(45) \cdot b_t(63) \oplus b_t(60) \cdot b_t(52) \cdot b_t(37) \cdot b_t(33) \\
& \oplus b_t(63) \cdot b_t(60) \cdot b_t(21) \cdot b_t(15) \oplus b_t(63) \cdot b_t(60) \cdot b_t(52) \cdot b_t(45) \cdot b_t(37) \\
& \oplus b_t(9) \cdot b_t(15) \cdot b_t(21) \cdot b_t(28) \cdot b_t(33) \oplus b_t(21) \cdot b_t(28) \cdot b_t(33) \cdot b_t(37) \\
& \cdot b_t(45) \cdot b_t(52). \qquad (2.3)
\end{aligned}$$

Note that the generation of $b_t(80)$ involves a bit from the LFSR in addition to the NLFSR feedback. The keystream output bit of Grain-v0 at time t, denoted as z_t, is derived from the current LFSR and NFSR states bits as follows:

$$\begin{aligned}
z_t \quad = \quad & b_t(0) \oplus s_t(25) \oplus b_t(63) \oplus s_t(64) \cdot s_t(3) \oplus s_t(64) \cdot s_t(46) \\
& \oplus s_t(46) \cdot s_t(25) \cdot s_t(3) \oplus s_t(64) \cdot s_t(46) \cdot s_t(3) \oplus b_t(63) \\
& \cdot s_t(46) \cdot s_t(3) \oplus b_t(63) \cdot s_t(64) \cdot s_t(46) \qquad (2.4)
\end{aligned}$$

Due to the weak design of Grain-v0, it was cryptanalyzed in [16, 17, 18]. Subsequently, in Grain-v1 the feedback function of the NLFSR and output function changed to

$$\begin{aligned}
b_t(80) \;=\;& s_t(0) \oplus b_t(62) \oplus b_t(60) \oplus b_t(52) \oplus b_t(45) \oplus b_t(37) \oplus b_t(33) \oplus b_t(28) \\
& \oplus b_t(21) \oplus b_t(14) \oplus b_t(9) \oplus b_t(0) \oplus b_t(60) \cdot b_t(63) \oplus b_t(37) \\
& \cdot b_t(33) \oplus b_t(9) \cdot b_t(15) \oplus b_t(45) \cdot b_t(52) \cdot b_t(60) \oplus b_t(33) \cdot b_t(28) \\
& \cdot b_t(21) \oplus b_t(9) \cdot b_t(28) \cdot b_t(45) \cdot b_t(63) \oplus b_t(60) \cdot b_t(52) \cdot b_t(37) \cdot b_t(33) \\
& \oplus b_t(63) \cdot b_t(60) \cdot b_t(21) \cdot b_t(15) \oplus b_t(63) \cdot b_t(60) \cdot b_t(52) \cdot b_t(45) \cdot b_t(37) \\
& \oplus b_t(9) \cdot b_t(15) \cdot b_t(21) \cdot b_t(28) \cdot b_t(33) \oplus b_t(21) \cdot b_t(28) \cdot b_t(33) \cdot b_t(37) \\
& \cdot b_t(45) \cdot b_t(52) \qquad (2.5)
\end{aligned}$$

and

$$\begin{aligned}
z_t \;=\;& b_t(1) \oplus b_t(2) \oplus b_t(4) \oplus b_t(10) \oplus b_t(31) \oplus b_t(43) \oplus b_t(56) \\
& s_t(25) \oplus b_t(63) \oplus s_t(64) \cdot s_t(3) \oplus s_t(64) \cdot s_t(46) \oplus s_t(46) \\
& \cdot s_t(25) \cdot s_t(3) \oplus s_t(64) \cdot s_t(46) \cdot s_t(3) \oplus b_t(63) \cdot s_t(46) \\
& \cdot s_t(3) \oplus b_t(63) \cdot s_t(64) \cdot s_t(46). \qquad (2.6)
\end{aligned}$$

The LFSR feedback remained unchanged.

Before any keystream is generated, the cipher must be initialized with a key and an IV. Let the bits of the key, K, be denoted k_i, $0 \le i < 80$, and the bits of the IV be denoted IV_i, $0 \le i < 64$. The initialization of the key is done as follows:

$$\begin{aligned}
b_0(i) &= k_i, \quad 0 \le i < 80 \\
s_0(i) &= IV_i, \quad 0 \le i < 64 \\
s_0(i) &= 1, \quad 64 \le i < 80
\end{aligned}$$

The cipher is clocked 160 times without producing any keystream. The output function is fed back and Xored with the input of both the LFSR and NLFSR.

The most successful documented attack on Grain is reported in [17]. It is applicable on Grain-v0. The attackers use second order fast correlation attack to calculate the LFSR state bits and using a simple technique they obtain the NLFSR state bits. The complexity of the proposed attack is 2^{43} operations and requires 2^{38} known keystream bits. As described above, due to the changes in the output function and the NLFSR feedback, this attack is not applicable to Grain-v1.

Another proposed attack on Grain is the time-memory trade-off attack. In [19], a time/memory/data trade off attack on stream ciphers has been analyzed and, using this approach, it is shown in [20] that for 160 state bits of Grain-v1, as an example of the trade offs, an attack can be mounted with a preprocessing complexity of 2^{103}, a time complexity of 2^{78} and the required memory and keystream data of 2^{64} and 2^{57}. Further, in [20], a guess-and-determine method is used so that the complexities are improved to 2^{71} for time complexity and required memory, $2^{106.5}$ for preprocessing complexity, and $2^{53.5}$ for required keystream. Since the key size for Grain-v1 is 80 bits, the total time complexity (considering both preprocessing time and runtime) of the proposed time-memory trade off attacks on Grain-v1 is worse than exhaustive key search. Currently the most efficient known attack on Grain-v1 and Grain-128 are still exhaustive key search [21].

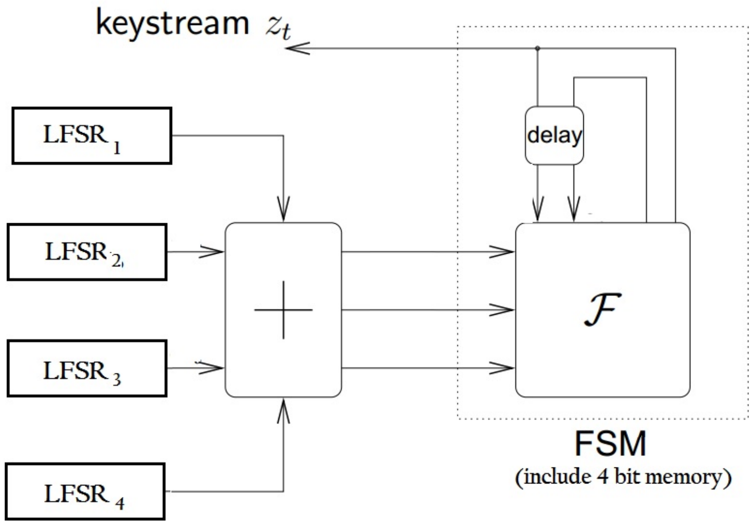


Figure 2.3: Architecture of E0 stream cipher

2.3 E0

E0 is another type of stream cipher. It is used in Bluetooth for wireless communication [6]. E0 has four LFSRs and four bit registers as memory. Figure 2.3 illustrates the cipher. The four-bit memory, c_t, causes the output of the cipher to depend on the current and the former state of the LFSRs. The lengths of the LFSRs are $L_1 = 25$, $L_2 = 31$, $L_3 = 33$ and $L_4 = 39$. The key size is $L_1 + L_2 + L_3 + L_4 = 128$. At each step the LFSRs are clocked once. The output of the LFSRs and the current values of the memories are combined to make the keystream (using a nonlinear function F). Then new value of memory is updated using current value of the memory and the summation of the four LFSR outputs.

Using an algebraic attack on E0, generates $2^{54.51}$ monomials. The number of required known keystream bits for algebraic attack is 2^{23} [22]. However the best cryptanalysis result belongs to a conditional correlation attack. In [23], it has been

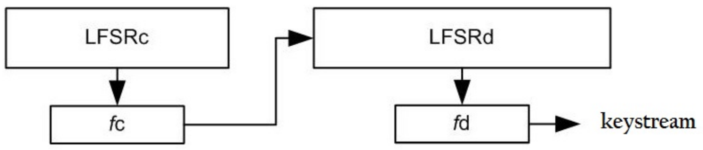


Figure 2.4: General architecture of LILI-128

shown that knowing the first 24 bits of $2^{23.5}$ frames, we can break E0 with the complexity of 2^{38}.

2.4 LILI-128

LILI-128 [24] consists of two LFSRs ($LFSR_c$ and $LFSR_d$). $LFSRc$ is 39 bits in length and controls the clock of $LFSR_d$ which is 89 bits in length. The bit values of c_{12} and c_{20} in $LFSR_c$ are passed through a function with two bits output, to determine whether $LFSR_d$ should be clocked once, twice, thrice or four times to produce keystream bits. The number of clocks, f_c, for $LFSR_d$, is calculated by

$$f_c = 2 \times c_{20} + c_{12} + 1. \tag{2.7}$$

The designers of LILI-128 publicized all the structure of the clock control subsystem and structure of the data generation subsystem. In Figure 2.4, the general structure of LILI-128 has been shown. LILI-128 was broken using Matlab software [25], on a personal laptop, given 2^{12} bits in about 1.7 hours by reconstructing its nonlinear filter function [26].

2.5 Side Channel Analysis

The low complexity of stream ciphers allows a straightforward approach to implementation in comparison to block ciphers. However, their straightforward design makes them vulnerable to side channel analysis attacks [27]. In reality, cryptographic algorithms are implemented in software or hardware on a physical device. Regardless of the robustness of resistance of a cipher to mathematical attacks, any implementation of a cipher can lead to new vulnerabilities called side channel analysis attacks. Side channel analysis has been an active area since 1996, when Paul Kocher published his paper on using timing information to attack the RSA, DSS and Diffie-Hellman public key cryptography algorithms [28].

In these attacks, a number of physical measurements of the cryptographic unit are made, for example power consumption, computing time or EMF radiation. These measurements are made over a large number of encryptions and then, using statistical techniques, the secret key embedded inside the cryptographic core is uncovered. These attacks work because there is a correlation between the physical measurements of consumed power taken at different points during the computation and the internal state of the processing device, which is itself related to the secret key. For example, in a smart-card when data is loaded from a memory, the memory bus has to carry the value of the data, taking an amount of power that depends on the data value [29, 30]. Since, the load instruction always happens at the same point within the computation, one can produce correlations between various runs of the application, eventually giving away the secret key of the smart-card.

Typically, in cryptanalysis, attackers try to find the key, or in the case of stream ciphers, the state bits. Many different methods have been used to cryptanalyze cryptographic algorithms. Using side channel information, the most well known attacks on cryptographic hardware are timing attack, template attack, power analysis and electromagnetic leakage attack.

2.5.1 Timing Attack

Timing attacks enable an attacker to extract secret information in a security system by observing the time it takes the system to respond to various queries or perform the cryptographic algorithm. The notable example of timing attack is timing attack of ECC [31] and RSA [28, 32]. The main operation in ECC is the double-and-add algorithm. The double-and-add algorithm is a series of point addition and point doubling over the curve. Point addition and point doubling include a series of multiplications, squaring, additions and divisions (or inversions) over the finite field. Execution of point doubling takes less time than point addition. For example, the ECC core implemented in [33] executes a point addition in 103 μs and a point doubling in 76 μs over $GF(P)$. Measuring the execution time of ECC, the attacker can guess the number of executed point addition and point doubling and calculate the key [31, 34].

2.5.2 Template Attack

A template attack is a strong probabilistic method for side channel analysis attack. It works by building up a set of templates for an intermediate value using a large number of acquired traces, where a trace is a recording of side channel information such as power consumption of the device being attacked while it is executing an algorithm. The classification stage then matches traces to a particular template using a probability distribution. The correct key value should be returned with a higher probability than the incorrect values. The computationally intensive and time consuming template building stage need only be completed once for a particular device. The same templates can then be used to mount multiple attacks on identical devices [35, 36].

2.5.3 Power Analysis Attack

A power analysis attack is a type of side channel analysis attack which assumes that the use of different keys implies differences in the power consumption. In this dissertation, we focus on side channel analysis attack based on gaining information from consumed power of the circuit. However, an electromagnetic leakage attack, is very similar in nature to power analysis attacks. In an electromagnetic leakage attack, the attacker measures the electromagnetic radiation of the chip to infer the internal data of the registers.

CMOS (Complementary Metal Oxide Semiconductor) is the dominant technology for ASIC (Application Specific Integrated Circuit) purposes. Minimal power consumption at steady state conditions determines the success of the technology in many present day consumer electronics. Unfortunately, the power consumption of this technology has a dependency on the data. This makes the implemented cryptographic algorithm in CMOS technology vulnerable to the side channel analysis attack based on power consumption.

As described in [37, 38, 39, 40], the major power consumption of transistors in a

CMOS circuit is dynamic power consumption. Dynamic power dissipation happens every time the state of a transistor changes (i.e. switches from zero to one or one to zero) causing the charge or discharge of the load capacitance. At the gate level, when the output of a gate changes, the state of its transistors changes. In other words, changing the output of a logic gate causes power dissipation in the circuit. In sequential circuits, the state of the circuit changes at the clock edges. Hence, the main dynamic power dissipation in sequential circuits happens at the triggering edges of the clock.

In power analysis, the attackers use the dynamic power consumption of the circuit to guess the number of changing gates and state bits of the circuit. There are two major methods that consider dynamic power dissipation of the circuit to recover the state bits or secret key of the cryptographic circuits, *simple power analysis* (SPA) and *differential power analysis* (DPA). While SPA directly uses the power consumption measurements to identify relevant power fluctuations related to cipher data, DPA uses statistical analysis and error correction techniques to extract information correlated to the state bits of the circuit. In the following sections, we review these two methods.

2.5.3.1 Simple Power Analysis

Previously proposed simple power analysis cryptanalysis of stream ciphers suggest using the dynamic power consumption measurements at the triggering edge of the clock (which we shall assume is the rising edge) to analyze the stream cipher. In the following, we will review the proposed analysis in [41] which is applicable to filter generator stream ciphers based on one LFSR and a nonlinear filtering function (Figure 2.5). Practical stream ciphers with this structure include Crypt-1 [42] and

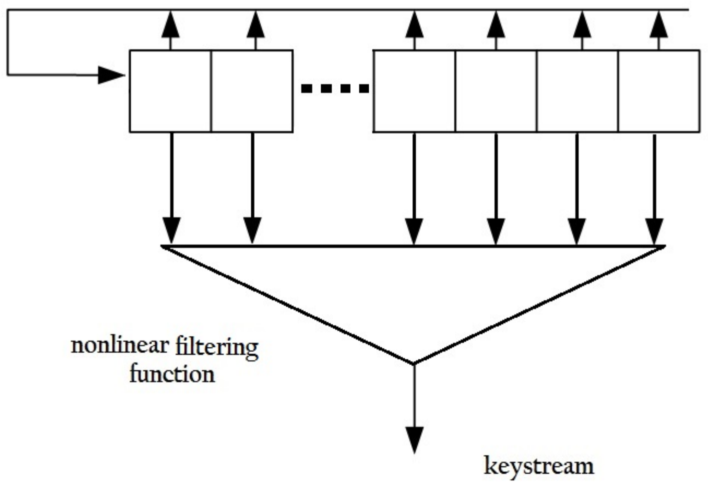


Figure 2.5: Architecture of filter generator stream cipher

Toyocrypt [43]. In such ciphers, the cipher key is typically used to initialize the bits of the LFSR. It should be noted that the attack of [41] is an idealized attack, assuming perfect mapping between power consumption information and cipher data.

During each clock cycle, assume each bit value in the LFSR is shifted to the right and the leftmost bit of the LFSR is updated with a linear combination of current register bit values (the feedback function in Figure 2.1). Changing the value of each bit in the register is due to change in gate outputs and transistor states and causes dynamic power consumption. At clock cycle t, the current state is represented as S_t and the state for the next clock cycle is given as S_{t+1}. The *Hamming distance* between S_t and S_{t-1} is given as HD_t where HD_t is calculated from

$$HD_t = \sum_{i=0}^{L-1} (s_t(i) \oplus s_{t-1}(i)), \tag{2.8}$$

where $s_t(i)$ represents the value of bit i of S_t with $s_t(0)$ being the rightmost bit of the

LFSR, $s_t(L-1)$ being the leftmost bit, and $\oplus$ representing Xor.

According to the Hamming distance power model used in the analysis [41], the dynamic power consumption of the cipher at clock cycle t is proportional to HD_t. Between two successive clock cycles, the difference between the Hamming distances must be one of three values: $HD_{t+1} - HD_t \in \{-1, 0, +1\}$. Defining the *theoretical power difference* to be PD_t given by

$$PD_t = HD_{t+1} - HD_t, \tag{2.9}$$

it can be seen that PD_t is proportional to the difference of the measured dynamic power consumption at two consecutive clock cycles at times t and $t+1$, which is an analog variable in watts and referred to as MPD_t. Simply, $PD_t \propto MPD_t$.

Substituting equation (2.8) into (2.9) results in (after simplifications)

$$PD_t = [s_{t+1}(L-1) \oplus s_t(L-1)] - [s_t(0) \oplus s_{t-1}(0)], \tag{2.10}$$

where the new bit value for state $t+1$, $s_{t+1}(L-1)$, will be the new value of bit $L-1$ based on the values of S_t. Considering operations over $GF(2)$, we can write

$$|PD_t| = s_t(L) \oplus s_{t-1}(L) \oplus s_t(0) \oplus s_{t-1}(0), \tag{2.11}$$

where we now denote $s_{t+1}(L-1)$ as $s_t(L)$ and $s_t(L-1)$ as $s_{t-1}(L)$ [1]. If the measured dynamic power consumption of the LFSR at clock cycle t is equal to the measured dynamic power consumption at clock cycle $t+1$ (that is, $MPD_t \approx 0$), then we can conclude $PD_t = 0$ and write $s_t(L) \oplus s_{t-1}(L) \oplus s_t(0) \oplus s_{t-1}(0) = 0$ and, if the measured

[1] In general, we can write $s_{t+j}(i) = s_t(i+j)$ with $s_t(i+j)$ representing the $(i+j)$-th bit following bit $s_t(0)$ in the LFSR/NLFSR sequence.

dynamic power consumption at time t and $t+1$ are not equal (that is, $MPD_t \neq 0$), we can conclude $PD_t \neq 0$ and write $s_t(L) \oplus s_{t-1}(L) \oplus s_t(0) \oplus s_{t-1}(0) = 1$.

It is known that, for any t, the bit values of S_t can be written as a linear function of the initial register state S_0 bits, that is, bits $\{s_0(i)\}$, where $0 \leq i < L$. Hence, for a stream cipher constructed as a nonlinear filter generator using one LFSR and a nonlinear filtering function [2], analyzing L power difference values, it is straightforward to find the initial L bit values of the LFSR and thereby determine the complete keystream sequence [41]. For this purpose, we can collect enough power samples to derive L power difference values and write L equations similar to equation (2.11), relating S_t through the linear expressions of the LFSR to the bits of S_0. Then we have a linear system of equations with L unknown variables and L equations, which is easily solved to determine the initial state of the LFSR, S_0, effectively finding the cipher key which is used to initialize S_0 in a typical stream cipher. Equivalently, finding the L bit values of the LFSR at any time t is sufficient to have broken the cipher, as all subsequent keystream bits are easily determined.

It is important to note that the described SPA method assumes that the analysis is capable of exactly determining theoretical power difference values (such that $|PD| \in \{1, 0\}$) from real power consumption measurements (which are analog values in units of watts). The theoretical PD values are then used directly to determine the register bit values. In practice, this is somewhat challenging and methods to overcome this challenge are discussed later in this dissertation.

2.5.3.2 Differential Power Analysis

Another approach to recover the state bits of a cryptosystem is called differential power analysis. It was proposed at first in 1999 [44]. Extensive research on DPA attacks shows its effectiveness even if the recorded power traces are inaccurate. DPA is more applicable to block ciphers and little research has investigated the application of DPA on stream ciphers. Notable reports of applying DPA on stream ciphers are [42], [45] and [46]. In [45], a theoretical DPA attack on A5/5 and E0 is offered. In [46], a known IV attack to Grain is proposed and [42] has offered a DPA attack of LFSR based stream ciphers, such as Crypto-1. In the proposed DPA attacks, it is necessary for the attacker to be able to perform encryption with different initialization vectors. That is, the cipher needs to be resynchronized many times. This limits the applicability of DPA in stream ciphers.

A precondition for a differential power analysis attack is that adversary knows the plaintext and the ciphertext to obtain the key. Let's assume an internal state bit of the cipher, is computed by $F(p_i, K)$, where F is a Boolean function, p_i is the i-th plaintext and K is the key. In DPA we divide the key to subkeys, k, and guess an initial value for each subkey. Then we check whether our guess was right or wrong. To check the correctness of the guess, for random input plaintext, p_i, we compute $F(p_i, k)$ (for the guessed value of the subkey) and we divide the results into two parts, S_0 and S_1. S_0 is the consumed power corresponds of set of computed values where $F(p_i, k) = 0$ and S_1 is for $F(p_i, k) = 1$. For all p_i, the difference between the mean values of two sets can be calculated

$$\delta_k = \sum_{i \in S_1} \frac{t_i}{|S_1|} - \sum_{i \in S_0} \frac{t_i}{|S_0|} \tag{2.12}$$

where t_i is the measured value of the cipher power for p_i. Every wrong guessed bit of subkey, k, reduces the absolute mean value ($|\delta_k|$) while the correct bit guess increases $|\delta_k|$, in the above equation. To find the correct subkey, we have to test all possible values for k and the greatest mean value of (2.12) shows the right guess of the subkey. Increasing the number of plaintexts helps the attacker to obtain more accurate results and decreases the effect of noise or inaccurate measurements. For example, in AES, F is defined as a function of 8 bits of subkey and using the above process we can recover the eight bits of AES at one time [47, 48].

2.6 Classical Attack of Stream Ciphers

The proposed classic mathematical attacks of stream ciphers are classified as known-plaintext attacks. In this section, we briefly review the basic concepts of the time-memory trade off attack, the algebraic attack, the correlation attack and the fast correlation attack.

2.6.1 Time-Memory Trade off Attack

The time-memory trade off attack is a known plaintext attack proposed in [49] and includes two phases. In the precomputation step, the attacker explores the general structure of the stream cipher and makes a table which consists of m bits of input and n bits of output. In other words, the attacker lists all possible values of m bits as input and calculates their corresponding outputs. In the second phase, the attacker, divides his output keystream to n-bit blocks and finds the corresponding m bit inputs in the table.

To make the attack more efficient, the attacker should make the table as big as possible and use some proposed techniques to sort them (in the precomputation phase). Increasing the table size causes increase in the required memory while reducing the required time to attack. Efficient implementation techniques have been offered in several papers, such as [19, 50], to make this attack practical for stream ciphers. A practical analysis of time-memory trade off attack of stream ciphers is proposed in [20] on Grain. It has been shown that recovering the state bits of Grain-v1 is possible with the memory and time complexity of 2^{71} and $2^{53.5}$ known keystream bits. The precomputation step needs $2^{106.5}$ steps.

2.6.2 Algebraic Attack

The algebraic attack is a powerful tool to cryptanalyze many stream ciphers previously believed very secure. The main idea behind this method is finding and solving a system of multivariate polynomial equations over a finite field. In other words, the algebraic attack reduces the cryptanalysis of the cipher into the problem of finding and solving a system of polynomial equations.

Solving a system of nonlinear polynomial equations over a finite field in general is an NP hard problem. But not all instances of NP hard problems are hard to solve. It might be possible to express a cipher in such a way that it is easier to solve than in exponential time. Solving such a system is called an algebraic attack. These attacks are motivated by the fact that the equation systems derived from the cipher are both sparse and overdefined [51]. The typical approach in an algebraic attack is converting the system of nonlinear equations to a system of linear equations. This process is called linearization. It causes a significant increase in the number of unknown variables. In Appendix A, we have provided a brief review to the proposed algebraic attacks called the relinearization and XL methods.

The applicability of the algebraic attack investigated in many stream ciphers such as HFE [52], Toyocrypt [53], Sfinks [1] [54], LILI-128 [55] and E0 [56]. Application of the algebraic attack is not limited to stream ciphers. Some research has been done to extend it to block ciphers and even public key cryptosystems as well. For example, in [57], the applicability of algebraic attack to *Advanced Encryption Standard* (AES) is studied. An example of applying algebraic attack to a public key cryptosystem is [58] in which they use algebraic attack to break MQQ public key cryptosystem.

2.6.3 Correlation and Fast Correlation Attack

Another powerful method of cryptanalyzing stream ciphers constructed of multiple FSRs is called the correlation attack. It is based on a divide-and-conquer approach. It has been shown that, for many LFSR based stream ciphers, there exists a measure of correlation between the keystream sequence and the outputs of the LFSRs, making it possible to determine the initial state of each of the LFSRs, independently [59]. Further it is possible to define higher order correlation attacks [60]. If there is a correlation between the keystream sequence and the addition of some LFSR state bits over the LFSR, still the attacker can use correlation attack. This is called a second order correlation attack.

The fast correlation attack is a more developed correlation attack, proposed for LFSR based stream ciphers. In the fast correlation attack, the attacker considers the

[1] an LFSR based stream cipher

linear relation between LFSR bits. Using the linear relation between LFSR bits, the attacker can reduce the timing complexity of the correlation attack. In Chapters 6 and 7, we have provided a more detailed review for correlation and fast correlation attacks.

2.7 Summary

In this chapter, we have reviewed the preliminaries for stream ciphers and side channel cryptanalysis. The main target of side channel cryptanalysis in our research is feedback shift register (FSR) based stream ciphers. We have studied the LFSR and NLFSR as a main building block of modern stream ciphers and Grain as an example of that. Also, we have studied the proposed simple power analysis for an LFSR based stream cipher, which has limited applicability on recent stream ciphers but is an important foundation for the work in this thesis. In the next chapters, we will discuss the development of the simple power analysis and propose new techniques which are applicable to practical implementations of modern stream ciphers.

Chapter 3

Applicability of Simple Power Analysis to Stream Ciphers Constructed Using Multiple LFSRs

As described in Section 2.5 the applicability of a simple power analysis, SPA, on stream ciphers has been identified in [41]. The proposed method is applicable to stream ciphers with just one linear feedback shift register. Since a number of modern stream ciphers use more than one LFSR, the direct methodology in [41] has limited applicability. In this chapter, we propose a method based on simple power analysis to attack stream ciphers with multiple LFSRs such as E0 [6]. Further, we consider the applicability of the attack to irregular clocking stream ciphers by examining LILI-128 [24]. It should be noted the proposed approach in this chapter is applicable in ideal environment, where the measured power of the circuit can be mapped to the theoretical state values of the circuit. In other words, there is no inaccuracy between power

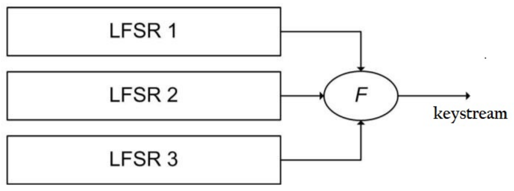


Figure 3.1: A stream cipher keystream generator with three LFSR

measurements and theoretical power differences. This work was initially presented in [61].

3.1 Extension of Simple Power Analysis to Ciphers with Multiple LFSRs

Consider now stream ciphers constructed from multiple LFSRs and a nonlinear combining function, referred to as a combination generator. An early example of such stream ciphers is the *Geffe* generator, which is constructed with three LFSRs and a nonlinear combining function [62]. We now consider the novel extension of the attack in [41] to such ciphers. A system with three LFSRs is illustrated in Figure 3.1, where F represents a nonlinear combining function.

As described before, in classical simple power analysis it is assumed the power consumption of the circuit at the rising edge of the clock is for D-flip flops. The power consumption of the other components has been ignored at the rising edge of the clock.

For simplicity in the discussion, let us assume a stream cipher with two LFSRs, $LFSR_S$ and $LFSR_R$, and bit values $s_t(i)$ and $r_t(j)$ where $0 \leq i < N$ and $0 \leq j < M$

where N and M are the sizes of the LFSRs [1]. The overall power difference of two LFSRs, $PD_t = HD_{t+1} - HD_t$, at each clock can now be from $\{-2, -1, 0, +1, +2\}$. Since each LFSR could have a power difference of -1, 0 or $+1$, if the power difference for both LFSRs is the same and equal to -1 or $+1$, then the overall PD_t is -2 or $+2$, respectively.

Although values of $PD_t = +2$ or -2 indicate that both LFSRs must have nonzero power differences, other values of overall PD_t will not get us any useful information about the individual LFSRs. For example, if the overall $PD_t = 0$, we cannot conclude whether both LFSR power differences are equal to zero or the power difference for one LFSR is equal to -1 and for the other one is equal to $+1$. Also, if the overall $PD_t = +1$ or $PD_t = -1$, we cannot distinguish for which LFSR the power difference is zero and for which LFSR the power difference is nonzero. However, for each clock cycle where overall $PD_t = +2$ or $PD_t = -2$, based on equation (2.11), we can conclude:

$$\begin{aligned} s_{t-1}(0) \oplus s_t(0) \oplus s_{t-1}(M) \oplus s_t(M) &= 1 \\ r_{t-1}(0) \oplus r_t(0) \oplus r_{t-1}(N) \oplus r_t(N) &= 1 \end{aligned} \tag{3.1}$$

where $s_t(i)$ and $r_t(i)$ represent the i-th bits of LFSR states at clock cycle t.

To break the stream cipher, we need to determine the $M + N$ bits of the LFSRs at a particular point in time. Hence, we require enough power difference values with $PD_t = +2$ or $PD_t = -2$ to obtain linear equations using (3.1) to solve for $M + N$ unknown variables. The minimum number of power difference values to set up the

[1]This method can be used for stream ciphers constructed more than two LFSRs. We will use it for E0 stream cipher, has four LFSRs

$M + N$ equations is M (if $M > N$) or N (if $N > M$). However, the minimum is unlikely to be achieved since usable power difference values must satisfy $PD_t = +2$ or $PD_t = -2$.

When we measure the consumed power of the circuit we should observe roughly five levels of power difference. The largest negative level should be assigned to $PD_t = -2$ and the largest positive level should be assigned to $PD_t = +2$.

The probability of a particular overall $PD_t = +2$ or $PD_t = -2$ is equal to $\frac{1}{8}$, since this occurs when the individual shift registers both have power differences of $+1$ or -1, each of which occurs with a probability of $\frac{1}{4}$. Hence, on average, we require $8 \times \max\{M, N\}$ power difference values to derive $M + N$ equations. Letting $T = \max\{M, N\}$, given n power difference values, it can be shown that the probability that there are enough usable power differences to form T equations is given by

$$Q_T(n) = \sum_{i=T}^{n} \binom{n}{i} (\frac{7}{8})^{n-i} (\frac{1}{8})^{i}. \tag{3.2}$$

Hence, for $T = 80$ and $n = 800$ power difference values, the probability that 80 equations can be derived is $Q_{80}(800) = .9877$. Assuming that all equations derived from the power differences are linearly independent, we can solve the system for the $M + N$ initial state bits of the two LFSRs by using two systems of equations. The systems of equations are linear and can be solved using appropriate mathematical computation tools such as Sage [63] or algebraic algorithms such as Gaussian elimination. However, the equations derived from the power difference values and the feedbacks are not necessarily all linearly independent. In fact, for an L-bit LFSR, given k randomly generated linear equations of the LFSR initial state bit values, from

[64] the probability that all k equations are linearly independent is

$$P_L(k) = \frac{\prod_{i=0}^{L-1}(2^L - 2^i)}{k! \times \binom{2^L-1}{k}} \tag{3.3}$$

for $k \leq L$.

If $k = L$, then $P_L(k)$ gives the probability that L randomly selected equations is enough to solve for the LFSR initial state bits. For example, for $k = L = 80$, $P_{80}(80) = .289$, implying that with 80 equations there is a 28.9% chance of being able to solve uniquely for the 80 state bits of the LFSR. Hence, in general, to ensure that we have a high probability of solving for $M + N$ bits when attacking the cipher based on two LFSRs, we should obtain somewhat more than $\max\{M, N\}$ equations from the power differences.

Consider now, the estimate of a lower bound on the probability, given k randomly generated linear equations with $k > L$, of being able to fully solve the system. $P_L(i, k)$ is defined as the representation of the probability which, given k bits randomly selected from a sequence generated by an L-bit LFSR, it is possible to generate a set of i linearly independent equations. From [64], for $k \leq L$, $P_L(k, k)$ can be generated:

$$P_L(k, k) = \frac{\prod_{i=0}^{L-1}\left(2^L - 2^i\right)}{k! \times \binom{2^L-1}{k}} \tag{3.4}$$

We are interested in situations where $k > L$ and $i = L$. Although we will not compute the probability directly in this case, we will derive a lower bound on $P_L(L, k)$ for $k > L$.

Consider a set of $k-1$ linear equations, $k > L$, formed from bits randomly selected from the sequence of an L-bit LFSR with the unknown variables being the initial L bits of the LFSR. Assume within the set of $k - 1$ linear equations, there is a subset of $i-1$ equations, $i \leq L$, that are linearly independent. Following the arguments presented in [64], the probability of randomly selecting a k-th linear equation that is linearly independent of all equations in the subset so that a subset of i linearly independent equations is formed is given by:

$$\Gamma_L(i,k) = \frac{\left(2^L - 2^{i-1}\right)}{(2^L - k - 2)} \tag{3.5}$$

The denominator represents the total number of equations left from which the k-th equation is drawn and the numerator represents the number of equations left which are linearly independent of the equations in the subset. The denominator and numerator are formed by considering that there are a total of 2^{L-1} linear equations of L variables and there are $2^{i-1}-1$ equations that are not linearly independent of the subset of $i-1$ equations. We are specifically interested in cases where $k \ll 2^{L-1}$ and, hence, since $i \leq L$, it is easy to see that $\Gamma_L(i,k) \geq .5$ and $\Gamma_L(i,k) \cong .5$ occurs for $i = L$.

In order to calculate the lower bound on $P_L(L,k)$ for $k > L$, we can use

$$P_L(L,k) \geq max\{P_L(j,j) \times \beta_L(j,k)\}(0 < j \leq L) \tag{3.6}$$

where $P_L(j,j)$ is given by (3.4) and $\beta_L(j,k)$ is the probability of adding $L-j$ more linear equations to the set of linearly independent equations given $k-j$ more randomly selected equations. Since $\Gamma_L(i,k) \geq .5$, we can compute a lower bound on β_L using the binomial distribution:

$$\beta_L(j,k) \geq \sum_{i=L-j}^{k-j} \binom{k-j}{i} \times 5^{k-j} \tag{3.7}$$

Using two equations (3.7) and (3.6), we can estimate the lower bound on $P_L(L,k)$ for $k > L$.

For example, if $L = 80$, it can be shown that obtaining 120 random equations will give a probability of over 99.99% of being able to solve for the 80 unknowns. Hence, for the cipher based on two LFSRs, if $\max\{M, N\} = 80$ bits, then, from equation (3.2), 1200 power difference values will give a 98.99% probability of obtaining 120 equations, which according to the equations (3.7) and (3.6), will give a probability of 99.99% of being solvable for the LFSR initial state bit values. Hence, for ciphers based on two LFSRs of sizes 80 bits and less, 1200 power samples will give a very high probability of being able to successfully apply SPA.

3.2 Application of the Attack to the E0 Stream Cipher

The E0 stream cipher [6] is a well-known stream cipher, used in Bluetooth which is used in short range, high speed communications such as mobile cell phones, PCs, and computer accessories. It is based on four LFSRs ($LFSR_a$, $LFSR_b$, $LFSR_c$, $LFSR_d$) with lengths of 25, 31, 33 and 39 bits [6]. In addition to four LFSRs, four bit registers save the state of the cipher as part of the nonlinear combining function. Hence, the equations used in the simple power analysis should be expanded to these four register bits. The output bit is a combination derived from the current bit values of LFSRs and the former state or register values.

Since at each clock, four LFSRs and four register bits could be changed, the overall PD_t can be from $\{\pm 8, \pm 7, \pm 6, \pm 5, \pm 4, \pm 3, \pm 2, \pm 1, 0\}$. The useful power differences are $PD_t = +8$ and $PD_t = -8$. When $PD_t = +8$ or $PD_t = -8$ we can conclude:

$$\begin{aligned}
a_{t-1}(0) \oplus a_t(0) \oplus a_{t-1}(25) \oplus a_t(25) &= 1 \\
b_{t-1}(0) \oplus b_t(0) \oplus b_{t-1}(31) \oplus b_t(31) &= 1 \\
c_{t-1}(0) \oplus c_t(0) \oplus c_{t-1}(33) \oplus c_t(33) &= 1 \qquad (3.8) \\
d_{t-1}(0) \oplus d_t(0) \oplus d_{t-1}(39) \oplus d_t(39) &= 1
\end{aligned}$$

where a, b, c and d represent LFSR state bits. In addition, four equations can be written for the four register bits of the combiner.

Noting that the largest LFSR size is 39 bits based on the discussions in former section, using 60 useful power difference values (i.e., $PD_t = +8$ or $PD_t = -8$), with the probability of more than 99.2%, we can find 39 linearly independent equations to solve $LFSR_d$. To find 60 useful power differences, equations (3.2) should be modified for E0 to (3.9).

$$Q_T(n) = \sum_{i=T}^{n} \binom{n}{i} (\frac{254}{256})^{n-i} (\frac{2}{256})^{i} \qquad (3.9)$$

From Equation (3.9) and (3.3) we can estimate 16000 power difference values results in a probability of 98.0%. Hence, with very high probability, 16000 power samples are enough to attack E0. Once the LFSR bit values are known, the four combining function state bits can be determined by exhaustively testing each possible value.

We can reduce the number of required power samples by determining the smallest LFSR at the first. When we collect enough $PD_t = +8$ or $PD_t = -8$ to find 25 linearly independent equations to solve $LFSR_a$, we can calculate its PD values and deduct from total PD of the circuit. Then, to obtain the state bits of the other LFSRs we look for $PD'_t = +7$ and $PD'_t = -7$, where PD'_t is the PD_t of circuit subtracting the

PD_t of $LFSR_a$. With 45 useful power difference values (i.e., $PD_t = +8$ or $PD_t = -8$) with the probability of more than 99.5% we have 25 linearly independent equations and we can calculate $LFSR_a$ state bits. Using equations (3.9) and (3.3) shows 12000 is enough to calculate the state bits of $LFSR_a$ and then, other LFSR state bits of E0 with a probability higher than 99%.

3.3 Application of the Attack to Irregular Clocking Stream Cipher, LILI-128

So far we have described an SPA attack on stream ciphers with regular clocks. In this section, we use SPA to attack a non-regular clocking LFSR stream cipher, LILI-128 [24]. In LILI-128, two LFSRs are employed ($LFSR_c$ and $LFSR_d$) to generate a random sequence. $LFSR_c$ is 39 bits in length and controls the clock of $LFSR_d$ which is 89 bits in length. The bit values of $c_t(12)$ and $c_t(20)$ in $LFSR_c$ are passed through a function with two bits output, to determine whether $LFSR_d$ should be clocked one, two, three or four times to produce key stream bits [24]. Since it is not known how many bits $LFSR_d$ is being clocked to produce each output bit, we cannot directly approach the equations for $LFSR_d$. Hence, at first we should find the bit values of $LFSR_c$.

Two different architectures have been offered for LILI-128 [24]. In the first architecture, two clocks are employed with different speeds. The first clock is used for $LFSR_c$ and the second one is for $LFSR_d$ which is four times faster. If $c_t(12) = 1$ and $c_t(20) = 1$, then $LFSR_d$ is clocked four times. To use simple power analysis and set up the equations, we should wait until $PD_t = +2$ or $PD_t = -2$ at the triggering edge of the first clock (i.e. the clock driving $LFSR_c$). When$PD_t = +2$ or $PD_t = -2$, we can write:

$$c_{t-1}(0) \oplus c_t(0) \oplus c_{t-1}(39) \oplus c_t(39) = 1. \tag{3.10}$$

No useful information can be obtained for $LFSR_d$, because t is not known for $LFSR_d$. Hence, at this point we cannot find any equation for $LFSR_d$. More information could be obtained by considering power consumption correlated to the $LFSR_d$ clock. If power consumption could be observed for $LFSR_d$ between two consecutive clocks of $LFSR_c$, indicating four shifts of $LFSR_d$ we can conclude:

$$\begin{aligned} c_t(12) &= 1 \qquad (3.11) \\ c_t(20) &= 1 \end{aligned}$$

Using equation (3.10) and (3.11), we can set up a system of linear equations to find the bit values of $LFSR_c$. Finding the bit values of $LFSR_c$, we can use the former power difference values to find the equations for bit values of $LFSR_d$.

In the second offered architecture for LILI-128 [24], just one clock has been used for both LFSRs. $LFSR_d$ is implemented using four copies of the feedback function and the irregular clocking is performed in one clock cycle. For this architecture, equation (3.11) can not be used; hence just equation (3.10) could be employed to realize $LFSR_c$ bit values.

Since the size of $LFSR_c$ is 39 bits, 60 equations with the probability of more than 99.2% can provide 39 linearly independent equations. In the second architecture, 600 power samples can provide 60 usable power difference values, with the probability of 97.5%. Hence, the second architecture is susceptible to SPA with 600 power samples with high probability. In the first architecture with the probability of $\frac{1}{8}$, equation (3.10) can be obtained and with the probability of $(\frac{1}{8})(\frac{1}{2})$ equation (3.11) can be employed in the system. After collecting 300 power samples, with the probability of more than 98.2%, 60 equations can be obtained to solve state bits of $LFSR_c$.

When bit registers of $LFSR_c$ are known, finding bits of $LFSR_d$ is similar to using SPA to attack one LFSR, proposed in [41]. To break $LFSR_d$, if we collect 110 equations, with the probability of more than 99% we will have 89 linearly independent equations.

3.4 Summary

In this chapter, we have extended the former method of simple power analysis attack proposed for one LFSR-based stream ciphers to ciphers based on multiple LFSRs. Also, we extend the proposed method to stream ciphers with irregular clocking LFSRs.

In order to illustrate the proposed methods, we applied them to well known stream ciphers E0 which includes four LFSRs and four bit registers and LILI-128 which includes two LFSRs, one with irregular clocking. We have shown that E0 could be broken using a few thousand power samples and LILI-128 is susceptible to simple power analysis with few hundred power samples. However, it should be noted that these results are based on the assumption that the Hamming distance model holds. That is, the theoretical power difference, which is related directly to the Hamming distance of the register values between clock cycles, is accurately determined from the analysis of power traces.

Chapter 4

Side Channel Analysis of NLFSR Based Stream Ciphers

An NLFSR has a similar structure to an LFSR as shown in Figure 2.1, except the feedback function is nonlinear. In order to make stream ciphers more secure, particularly against algebraic attack, NLFSRs are widely used in stream ciphers. For example, the Grain stream cipher [8] combines the outputs of an LFSR and NLFSR to produce the keystream. Since in an NLFSR, the feedback is nonlinear, using the described method in Section 2.5 and [41] results in a system of nonlinear equations which are difficult to solve. In a secure NLFSR, the order of equations relating output bits to the initial state bits increases very quickly and makes it difficult to solve the system. Hence, the formerly proposed methods of using simple power analysis for LFSRs are not applicable to NLFSR based stream ciphers. In this chapter we propose a new method to use simple power analysis against NLFSR based stream ciphers. The proposed method is applicable in ideal environments in which the measured power difference of the circuit matches the theoretical power difference values of the circuit. Then, we apply the proposed method to the Grain stream cipher to get the state bits from power consumption measurements. The results in this chapter were initially presented in [65].

4.1 Idealized SPA Applied to NLFSRs

Since, in a typical stream cipher, the key bits are used to initialize the NLFSR state, finding the state of the NLFSR (i.e., the L bits of the register) at any time is sufficient to break the system and determine the subsequent keystream bits. As in the previous section, we assume that the measured power consumption resulting in the measured power difference at time t, MPD_t, can be accurately converted to the theoretical power difference, PD_t. (In subsequent sections, we will discuss practical issues such as the inaccurate determination of PD_t values.)

Consider a consecutive series of PD_t values for an NLFSR with the length of L bits and denote the i-th bit of the NLFSR at time t as $b_t(i)$. In order to calculate NLFSR bit values, we should modify the former equations proposed in Section 2.5 to analyze an LFSR. Similar to equation (2.10), we can write:

$$PD_t = [b_t(L) \oplus b_{t-1}(L)] - [b_t(0) \oplus b_{t-1}(0)]. \tag{4.1}$$

Then, when $PD_t = +1$, we conclude

$$\begin{aligned} b_t(L) \oplus b_{t-1}(L) = 1 \\ b_t(0) \oplus b_{t-1}(0) = 0 \end{aligned} \tag{4.2}$$

and, when $PD_t = -1$, we can write

$$\begin{aligned} b_t(L) \oplus b_{t-1}(L) &= 0 \\ b_t(0) \oplus b_{t-1}(0) &= 1. \end{aligned} \tag{4.3}$$

To apply simple power analysis to an LFSR we only used the absolute value of PD_t. However to apply SPA to an NLFSR, the attacker should consider whether PD_t is greater or less than zero. When $PD_t = 0$, the two bracketed Xor results of equation (4.1) are both equal to either 0 or 1 and we can write

$$b_t(L) \oplus b_{t-1}(L) = b_t(0) \oplus b_{t-1}(0). \tag{4.4}$$

As long as $PD_t \neq 0$, we can find a relation between two consecutive values of the NLFSR bits, using equations (4.2) or (4.3).

To analyze the NLFSR, we must obtain L consecutive bits of the NLFSR. Equations (4.2) and (4.3) could determine the relation between two bits of the NLFSR when $PD_t = +1$ or $PD_t = -1$. However, when $PD_t = 0$, we cannot use equations (4.2) and (4.3) directly. Instead, we make use of equation (2.11) for PD_t and PD_{t+L} to obtain

$$\begin{aligned} |PD_t| \oplus |PD_{t+L}| &= b_t(L) \oplus b_{t-1}(L) \oplus b_t(0) \oplus b_{t-1}(0) \oplus b_{t+L}(L) \\ &\quad \oplus b_{t+L-1}(L) \oplus b_{t+L}(0) \oplus b_{t+L-1}(0) \\ &= b_t(0) \oplus b_{t-1}(0) \oplus b_t(2L) \oplus b_{t-1}(2L) \end{aligned} \tag{4.5}$$

where we have made use of $b_{t+j}(i) = b_t(i+j)$. Also, it can be shown that

$$PD_t + PD_{t+L} = [b_t(2L) \oplus b_{t-1}(2L)] - [b_t(0) \oplus b_{t-1}(0)]. \tag{4.6}$$

The value of PD_{t+i} must be $+1$, 0 or -1 implying $|PD_{t+i}| \in \{0, 1\}$. Since $|PD_t| \oplus |PD_{t+L}|$ will be either 1 or 0, if $PD_t = 0$, then we can write equation (4.2) or (4.3) for PD_{t+L} if $|PD_{t+L}|$ is 1 and using equation (4.5) find the relation between $b_t(0)$ and $b_{t-1}(0)$. For example, let us assume $PD_t = 0$. If $PD_{t+L} = +1$ or -1, then $b_t(2L) \oplus b_{t-1}(2L)$ and $b_t(L) \oplus b_{t-1}(L)$ are known from either equation (4.2) or (4.3) (with t replaced with $t + L$) and since the left side of equation (4.5) is known from power measurements then $b_t(0) \oplus b_{t-1}(0)$ can be inferred. If $PD_{t+L} = 0$, then power differences from cycle $t + 2L$ must be considered.

Now using equations (4.2) or (4.3) and (4.5), if necessary, the relationships between L pairs of consecutive bits are known. Although the actual values of the bits are not known, there are only two possibilities and both can be tested to determine which results in the correct state of the NLFSR. Since for this method, the feedback relation is not used, we can use the approach for both an NLFSR and LFSR. This method has the advantage that there is no need to solve a system of equations.

4.2 Complexity vs Available Power Samples

From equation (4.1), it is easy to see that the probability of PD_t equal to zero is $\frac{1}{2}$. Hence, we need to obtain PD_{t+L} for, on average, $\frac{1}{2}$ of L consecutive PD_t values. On average, $\frac{1}{2}$ of the values of PD_{t+L} are equal to the zero and we need to collect PD_{t+2L} values. In other words, on average for $\frac{1}{2}$ of L consecutive bits we are targeting, we need to collect PD_{t+L} values; for $\frac{1}{4}$ of the L consecutive bits, we need to collect PD_{t+2L} values, etc. In practical applications to analyze the sequence of an NLFSR, it is sufficient to find any consecutive L bits of the NLFSR. Hence, the analysis initially collects a number of consecutive power samples and then analyzes the values. In order to estimate the probability of a successful analysis, we assume $n \times L$ consecutive power difference values have been collected. The probability of all PD_{t+iL} values being zero for $0 \leq i < n$ and a fixed value of t (and therefore not being usable to determine bits in the register) is 2^{-n}. If we assume the occurrence of $PD_t = 0$ for different values of t are independent, then, given $n \times L$ power difference values, the probability that this is enough samples to analyze the NLFSR is $[1-2^{-n}]^L$. For $L \ll 2^n$, this probability is approximately $1 - 2^{-n}L$. So, for example, for $L = 80$, 800 consecutive power samples (i.e., $n = 10$) will allow successful analysis with a probability of about 92%.

If the number of available PD_t are limited, still we can calculate the state bits of the NLFSR and successfully analyze the NLFSR. This can be done by increasing the complexity of the attack. If a target PD_t is equal to zero and all available PD_{t+iL} (for $0 \leq i < n$) are zero, we can not directly guess the relationship of $b_t(0)$ and $b_{t-1}(0)$. In that case, we should test both possible relationships for $b_t(0)$ and $b_{t-1}(0)$ and check the correctness of each of them. The possible relationships between these two boolean values are $b_t(0) = b_{t-1}(0)$ and $b_t(0) \neq b_{t-1}(0)$.

As discussed before, the probability of PD_t equal to zero is $\frac{1}{2}$. Let the available number of PD_t values, N equal to $2L$ (i.e. $n = 2$). On average for $\frac{1}{4}$ of PD_t, we can not use equations (4.1), (4.2) and (4.3) and we should check all possible relationships for the corresponding consecutive bits. This increases the average-case complexity of the attack from 2 to $2^{\frac{L}{4}+1}$. For example, to analyze an 80 bit length NLFSR with available 160 PD_t values, the average-case complexity of the attack is 2^{21}. If the number of available PD_t increase to 320, the average-case complexity of the attack is reduced to 2^6. In general, the average-case complexity of the attack to an L bit length when $n \times L$ samples are used is given by $2^{L\times 2^{-n}+1}$.

To check the correctness of a guess, after calculating L consecutive bits ($b_t(0)$ to

$b_t(L-1)$) we should use the feedback to calculate extra bits $b_t(j)$ for $j \geq L$. Using the calculated $b_t(i)$, we should compute PD_t values and compare them with the available PD_t. If they match, we can conclude our guess is correct and if they don't match, the guessed relations are incorrect.

4.3 Applying SPA to Grain

As a practical application for the proposed method, we consider the stream cipher Grain [8]. Grain is a keystream generator designed for efficient hardware implementations, based on the nonlinear mixing of data from an 80-bit linear feedback shift register (LFSR) and an 80-bit nonlinear feedback shift register (NLFSR).

Since Grain uses two feedback shift registers (one LFSR and one NLFSR), we need to consider the methods summarized in Chapter 3 which describe a theoretical attack on stream ciphers with multiple LFSRs, where it is assumed that the attack takes place in circumstances where measured power traces perfectly map to the correct PD_t values. However, the proposed attack of Chapter 3 can not be applied directly on NLFSR based ciphers, such as Grain, since it relies on constructing and solving a system of linear equations.

To extend the attack to Grain, we can use the proposed method discussed in Section 4.1 which is applied to an NLFSR assuming perfect mapping from power measurements to the correct PD_t values. Since for the Hamming distance power model, we know that the overall power consumption of Grain is approximated by the summation of power consumption of the LFSR and the NLFSR (and it is assumed power consumed in other parts of the circuit at the triggering edge of the clock is negligible), measuring the power at the triggering edge of the clock, embodies the power consumption of the D flip-flops of both the LFSR and NLFSR. If we assume the power consumption of the circuit at time t (at the triggering edge) is the summation of the power consumption of the LFSR and the NLFSR (which is also proportional to the Hamming distance of their consecutive states), then we can conclude the overall dynamic power dissipation of the circuit at the triggering edge of the clock is proportional to $HD_t^{LFSR} + HD_t^{NLFSR}$. Hence, we can define the power difference of the circuit as

$$\begin{aligned} PD_t^{Grain} &= [HD_{t+1}^{LFSR} + HD_{t+1}^{NLFSR}] \\ &\quad -[HD_t^{LFSR} + HD_t^{NLFSR}] \\ &= PD_t^{LFSR} + PD_t^{NLFSR}. \end{aligned} \tag{4.7}$$

As shown in Section 2.5 and Section 4.1, PD_t^{LFSR} and PD_t^{NLFSR} values can be -1, 0 or $+1$, and, hence, $-2 \leq PD_t^{Grain} \leq +2$. As described in Section 3.1, if $PD_t^{Grain} = +2$ or -2, then we can conclude $(PD_t^{LFSR}, PD_t^{NLFSR}) = (+1, +1)$ or $(-1, -1)$, respectively. Hence, for $PD_t^{Grain} = +2$ or -2, we can use the proposed method in Section 2.5 and set up a system of linear equations to get the bit values of the LFSR. To complete the attack and get the bit values of the NLFSR, we use the proposed method described in Section 4.1.

To calculate the bit values of the NLFSR, we should have 80 consecutive PD_t^{NLFSR} values. To obtain 80 consecutive PD_t^{NLFSR} values we should at first collect enough power samples so that we have several hundred values of $PD_t^{Grain} = +2$ or -2 and we can find 80 power differences that lead to independent linear equations. Using these, we can calculate LFSR bit values. After calculating LFSR bit values, we should calculate PD_t^{LFSR} values for a few hundred consecutive clocks. Finally, deducting calculated PD_t^{LFSR} from the measured PD_t^{Grain}, we have a few hundred PD_t^{NLFSR} values. Using the proposed method in Section 4.1, we can calculate the bit values of the NLFSR.

The probability of $PD_t^{Grain} = +2$ or -2 is 1/8. As discussed in Section 3.1, when considering the 80-bit LFSR of Grain, to solve the system of 80 linear equations, somewhat more than 80 power difference values are required to ensure that we can obtain 80 linearly independent equations. Based on the analysis provided in Section 3.1, from 120 random linear equations, the probability that at least 80 equations will be linearly independent is greater than 99.99%. To obtain 120 equations, on average, 960 power samples should be collected. Using 1200 power difference values, as calculated in Section 3.1, the probability of 120 usable power difference values is greater than 98.99%. Making use of the analysis method in Section 4.1, the probability of a successful attack on an 80-bit NLFSR when 1200 power samples have been collected is $(1-2^{-15})^{80} \approx 99.8\%$. Hence, we can conclude that with 1200 power samples, Grain is theoretically susceptible to an SPA attack with very high probability. This represents an attack on Grain that is substantially less complex than exhaustive key search, which requires as much as the analysis of 2^{80} values for the 80-bit key of Grain.

4.4 Summary

In this chapter, we have discussed the application of a simple power analysis attack to NLFSR based stream ciphers and we have applied the techniques to the Grain stream cipher. We assume an ideal environment where the Hamming distance model can be applied perfectly to relate power measurements directly to changes in the cipher's state registers. Under these conditions, Grain would be susceptible to power analysis with only a few hundred power samples. However, this is an idealized result and difficulties would exist in mounting a practical attack which can not assume that measured power differences can be perfectly related to register data. Nevertheless, the results presented here do illustrate the potential vulnerability of stream ciphers based on LFSRs and NLFSRs to power analysis attacks.

Chapter 5

Practical Application of SPA

The analysis outlined in the previous sections and the previous works (such as [41]) is idealized in that it assumes a perfect determination of PD_t values from measured power differences, MPD_t. In this and the following sections, we consider the practical issues associated with applying simple power analysis to a practical CMOS circuit realization of an LFSR and/or an NLFSR when the measured power difference may not lead to the correct determination of PD_t. The first part of the research in this chapter is presented in [66].

5.1 Power Consumption of a Single D Flip-Flop

The D flip-flop, as building block of LFSRs and NLFSRs, is the main power consumer of stream ciphers. At the triggering edge of the clock, at first the D flip-flop gates change and subsequently, other gates which are connected to the output of the D flip-flops such linear or nonlinear combinational functions will change. Hence, at the triggering edge of the clock, theoretically the measured power consumption of the

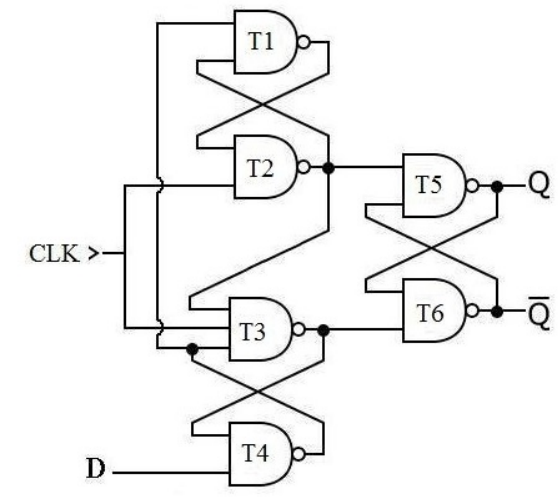
LFSR/NLFSR is mainly due to the changes in the D flip-flops.

In this section, we study two typical positive edge triggered D flip-flops, shown in Figure 5.1 (a) and (b). For our cryptographic circuits, we consider the first D flip-flop shown in Figure 5.1 (a). The first typical D flip-flop includes six NAND gates ($T1, T2, ..., T6$), two independent inputs (clock and D) and two dependent outputs (Q and $\overline{Q}$). The D flip-flop state, Q, changes only at the rising edge of the clock. The dynamic power (which is typically the dominant factor in power consumption of CMOS circuits) of the D flip-flop depends on the number of changing gates (resulting in transistor state changes).

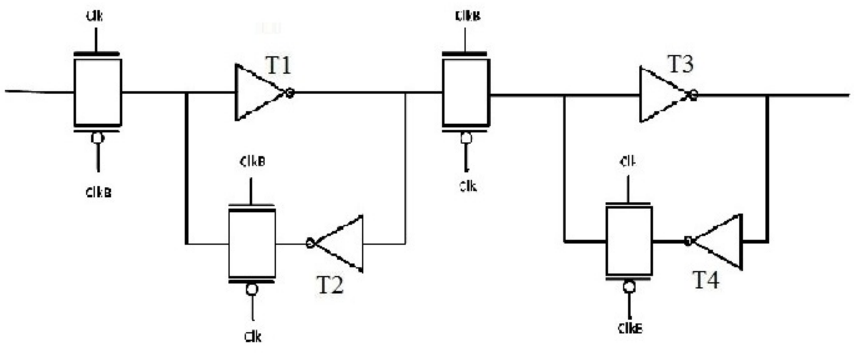
The second D flip-flop has three inputs. An independent input as D and two dependent input signals, clock and inverted clock (Clk and ClkB). The output signal is Q. The drawback of this D flip-flop is that it needs two clock signals; however the fewer number of transistors and less power consumption are advantages of this structure to the first one. It is constructed with four inverter gates and four CMOS transmission gates, which is in total 16 transistors.

5.1.1 Power Consumption of the D Flip-flop at the Rising Edge of the Clock

Previously proposed attacks assume the power consumption of the circuit at the rising (i.e., triggering) edge of the clock. Since, at the rising edge of the clock, the value of the register can change, we can conclude some gate outputs and transistor states are changed. As can be seen from Figure 5.1 (a), when $D = 0$ and $Q = 0$, at the rising edge of the clock only $T3$ changes, and, when $D = 1$ and $Q = 1$, at the rising edge



(a)



(b)

Figure 5.1: The typical architectures for D flip-flop: (a) classical structure; (b) alternate structure.

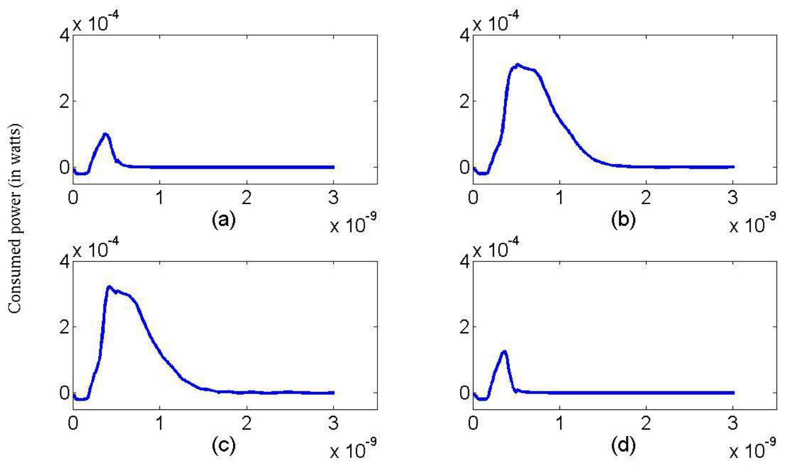


Figure 5.2: Power consumption of classical D flip-flop at the rising edge (in watts) versus time (in seconds), when (a) $D = 0$ and $Q = 0$, (b) $D = 0$ and $Q = 1$, (c) $D = 1$ and $Q = 0$, (d) $D = 1$ and $Q = 1$.

only $T2$ changes. When $D = 0$ and $Q = 1$, at the rising edge of the clock, three gates ($T3$, $T5$ and $T6$) change. Three gates ($T2$, $T5$ and $T6$) also change, when $D = 1$ and $Q = 0$. Hence, we expect more power to be consumed at the rising edge when $D = 1$ and $Q = 0$ or when $D = 0$ and $Q = 1$, compared to when $D = 0$ and $Q = 0$ or when $D = 1$ and $Q = 1$. In other words, when there is a D flip-flop state change, we expect more power consumption. This is consistent with the Hamming distance power model used in our proposed analysis and the approach of others.

In Figure 5.2, the power consumption of a single classical D flip-flop (shown in Figure 5.1 (a)) for different inputs and outputs is shown. In Figure 5.2, the vertical axis represents the consumed power and the horizontal axis represents time. The rising clock edge occurs at $t = 0$ and the rise time is 20 ns. To investigate our methods, we have used Cadence Virtuoso Spectre Circuit Simulator version 5.10.41

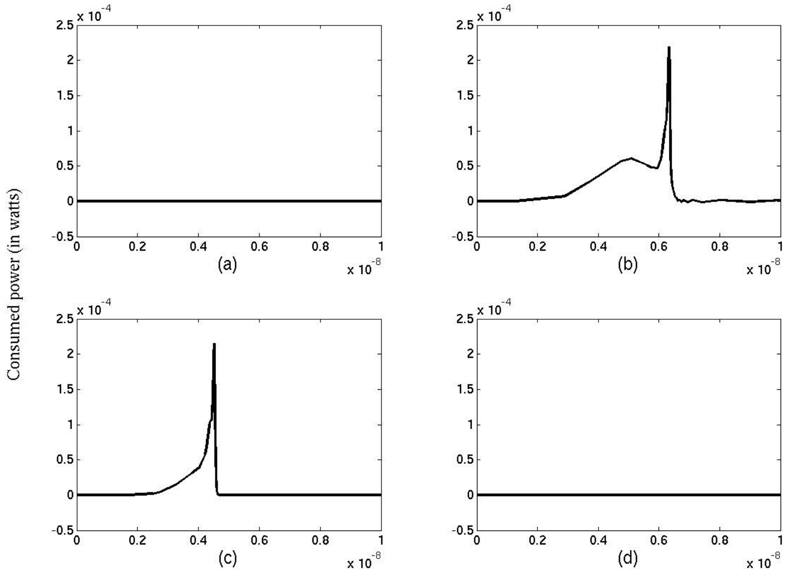


Figure 5.3: Power consumption of alternate D flip-flop at the rising edge (in watts) versus time (in seconds), when (a) $D = 0$ and $Q = 0$, (b) $D = 0$ and $Q = 1$, (c) $D = 1$ and $Q = 0$, (d) $D = 1$ and $Q = 1$.

to obtain the power consumption of the circuit. All the circuits here are prototyped in TSMC 180 nm standard cell CMOS technology. The supply voltage of all circuits is 1.8 volts and the experiments have been done assuming room temperature and default noise.

Under the same condition, we measure the power consumption of the alternate D flip-flop, shown in Figure 5.1 (b). The results for different inputs and outputs are shown in Figure 5.3. As can be seen from Figure 5.1 (b), when $D = 0$ and $Q = 0$ and when $D = 1$ and $Q = 1$ at the rising edge of the clock, no gate changes. When $D = 0$ and $Q = 1$, at the rising edge of the clock, two gates ($T3$ and $T4$) change and the same two gates change when $D = 1$ and $Q = 0$.

For an LFSR or NLFSR, the power consumption of the circuit at the rising edge of the clock corresponds to the summation of power consumption of each single D flip-flop (plus a small amount of power consumption due to the subsequent changes in the combinational logic in the feedback and output functions). Hence, HD_t, in general, is expected to be proportional to the summation of the consumed power of each D flip-flop.

5.1.2 Power Consumption of the D Flip-flop at the Falling Edge of the Clock

Studying the architectures of the positive edge triggered D flip-flop, we can see at the falling (i.e., non-triggering) edge of the clock, we have changes in some gates and transistor states. For the classical D flip-flop (Figure 5.1 (a)), when $D = 1$ and $Q = 1$, at the falling edge of the clock, one gate will change ($T2$), and, when $D = 0$ and $Q = 0$, $T3$ will change. Meanwhile, for $D = 0$ and $Q = 1$, two gates ($T1$ and $T2$), and, for $D = 1$ and $Q = 0$, three gates ($T1$, $T3$ and $T4$) will change at the falling edge. The power consumption of a D flip-flop at a falling edge of the clock for different D and Q are shown in Figure 5.4 where the vertical axis represents the consumed power and the horizontal axis represents time. The falling clock edge occurs at $t = 0$ and the rising time is 20 ns.

Studying the alternate D flip-flop of Figure 5.1 (b), shows that some gates can change at the falling edge of the clock for this architecture. When $D = 0$ and $Q = 0$ and when $D = 1$ and $Q = 1$, at the falling edge of the clock, no gate changes. When $D = 0$ and $Q = 1$, at the falling edge of the clock, two gates ($T1$ and $T2$) change.

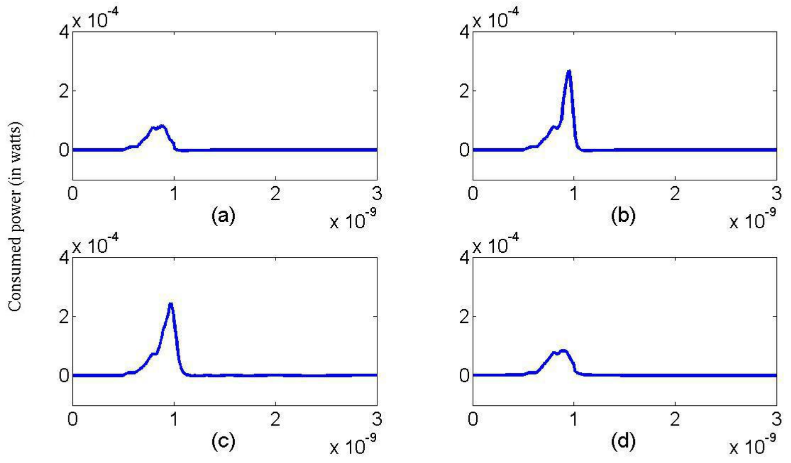


Figure 5.4: Power consumption of classical D flip-flop at the falling edge (in watts) versus time (in seconds), when (a) $D = 0$ and $Q = 0$; (b) $D = 0$ and $Q = 1$; (c) $D = 1$ and $Q = 0$; (d) $D = 1$ and $Q = 1$.

When $D = 1$ and $Q = 0$, $T1$ and $T2$ gates change at the falling edge of the clock. The power consumption of a D flip-flop at the falling edge of the clock, for different input and outputs (D and Q) are shown in the Figure 5.5.

Considering the consumed power at the falling edge of the clock to analyze the cryptographic circuits has not been discussed in the previous literature. In the subsequent sections, we use the power consumption of the circuit at the falling edge of the clock, in addition to the rising edge, to analyze cryptographic circuits. Obviously, this technique is not only applicable to stream ciphers and it could be applied to block ciphers and public key cryptographic circuits, as well.

The apparent advantage of simple power analysis using the falling edge (which we will refer to as falling edge SPA or FESPA) is that on a falling edge D flip-flop states do not change. Hence, there is no change in the state of the circuit (until the

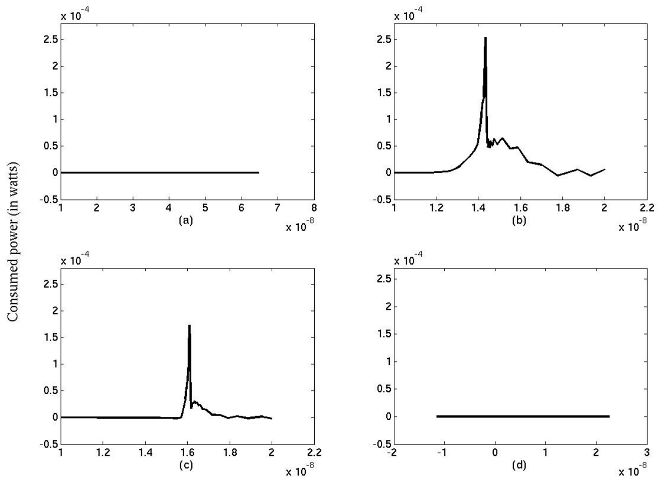


Figure 5.5: Power consumption of alternate D flip-flop at the falling edge (in watts) versus time (in seconds), when (a) $D = 0$ and $Q = 0$; (b) $D = 0$ and $Q = 1$; (c) $D = 1$ and $Q = 0$; (d) $D = 1$ and $Q = 1$.

subsequent rising edge) and power consumption of combinational logic for feedback or output does not interfere with measurements on the falling edge. In contrast, on the rising edge of the clock, the state of the circuit changes and the measured power consumption is the consumed power of D flip-flops followed by the consumed power of combinational circuits including feedback and output. Hence, on the falling edge, we expect to have better correlation between the measured power consumption of the circuit and subsequent changes in the register bits then at the rising edge. That is, the values of PD_t determined by measurement should have fewer errors for the falling edge and the Hamming distance model used in SPA would seem to be more accurate for the falling edge, rather than the rising edge. However, this turns out to not be the case. Consider that, as shown in Figure 5.4, the power consumption curve of a D flip-flop at the falling edge has sharp tips in comparison to the power consumption graph for the rising edge (for our cryptographic circuits, we have used the D flip-flop of Figure 5.1 (a)). Hence, in circuits with multiple D flip-flops, when the clock signal has small differences in delay to the flip-flops, the tips may not align. As a result, the power consumption at the falling edge for the overall circuit will not necessarily correlate exactly to the sum of the individual D flip-flops' power consumption. Therefore, we expect that, due to variation in clock propagation delays in large CMOS circuits, challenges will exist with FESPA. However, although it may be difficult to use FESPA on its own, as we shall see, the extra information derived from the falling edge is useful in combination with information from the rising edge for a practical application of SPA.

Most of the proposed architectures for D flip-flops are based on *master-slave* structure. Consider a D flip-flop based on master-slave structure at time t. At the falling edge of the clock, the *master* stores the value of the input. If the stored value in the master at time $t-1$ is different than the input value of the D flip-flop at time t, storing the new value in the master makes a change in the gate and transistor states of the master of the D flip-flop. Hence, if the input value at time t is different than input value at time $t-1$ (which is equal to the output of the D flip-flop at time t) definitely we expect power consumption in the master part of the D flip-flop. If the stored value at time $t-1$ and input at time t are identical, a major change is not expected in the master and therefore large power consumption is not expected.

The same concept is applicable on rising edge and *slave* part of the D flip-flops. If the stored value of the master at time t is different than the output value of the D flip-flop (or slave), we expect change in the states of the slave and/or its transistors and gates. If the stored value at time $t-1$ at the slave and its input at time t are identical, no major change and power consumption is expected in the slave part of the D flip-flop. As result, in general, we expect more power consumption for master-slave based D flip-flops when $D \neq Q$ rather than when $D = Q$. Therefore we expect the proposed techniques in this research to be applicable to most of the master-slave based D flip-flop structures.

5.2 Developing Falling Edge SPA of LFSR/NLFSR

In this section, we consider the application of SPA on an LFSR or NLFSR based on power consumption information from the falling edge. As shown in Figure 5.4 and Figure 5.5, based on D and Q values of the D flip-flop at the falling edge, D flip-flops consume different values of power. To describe the consumed power for different inputs, we define a new variable representing the Hamming distance, $\Delta(i)$, between D and Q for bit i of the register. If $D = 0$ and $Q = 0$, or $D = 1$ and $Q = 1$ (when the power consumption of the D flip-flop is small), $\Delta(i) = 0$. If $D = 0$ and $Q = 1$, or $D = 1$ and $Q = 0$ (when the power consumption of the D flip-flop is large), $\Delta(i) = 1$.

Assume an LFSR or NLFSR with the size of L. At time t, we represent the state of the i-th D flip-flop as $s_t(i)$, $0 \leq i < L$, The overall power consumption of the register at the falling edge for time t is proportional to

$$HD_t = \sum_{i=0}^{L-1} \Delta_t(i), \tag{5.1}$$

where $\Delta_t(i)$ is the Hamming distance between D and Q for $s_t(i)$, the register bit i at time t. Note that we label time t in the context of the falling edge as the falling edge preceding the rising edge at time t. Hence, $\Delta_t(i) = s_t(i) \oplus s_{t-1}(i)$ and

$\Delta_t(i)$ is proportional to the power consumption of the corresponding D flip-flop. At the time $t+1$, the state of each register bit is shifted to the next register (that is, $s_t(i+1) = s_{t+1}(i)$) and we can write $\Delta_t(i+1) = \Delta_{t+1}(i)$ or $\Delta_t(i) = \Delta_{t+1}(i-1)$. Then, the power consumption of the LFSR/NLFSR at the falling edge of time $t+1$ is proportional to

$$\begin{aligned} HD_{t+1} &= \sum_{i=0}^{L-1} \Delta_{t+1}(i) \\ &= \sum_{i=0}^{L-1} \Delta_t(i+1) \\ &= \sum_{i=1}^{L} \Delta_t(i). \end{aligned} \tag{5.2}$$

The theoretical power difference of the LFSR/NLFSR at time $t+1$ and t is defined as PD_t (as for the rising edge) and is equal to

$$\begin{aligned} PD_t &= HD_{t+1} - HD_t \\ &= \sum_{i=1}^{L} \Delta_t(i) - \sum_{i=0}^{L-1} \Delta_t(i) \\ &= \Delta_t(L) - \Delta_t(0). \end{aligned} \tag{5.3}$$

As a result, using the definition of $\Delta_t(i)$, we get

$$PD_t = [s_t(L) \oplus s_{t-1}(L)] - [s_t(0) \oplus s_{t-1}(0)], \tag{5.4}$$

which is identical to equation (4.1) except that the power difference refers to the power difference of the falling edge prior to the rising edge at time t. Hence, measuring the difference of the power consumption of the LFSR/NLFSR at two consecutive falling edges of each clock helps us to guess PD_t and, consequently, the relation between $s_t(L)$, $s_{t-1}(L)$, $s_t(0)$ and $s_{t-1}(0)$.

Similar to the rising edge, if the measured power difference is small (which corresponds to $PD_t = 0$), then:

$$s_t(L) \oplus s_{t-1}(L) = s_t(0) \oplus s_{t-1}(0). \tag{5.5}$$

When the measured power difference is significant and positive ($PD_t = +1$)

$$\begin{aligned} s_t(L) \oplus s_{t-1}(L) &= 1 \\ s_t(0) \oplus s_{t-1}(0) &= 0, \end{aligned} \tag{5.6}$$

and when the measured power difference is significant and negative ($PD_t = -1$)

$$\begin{aligned} s_t(L) \oplus s_{t-1}(L) &= 0 \\ s_t(0) \oplus s_{t-1}(0) &= 1. \end{aligned} \tag{5.7}$$

These relations are similar to equations (4.1), (4.2) and (4.3) from rising edge SPA (RESPA). Hence, we can theoretically use the same technique to analyze LFSRs or NLFSRs for the falling edge as for the rising edge power analysis outlined in the previous sections.

5.3 Categorization of Power Measurements

So far, the proposed simple power analyses of stream ciphers have ignored the effect of inaccurately mapping from analog MPD values to discrete theoretical PD values caused by effects such as power consumption sources other than the flip-flops (i.e., the combinational logic in the feedback or output functions) and clock skew. In this section, we study the effect of inaccuracies in categorizing measured power consumption for simple power analysis based on experimental results from simulation.

When we measure the power consumption of the circuit and subtract their values at consecutive rising/falling edges (to obtain MPD), we have analog values, which we should map to discrete PD values, where $PD \in \{+1, 0, -1\}$. For convenience, we often drop the subscript t when referring to power difference values. In the following, we offer a method to map or categorize MPD values to $\{+1, 0, -1\}$. Then we offer some techniques to distinguish incorrectly categorized MPD, that is, incorrectly determined values for PD. The categorized value for MPD is denoted by PD^g, while the correct theoretical power difference based on actual data in the register is simply notated PD. Hence, when we map the measured power difference at time t, MPD_t, to a categorized power difference PD_t^g, a correct categorization would mean that $PD_t^g = PD_t$.

It should be noted that if measured power differences are randomly mapped to a category of $\{-1, 0, +1\}$, the probability that the categorization would be correct, is given by

$$\sum_{i \in \{-1,0,+1\}} P\left(PD^g = i | PD = i\right) \cdot P\left(PD = i\right) \tag{5.8}$$

where $P\left(PD = i\right)$ represents the probability that a power difference equals i and the conditional probability is calculated as $P\left(PD^g = i | PD = i\right) = P\left(PD^g = i\right) = P\left(PD = i\right)$ since the categorization is random relative to the actual PD value. Since it is reasonable to assume in an LFSR or NLFSR that the probabilities of generating 0 and 1 are equal, the probability of $PD = +1$, $PD = 0$ and $PD = -1$ are .25, .5 and .25, respectively. This results in the probability of correct categorization being 37.5% and the probability of an incorrect categorization being 62.5%.

5.3.1 Categorizing MPD

Since it is reasonable to assume in an LFSR or NLFSR that the probability of generating 0 and 1 are equal, the probability of $PD = +1$, $PD = 0$ and $PD = -1$ are .25, .5 and .25, respectively. In one simple approach to categorize the measured power difference (MPD) to their corresponding PD^g values, at first we sort MPD values for different t in order of their value. The smaller 25% of MPD values should be categorized to $PD^g = -1$. The largest 25% of MPD values should be categorized to $PD^g = +1$. The remaining MPD values should be categorized to $PD^g = 0$.

In our analysis, we use this method to categorize MPD. However, this method is not perfect and some MPD values may be categorized incorrectly. We have applied this method to an 80-bit NLFSR, although the approach would be equally applicable to an LFSR. For the implemented NLFSR we have used the classical D flip-flop structure shown in the Figure 5.1 (a). The NLFSR is prototyped in TSMC 180 nm standard cell CMOS technology. The supply voltage is 1.8 V and the rise time for the clock is 500 ps. The NLFSR used is equivalent to the NLFSR used in the Grain-v1 stream cipher [8] and its feedback is defined as

$$\begin{aligned} b_t(80) = {} & b_t(62) \oplus b_t(60) \oplus b_t(52) \oplus b_t(45) \oplus b_t(37) \oplus b_t(33) \oplus b_t(28) \\ & \oplus b_t(21) \oplus b_t(14) \oplus b_t(9) \oplus b_t(0) \oplus b_t(60) \cdot b_t(63) \oplus b_t(37) \\ & \cdot b_t(33) \oplus b_t(9) \cdot b_t(15) \oplus b_t(45) \cdot b_t(52) \cdot b_t(60) \oplus b_t(33) \cdot b_t(28) \\ & \cdot b_t(21) \oplus b_t(9) \cdot b_t(28) \cdot b_t(45) \cdot b_t(63) \oplus b_t(60) \cdot b_t(52) \cdot b_t(37) \cdot b_t(33) \\ & \oplus b_t(63) \cdot b_t(60) \cdot b_t(21) \cdot b_t(15) \oplus b_t(63) \cdot b_t(60) \cdot b_t(52) \cdot b_t(45) \cdot b_t(37) \\ & \oplus b_t(9) \cdot b_t(15) \cdot b_t(21) \cdot b_t(28) \cdot b_t(33) \oplus b_t(21) \cdot b_t(28) \cdot b_t(33) \cdot b_t(37) \\ & \cdot b_t(45) \cdot b_t(52) \end{aligned} \tag{5.9}$$

We use cadence to implement and simulate the power consumption of the NLFSR. All D flip-flops of the NLFSR loaded with one and ran for 20000 clock cyles. After collecting around 20000 power samples through simulation and applying our categorization method, we found about 16 percent of rising edge MPD values were categorized incorrectly. Incorrect categorization occurred for falling edge MPD values in about 32 percent of the cases. More analysis on the experimental results shows the probabilities of incorrectly categorizing an actual $PD = +1$ (or $PD = -1$) to $PD^g = -1$ (or to $PD^g = +1$) is negligible. In other words, virtually all categorization errors occur by incorrectly assigning $+1$ or -1 to 0, or 0 to $+1$ or -1.

Because of the abovementioned categorization errors, we must modify the proposed SPA in Sections 4.1 and 5.2 for real applications. In doing so, we must ensure that we can identify correctly categorized power differences with high probability and must reject power differences for which we are not confident in their correct categorization.

We have also applied our simulations to the full NLFSR based on the alternate

D flip-flop shown in Figure 5.1 (b) using a clock with a 50 ps transition time. Utilizing the power measurement categorization techniques resulted in measured power differences being incorrectly categorized 38% of the time for the rising edge and 50% of the time for the falling edge. Such probabilities are significantly different than the 62.5% probability of incorrectly categorizing if the power differences were randomly categorized and may therefore form the basis for a power analysis attack. However, compared to 16% and 32% for the classical D flip-flop, the incorrect categorization probability is substantially worse and we would expect that the attack will not have as much success as the results for the classical D flip-flop.

We conjecture that these poorer results occur because, as can be seen in Figures 5.3 and 5.5 the spikes of power consumption do not correlate in time well for this D flip-flop. So that the overall power consumption on a rising clock edge does not correlate well to the Hamming distance in the NLFSR data compared to the classical D flip-flop. In the dissertation, we have used the power consumption data generated from simulation of an NLFSR constructed using the classical D flip-flop structure of Figure 5.1 (a) to illustrate the potential applicability of the attack.

5.4 Basic Methods to Determine Correctly Categorized PD

Here we offer some techniques which help us to find, with high probability, correct PD^g, i.e., correctly categorized MPD values such that the measurement determined power difference, PD^g, equals the power difference that should result from the actual data, PD. For each of the proposed methods, we have determined experimentally (through simulation of the 80-bit NLFSR constructed from the D flip-flop shown in Figure 5.1 (a)) the probability of correct categorization, as well as the probability that the condition has occurred to allow us to categorize an MPD value with confidence.

5.4.1 Rising Edge/Falling Edge Equivalence

When we measure the power consumption of the circuit in simple power analysis, we can assume we have access to power consumption at both rising and falling edges. Based on experiments for our NLFSR, the probability of an incorrect PD^g in RESPA and FESPA are .160 and .320, respectively. Then, for any clock cycle, if the categorized values are the same (i.e., $PD^g_{Rising} = PD^g_{Falling}$) and we assume that the probability of correctness for the rising edge and the falling edge are independent, the probability that the categorized PD is incorrect is determined as the probability that both values are wrong and is therefore given by $.160 \times .320 = .051$. In other words, if categorization using falling edge and rising edge show the same value, this value is correct with a theoretical probability of .949, which is similar to the experimentally measured probability of .950. This represents a much higher level of confidence then taking, on their own, either the rising edge or falling edge categorization (which have probabilities of .840 and .680, respectively). Our experiments show that we can use this technique to ensure correct categorization for about 60% of the measurements from different clock cycles, with this high probability.

5.4.2 Robust Threshold

Another technique to help categorize MPD values accurately is using a more *robust threshold* value. In this technique, we change the threshold and, instead of 25%, we categorize the smallest and largest 12.5% as $PD = -1$ and $PD = +1$, respectively, and the middle 25% as $PD = 0$. In this approach, categorizations are correct with higher probability. We refer to the assigned PD values by this method as robust PD^g and use the label PD^{rg}.

Obviously, this technique can be applied to 50% of the MPD (for both rising edge and falling edge). Our experiments show, using this approach, PD^{rg} for rising edge is correct with a probability of .955, while for the falling edge PD^{rg}, the probability of correctness is .750.

5.4.3 Sequence Consistency

Another technique, which we call the *sequence consistency* method, can be used to improve categorization success by distinguishing correct categorizations from incorrect ones. To find the incorrect categorizations, we can use equation (4.6). In (4.6), the right side of the equation cannot be larger than +1 or smaller than −1; hence, at the left side PD_{t+L} cannot be equal to PD_t, unless both are equal to zero. Extrapolating equation (4.6), if we add j consecutive PD terms separated by L clock cycles, we get

$$\begin{aligned} &PD_t + PD_{t+L} + \ldots + PD_{t+(j-1)L} \\ &\quad = [s_t(jL) \oplus s_{t-1}(jL)] - [s_t(0) \oplus s_{t-1}(0)]. \end{aligned} \tag{5.10}$$

The right side must be from the set $\{-1, 0, +1\}$ and, hence, the summation of any j consecutive PD values L bits apart can never be larger than one or smaller than minus one. Hence, if $PD_t = +1$, then PD_{t+L} and PD_{t-L} must be either 0 or -1. Similarly, $PD_t = -1$ implies $PD_{t+L} = 0$ or $+1$ and $PD_{t-L} = 0$ or $+1$. If in any sequence of PD^g_{t+iL} values, we see two consecutive $+1$ values, we know that at least one of them is categorized incorrectly. In other words, a correct sequence of PD^g values separated by L clock cycles, starting with a value of $PD^g = +1$, must be followed by a string of some number of values of $PD^g = 0$ and then $PD^g = -1$. In contrast, a $+1$ value, any number of 0 values and then $+1$ indicates an incorrect categorization, i.e., at least one PD^g value is wrong. Similar analysis is true for a sequence starting with $PD^g = -1$.

In applying the sequence consistency technique, consider a sequence of three categorized values: $\{PD^g_{t-L}, PD^g_t, PD^g_{t+L}\}$. We can increase our confidence in a correct categorization of PD^g_t by considering the full sequence. For example, in RESPA, if we have categorized each value of sequence $\{PD^g_{t-L}, PD^g_t, PD^g_{t+L}\}$ as $\{+1, -1, +1\}$ independently, the probability that PD^g_t is correct is .840 (as determined by experiment). However, using the sequence consistency method, the probability of correctness of PD^g_t for this sequence is increased to .984. If the categorized sequence is $\{+1, -1, +1\}$, it means the actual PD sequence could be $\{0, -1, 0\}$, $\{0, -1, +1\}$, $\{+1, -1, 0\}$, $\{+1, -1, +1\}$, $\{0, 0, 0\}$, $\{0, 0, +1\}$ or $\{+1, 0, 0\}$. Sequences like $\{-1, 0, 0\}$ are not possible as the actual sequence because we assume the probability of categorizing an actual $PD = -1$ as $PD^g = +1$ is negligible. If any of the first four sequences is the actual sequence, our categorization of $PD^g_t = -1$ is correct and if any of the last three cases is the actual sequence, our categorization is incorrect. The probability of occurrence for each sequence is equal to $\frac{1}{16}$, except $\{0, 0, 0\}$ which is $\frac{1}{8}$. If we let the probability of any individual PD_t being correctly categorized be represented by P_{cr}, then the probability of an individual $PD_t = 0$ incorrectly categorized as $+1$ is equal to $\frac{1}{2}(1 - P_{cr})$ (A similar probability occurs for incorrectly categorizing 0 to -1). Hence, the probability of actual sequence $\{0, -1, +1\}$ categorized as $\{+1, -1, +1\}$ is equal to $\frac{1}{2}(1 - P_{cr})P_{cr}P_{cr}$. The probability of observing a sequence as $\{+1, -1, +1\}$ is equal to summation of the probabilities of occurrence of each sequence (either $\frac{1}{16}$ or $\frac{1}{8}$) times the probability of categorizing that sequence as $\{+1, -1, +1\}$.

From all possible sequences, we have selected 8 sequences with high probability for our purpose and list them in Table 5.1. In Appendix A, we have listed the probabilities for all possible sequences. Using the occurrence of these sequences on either the rising or falling edge as indicators of correct categorizations could increase the probability of categorizing MPD values correctly to a probability of .913 (as determined by experiment) and could be applied to 78% of all measured power differences.

5.5 Advanced Categorization Methods

In our analysis, we require PD^g with high probability of correctness. In the previous section, we have introduced some methods to distinguish PD^g which are likely to be correct. In this section, we derive PD^g with even higher probability of correctness by selecting PD^g values for which at least two of the above techniques are applicable. We list them as follows:

(I) *RE/FE Equivalence and Robust Threshold on RE*

Sequence $\{PD^g_{t-L}, PD^g_t, PD^g_{t+L}\}$	Probability of $PD^g_t = PD_t$ for rising edge	Probability of $PD^g_t = PD_t$ for falling edge
$\{+1, -1, +1\}$	.984	.918
$\{-1, +1, -1\}$	.984	.918
$\{+1, 0, -1\}$	.968	.852
$\{-1, 0, +1\}$	.968	.852
$\{0, +1, -1\}$	.906	.781
$\{0, -1, +1\}$	.906	.781
$\{-1, +1, 0\}$	.906	.781
$\{+1, -1, 0\}$	.906	.781

Table 5.1: Probability of $PD^g_t = PD_t$ for sequences of three PD^g values for rising edge ($P_{cr} = .840$) and falling edge ($P_{cr} = .680$).

In this case, two categorized PD^g values of rising edge and falling edge that are the same and consistent with the robust threshold of the rising edge are assumed correct. Notationally, we represent it as $PD^g_{Rising} = PD^g_{Falling}$ and $PD^g_{Rising} = PD^{rg}_{Rising}$.

The experimental results from the 80-bit NLFSR show that the probability of correctness for this case is .992, while the probability that such consistency occurs for a given clock cycle is .315.

(II) *RE/FE Equivalence and Robust Threshold on FE*

A similar approach could be taken based on consistency with the categorization based on the falling edge robust threshold. The experimental results show the probability of correctness for this case is .974, while the probability of occurrence of this case is .326.

(III) *RE/FE Equivalence and Sequence Consistency*

In this case, the two categorized values of rising edge and falling edge are the same and the sequence consistency method confirms their correctness. Notationally, we represent this as $PD^g_{Rising} = PD^g_{Falling}$ and $PD^g_{Rising} = PD^{sg}_{Rising}$ or $PD^g_{Rising} = PD^{sg}_{Falling}$, where PD^{sg} is used to represent a categorization of an MPD value based on the sequence consistency method.

Our experiments show the correctness of PD^g values in this case are .987, while the probability of such an occurrence is .326.

(IV) *Robust Threshold on RE/FE and Sequence Consistency*

In this case, the PD^g value is determined by the robust threshold of RESPA or

FESPA and the sequence method confirms it. Experiments show the probabilities of correctness and occurrence are .998 and .219, respectively.

Condition of PD		Probability of correctness (P_c)	Probability of occurrence (P_o)
PD^g_{Rising}		.840	1
$PD^g_{Falling}$		.680	1
$PD^g_{Rising} = PD^g_{Falling}$		.949	.600
Robust Threshold		.955	.500
Sequence Consistency		.913	.780
$PD^g_{Rising} = PD^g_{Falling}$	Robust Threshold RE	.992	.315
$PD^g_{Rising} = PD^g_{Falling}$	Robust Threshold FE	.974	.326
$PD^g_{Rising} = PD^g_{Falling}$	Sequence Consistency	.987	.326
Robust Threshold FE/FE	Sequence Consistency	.998	.219
(I) or (II) or (III) or (IV)		.975	.467

Table 5.2: Probability of correctness for PD for different methods

In Table 5.2 we summarize the experimental data. During the determination for any PD^g, if at least one of cases I, II, III or IV occurs, we assume that the categorization is correct. Based on the experimental results, the probability of at least one of the mentioned cases occurring for a PD^g is .467 and the probability of correctness is .975.

5.6 Analyzing an NLFSR

Although the techniques outlined here are equally applicable to LFSRs and NLFSR, in order to illustrate the approach we now consider the application of simple power analysis to the 80-bit NLFSR, using the probabilities derived from experimental results for the categorization methodologies previously described. On average, upon categorization of power measurement values, we expect that at least one of cases I, II, III, and IV occurs for $.467 \times 80 \approx 37$ PD^g values of the 80 bits and the resulting PD^g values are correct with high probability of about .975. However, half of these PD^g values will be equal to 0 and, as indicated in the Section 4.1, we cannot use them to obtain information on the NLFSR state bits.

Based on equations (4.2), (4.3) and (4.4), if we know $PD_{t-L} = +1$ or -1, we can find $s_t(0) \oplus s_{t-1}(0)$. Similarly, there are many scenarios for which knowing PD_{t-2L}, PD_{t+L}, or PD_{t+2L} are equal to $+1$ or -1 will allow us to determine $s_t(0) \oplus s_{t-1}(0)$. In Table 5.3, we have listed possible scenarios for PD^g_{t-2L}, PD^g_{t-L}, PD^g_t, PD^g_{t+L} and PD^g_{t+2L} that we could use to guess the relationships between $s_t(0)$ and $s_{t-1}(0)$. In the table, if from cases I, II, III or IV, $PD^g = \pm 1$, we present it as ± 1 and, if from cases I, II, III or IV, $PD^g = 0$, we present it as 0. If cases I, II, III and IV are not applicable

Scenario	PD^g_{t-2L}	PD^g_{t-L}	PD^g_t	PD^g_{t+L}	PD^g_{t+2L}	Probability
A	X	X	± 1	X	X	.233
B	X	± 1	$\hat{C}$	X	X	.124
C	X	± 1	0	X	X	.054
D	± 1	0	$\hat{C}$	X	X	.029
E	± 1	0	0	X	X	.013
F	$\hat{C}$	$\hat{C}$	0	± 1	X	.016
G	0	$\hat{C}$	0	± 1	X	.007
H	$\hat{C}$	0	0	± 1	X	.007
I	0	0	0	± 1	X	.003
J	$\hat{C}$	$\hat{C}$	0	0	± 1	.004
K	$\hat{C}$	0	0	0	± 1	.002
L	0	$\hat{C}$	0	0	± 1	.002
M	0	0	0	0	± 1	.001
N	± 1	$\hat{C}$	0	0	± 1	.002

Table 5.3: Cases used to determine $s_t(0) \oplus s_{t-1}(0)$.

to PD^g, we cannot be confident in the categorization of PD^g and we present it as $\hat{C}$. If the value of PD^g is not critical to defining the scenario in the table, we show it with "X" (i.e., the value is a "don't care").

The probability of occurrence for each scenario in the table is given in the right column. To calculate the listed probabilities, we assume the probability of $PD^g_t = +1$ or -1 is equal to the probability of $PD^g_t = 0$ and is therefore given by $\frac{.467}{2} = .233$. The probability of $PD^g_t = \hat{C}$ is $1 - .467 = .533$. Hence, the probability of scenario A is equal to the probability of $PD^g_t = +1$ or -1 and is therefore .233. The probability of scenario B is the probability of $PD^g_{t-L} = +1$ or -1 and $PD^g_t = \hat{C}$, which is $.233 \times .533 = .124$, where we have made the reasonable assumption that the power differences at times separated by L clock cycles are independent. For scenario C, the probability is calculated as $.233 \times .233 = .054$, which is equal to the probability of $PD^g_{t-L} = +1$ or -1 and $PD^g_t = 0$. The rest of the probabilities of Table 5.3 are calculated similarly.

All cases in the table are mutually exclusive; hence, the sum of the right column, which equals about .49, is the probability that one of the scenarios occurs. Therefore, we have about $80 \times .49 \approx 39$ relationships of pairs of consecutive bits with high probability of correctness and we can guess the remaining $80 - 39 = 41$ relationships. Considering the scenarios of Table 5.3, on average we would need about 55 PD^g values with high probability in order to determine the 39 Xor relationships with high probability. This is explained as follows. If either scenario A or B occurs, (which will happen with a probability of .233+.124=.357), we need only one PD^g with high probability. If scenarios C, D or F occur (which will happen with the probability of .099), we need two PD^g values with high probability of correctness. For scenarios E, G, H and J, we need to know three PD^g values and for I, K, L and N, we have to know four PD^g values. For M, we need to know five PD^g values. Hence, on average, we need to know $80 \times (1 \times .357 + 2 \times .099 + 3 \times .031 + 4 \times .009 + 5 \times .001) \approx 55$ PD^g values with high probability of correctness to know 39 bits of the NLFSR. The 55 PD^g values are drawn from power consumption data spanning from $t - 2L$ to $t + 2L$ for values of t spanning $L = 80$ bits of the register. Hence, power trace information is required over a span of $5L = 400$ clock cycles.

As studied before, if any of cases I, II, III or IV could be applied to determine a PD^g value, it is correct with the probability of .975. Hence, assuming 55 PD^g values are used to generate the 39 Xor expressions and the correctness of each PD^g is independent, the set of 39 Xor expressions are correct with the probability of $.975^{55} = .248$. In other words, if we apply our analysis using a typical set of power consumption values, our analysis will be successful about 25% of the time.

If we have enough power samples and we could apply our analysis to 16 independent sets of measured power consumption values, with the probability of $1 - (1 - .248)^{16} = .990$, we have at least one successful analysis. The resulting overall complexity of the analysis can be derived by considering the exhaustive search for the 41 Xor expressions not found from the PD^g values for each of the 16 applications of the analysis giving a computational complexity of about $16 \times 2^{41} \approx 2^{45}$ operations, where an operation involves the analysis of the PD^g values for the cases of Table 5.3. In comparison, a cryptanalysis based on exhaustively searching for the proper state of the NLFSR would be expected to take about 2^{80} steps. Hence, significant reduction in the analysis complexity can be achieved by examining the power consumption information and applying simple power analysis.

To decrease the complexity of the analysis, we can expand the cases of I, II, III and IV to include the basic categorization techniques of Section 5.5. For example, we may assume that, if PD^g from the falling edge and rising edge are the same, this PD^g is correct with high probability. Also, if the robust threshold on the rising edge (not falling edge) is applicable, the PD^g may be assumed to be correct with high probability. Now if one of cases I, II, III, or IV or the categorizations based on Sections 5.4.1 or 5.4.2 can be used, the probability of correctness of PD^g values is decreased to .964. However, one of these cases occurs with a probability of .784. Hence, the presented probabilities in Table 5.3 change. For example, the probabilities for scenarios A, B, and C change to .392, .085, and .154, respectively. We have listed the new probabilities in Table 5.4. The summation of the probabilities for all scenarios is now about 80%. Using the new probabilities, we can obtain about $.80 \times 80 \approx 64$ Xor relationships based on about 107 PD^g values (spanning $5L = 400$ clock cycles) which are all correct with an expected probability of about $.964^{107} \approx .02$. This gives an expected success rate for the analysis of only 2%. However, we can increase the probability of success to more than 98%, if we repeat the analysis on 200 independent sets of power trace data. The resulting complexity would be about $200 \times 2^{16} \approx 2^{24}$ operations.

Although, the first approach has higher complexity (2^{45} operations), it requires fewer power samples (i.e., $16 \times (5 \times 80) = 6400$ clock cycles of power samples for the 80-bit NLFSR). However, the second approach, with lower computational complexity (2^{24} operations) needs more power samples (i.e., $200 \times (5 \times 80) = 80000$ samples).

As mentioned earlier, although we have focused our experiments and analysis on an NLFSR, the techniques could be applied to LFSR. However, in the next section

Scenario	PD^g_{t-2L}	PD^g_{t-L}	PD^g_t	PD^g_{t+L}	PD^g_{t+2L}	Probability
A	X	X	±1	X	X	.392
B	X	±1	$\hat{C}$	X	X	.085
C	X	±1	0	X	X	.154
D	±1	0	$\hat{C}$	X	X	.033
E	±1	0	0	X	X	.060
F	$\hat{C}$	$\hat{C}$	0	±1	X	.007
G	0	$\hat{C}$	0	±1	X	.013
H	$\hat{C}$	0	0	±1	X	.013
I	0	0	0	±1	X	.023
J	$\hat{C}$	$\hat{C}$	0	0	±1	.002
K	$\hat{C}$	0	0	0	±1	.005
L	0	$\hat{C}$	0	0	±1	.005
M	0	0	0	0	±1	.009
N	±1	$\hat{C}$	0	0	±1	.005

Table 5.4: Recalculated probabilities for determining $s_t(0) \oplus s_{t-1}(0)$.

we consider an improved analysis technique which is only applicable to LFSRs.

5.7 An Improved Approach to SPA of LFSR Based Stream Ciphers

In the previous section, we have proposed and applied a simple power analysis attack to an NLFSR based stream cipher with inaccurate measured data. Evidently, the proposed method is also applicable to LFSR based stream ciphers. In this section we propose an improved simple power analysis method useful with inaccurate measured data and applicable merely to LFSR based stream ciphers. In the methods studied in the previous sections the complexity of the attack increases significantly with increasing noise. In this dissertation we use the term *noise* for inaccurate measurements or incorrect mapping of measured data and theoretical power difference values. The advantage of the proposed simple power analysis in this section is that the complexity of the attack does not increase with increasing noise. However, increasing noise does increase the required number of power samples.

The proposed cryptanalysis method in this section is based on an algebraic method. Applying algebraic techniques for side channel attack (known as algebraic side channel attack) was at first proposed in [67] and later developed in other research such as [68] and [69]. All the proposed attacks have been applied to block ciphers (AES and PRESENT). AES and PRESENT are designed to be very resistant for algebraic attacks and the obtained system of equations are very difficult to solve (the degree of the system of equation is very high). In [67, 68] and [69] the attacker uses side channel information to provide better equations (with lower degree) and uses them as additional equations to solve the system of equation. Since they assume their system is an error-free system (they assume all the measured power consumptions match the theoretical values of the cryptosystem), the obtained equations from power measurements are linear and reduce the complexity of the classical algebraic attack, significantly.

The main difference of an LFSR and an NLFSR is that in an LFSR there is always a linear relation between the bit values of the state bits at different times, while for NLFSR these relations are nonlinear. In other words, in an LFSR, state bits at time t', $s_{t'}(i)$, $0 \leq i < L$, can be described by a linear relationship of state bits at time t, $s_t(j)$, where $0 \leq j < L$. Hence, if we have a system of equations in which the unknown values are $s_{t'}(i)$, we can transform it to a system of equations in which the unknown variables are $s_t(j)$, where $0 \leq j < L$. Since, the relationships between $s_{t'}(i)$ and $s_t(j)$ are linear in an LFSR, the transform does not change (increase) the degree of the system of equations.

In an ideal circumstance of power analyzing an LFSR, it is assumed that we can exactly determine the theoretical power difference values ($PD \in \{-1, 0, +1\}$), from real power consumption measurements (which are analogue values in the unit of watts). The theoretical PD values are used directly to determine the register bit values of the LFSR. In practice this is somewhat challenging and some power differences are determined incorrectly. Since, it is not clear which PD are categorized correctly and which are categorized incorrectly, direct use of equation (4.1) provides a system of linear equations in which some equations are correct and some are incorrect. This system of equations is studied as *MAX-LIN* in many papers [70, 71, 72]. The complexity of the proposed methods to solve this system of equations are still high (no polynomial complexity has been offered for the *MAX-LIN* problem). [1]

As described in Section 5.3.1, analyzing the experimental results shows the probability of incorrectly categorizing an actual $PD = +1$ to $PD^g = -1$ is negligible. Similarly, the probability of incorrectly categorizing a $PD = -1$ to $PD^g = +1$ is negligible. In other words, virtually all categorization errors occur by incorrectly assigning $PD = \pm 1$ to $PD^g = 0$ or $PD = 0$ to $PD^g = \pm 1$. Hence, for any $PD^g = +1$, we know the actual PD is $+1$ or 0. As well, for any $PD^g = -1$, the actual PD value is -1 or 0.

For an LFSR with bits $s_t(i)$, equation (4.1) can be written as

$$PD_t = g_t - h_t \tag{5.11}$$

where

$$\begin{aligned} g_t &= s_t(L) \oplus s_{t-1}(L) \\ h_t &= s_t(0) \oplus s_{t-1}(0). \end{aligned} \tag{5.12}$$

For any $PD \neq +1$ (or $PD^g = -1$) the possible values for (g, h) are $(0, 0)$, $(0, 1)$ and $(1, 1)$. Similarly, for any $PD \neq -1$ (or $PD^g = +1$) the possible values for (g, h) are $(0, 0)$, $(1, 0)$ and $(1, 1)$. These values are summarized in Table 5.5. From Table 5.5 for any $PD^g_t = +1$ we can write

$$(g_t \oplus 1) \cdot h_t = 0 \tag{5.13}$$

[1]Other proposed methods to solve such a system is include *Artificial Intelligence* algorithms, such as *Genetic Algorithms*. However, our simulation shows using AI is not practical over finite fields and stream ciphers (especially with increasing noise (or decreasing P_{cr})).

$PD_t \neq +1$		$PD_t \neq -1$	
g_t	h_t	g_t	h_t
0	0	0	0
0	1	1	0
1	1	1	1

Table 5.5: Possible values of g_t and h_t, for $PD_t = \pm 1$

and for any $PD_t^g = -1$, we can write

$$(h_t \oplus 1) \cdot g_t = 0. \tag{5.14}$$

Since h and g for any time, can be described with state bits of the LFSR, S_t, for a certain time, t, using linear equations, the degrees of equations (5.13) and (5.14) after substituting h and g with the corresponding relationships using $s_t(i)$ $(0 \leq i < L)$ are still two. Collecting enough $PD^g = \pm 1$ we can set a system of equations with h and g and then using equations (5.12), (5.13) and (5.14) and feedback relation we can convert it to a system of nonlinear equations of degree two, where its unknown variables are $s_t(i)$, where $0 \leq i < L$ for any value of t. Solving this system of equations, recovers the state bits of the LFSR at a particular time t. The systems of equations can be solved using conventional algebraic methods such as *relinearizion* [52] or *XL* [73] and appropriate mathematical tools such as Sage [63]. A similar approach applied to Grain will be discussed in detail in Chapter 7.

5.8 Analyzing an LFSR

The complexity of this attack is the same as the complexity of solving the nonlinear system of equations described above. Since the degree of the system of equations is low, the complexity of solving is not very high. The conventional algebraic methods to solve a system of nonlinear equations (relinearizion and XL) are described in Appendix A. To solve a system of nonlinear equations, we should convert it to a system of linear equations. In the new linear system of equations, any products of unknown variables of the primary or nonlinear system is assumed as a new variable in the secondary system of equations. For example, if in a primary system of equations, $s_t(i)$ and $s_t(j)$ (where $0 \leq i, j < L$) are two unknown variables, in the secondary system of equation, $s_t(i) \cdot s_t(j) = x_{ij}$ is assumed to be an unknown variable. Hence, in our cryptanalysis method, the maximum number of unknown variables to attack an LFSR with the size of L bits is $\binom{L}{2} + L$. Letting $L = 80$, the number of unknown variables, U, is

$$U = \binom{80}{2} + 80 = 3240 \approx 2^{11.66}. \tag{5.15}$$

After converting the system of nonlinear equations to a system of linear equations, we can use the conventional algorithms such as Gaussian elimination to solve it. The complexity of using Gaussian elimination to solve a linear system of equations with U unknown variables is U^3 [74]. Then, the complexity of the attack for an 80 bit LFSR is 2^{35}. A more conventional algorithm to solve a system of linear equations is the proposed algorithm in [75]. Using the proposed method in [75], the complexity of solving the system of equations is at most $7 \times (2^U)^{log_2 7}$. For an 80 bit LFSR this complexity is $7 \times (2^{11.66})^{log_2 7} = 2^{36}$. Hence, the total complexity of cryptanalyzing an 80 bit LFSR with the proposed method is 2^{35}.

If we use XL method to solve a system of nonlinear equations with 80 unknown variables, we need more equations to confirm we can generate enough linearly independent equations. Assume from 100 equations, with high probability we can generate enough linearly independent equations in the secondary system of equations. This assumption is consistent with similar assumptions in [53, 73]. Since, each $PD^g = +1$ or $PD^g = -1$ generates an equation for the primary system of equations, the expected number of required power samples to have 100 equations is $2 \times 100 = 200$ power samples (assuming the probability of categorizing $PD^g = +1$ or $PD^g = -1$ is .5).

If we have N power samples, the probability of obtaining at least U' useful power samples where $PD^g = \pm 1$ (or we call it *target PD values*) is

$$P_s = \sum_{i=U'}^{N} \binom{N}{i} (\frac{1}{2})^N. \tag{5.16}$$

For example, if we have collected 300 power samples, with the probability of greater than 99.99%, we have more than 100 target PD^g values. Note that $PD^g = 0$ is not a target value and such cases are ignored.

Here, we assumed the the probability of categorizing actual $PD = +1$ to -1 and $PD = -1$ to $+1$ is negligible. With increasing the noise (or the rate of incorrect categorization), it is possible that will not be the case. In that case, we can use other proposed techniques in Section 5.4 to choose target PDs. Using the other PD values does not change the degree of the system of equations, however more power samples will be needed to set up the system of equations. In that scenario, the probability of success (P_s) in equation (5.16), is changed to

$$P_s = \sum_{i=U'}^{N} \binom{N}{i} (p)^i (1-p)^{N-i}, \tag{5.17}$$

where p is the probability of categorization of a target PD value. For example, with robust threshold method, it should be assumed that if $PD_t^{rg} = +1$, then the actual PD_t is not equal to -1 ($PD_t \neq -1$). As well for $PD_t^{rg} = -1$ it should be assumed the actual PD_t is not equal to $+1$ ($PD_t \neq +1$). In this case the expected number of power samples is $4 \times 100 = 400$, assuming the probability of occurrence of $PD_t^{rg} = +1$ or $PD_t^{rg} = -1$ is .25, and if we have 600 power samples, the probability of collecting at least 100 target PD values is greater than 99.99%.

5.9 Summary

In this chapter, we have proposed a simple power analysis of NLFSR, a component typically found in stream ciphers, in a non-ideal environment where the measured difference is not perfectly categorized. Also, we consider power consumption of a typical CMOS D flip-flop and propose use of power samples at the falling (or non-triggering) edge of the clock for the analysis. Furthermore, we applied the analysis to an 80-bit NLFSR using practical simulated power trace data for a 180 nm CMOS circuit. We have shown that if we use falling edge and rising edge power consumption information and the proposed techniques in this paper, we can successfully analyze with high probability the NLFSR with time complexity of about 2^{45} operations using about 6400 power samples or 2^{24} using about 80000 power samples. This is significantly less than the complexity of 2^{80} for exhaustive search for the NLFSR state. It should be noted that the techniques applied to the 80-bit NLFSR in this paper, apply equally to an LFSR. The proposed method can be improved by using Error Correction Code or ECC techniques. However the complexity of the attack or limit of computation will be same.

Furthermore, we have studied a simple power analysis of LFSR, which is applicable on stream ciphers with inaccurate measurements. We have shown that the proposed method is applicable to an 80 bit LFSR, with the timing complexity of 2^{35} and only a few hundred power samples.

These results indicate that practical implementations of stream ciphers based on either LFSRs and/or NLFSRs are vulnerable to side channel analysis attacks even when ideal assumptions are not applicable and care must be taken to design implementations which do not leak power consumption information.

Chapter 6

Using Simple Power Analysis for Correlation Attack

The *correlation attack* is a mathematical method of cryptanalyzing FSR-based stream ciphers proposed in [59]. It is a divide-and-conquer technique, applicable to stream ciphers constructed with multiple LFSR/NLFSRs. Modern stream ciphers are designed in a way to be invulnerable against it and correlation attack is not applicable to recent stream ciphers. In this section, a developed correlation attack to be applicable in side channel cryptanalysis is described. This crypanalyzing method is applicable to stream ciphers constructed with multiple LFSRs and/or NLFSRs. The results presented in this chapter may be found in [76].

6.1 Preliminaries: Correlation Attack

A correlation attack is a known-plaintext attack, based on finding a correlation between the state bits of an individual FSR in the stream cipher with multiple FSRs

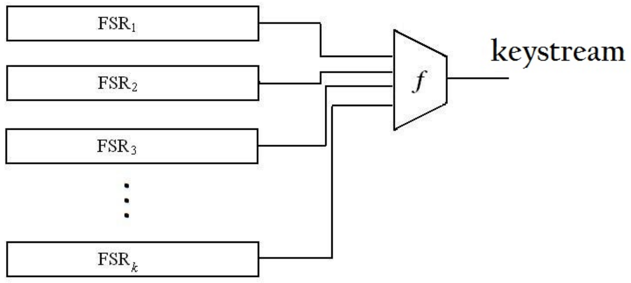


Figure 6.1: Nonlinear combination generator

and the output bits or keystream bits of the cipher. Assume a keystream generator or a stream cipher containing k FSRs where the output is given by a nonlinear function of the FSRs' bits (Figure 6.1). This is a classical stream cipher construction and referred to as a nonlinear combination generator. A generic attack which always applies to this type of cipher is the *exhaustive search* for the initial states of the FSRs. For each possible initial configuration, the generated keystream should be calculated and the correct initial state is deduced when the produced sequence by the combining function is the same as the observed keystream. Such a generic attack requires at least $2^{\sum L_i}$ trials, where L_i is the bit size of the i-th FSR. In typical stream ciphers, this time complexity is too large, making this attack infeasible.

Let $s_t^i(j)$ denote the j-th bit of i-th FSR at time t and z_t denote the output keystream at the same time. Since the output keystream is a function of FSR bits, we can write $z_t = f(s_t^1(0), s_t^2(0), \cdots, s_t^k(0))$. To produce an unbiased sequence, f must be a balanced function, i.e., produce 0 and 1 with the same probability of $\frac{1}{2}$.

The main aim of correlation attack is to find the state bits of the stream cipher. The probability of $z_t = s_t^i(0)$ is denoted with $P(z_t = s_t^i(0))$ and is equal $\frac{1}{2} + \epsilon_i$. Large values of $|\epsilon_i|$ make the stream cipher more vulnerable to the correlation attack.

Assume that a segment of N keystream bits is being observed by an attacker. In a correlation attack, the attacker does an exhaustive search over one FSR's state bits. For each guess he should calculate N LFSR output bits, i.e., $s_t^i(0)$ for different t. Then, he should compare the observed z_t and calculated $s_t^i(0)$ to check the rate of the equality of z_t and $s_t^i(0)$. If N is big enough, for the correct guess, this rate is close to the probability of equality of z_t and $s_t^i(0)$, $P(z_t = s_t^i(0))$. Since f is a balanced function, for incorrect guesses this rate is close to $\frac{1}{2}$. This process should be done for each LFSR and NLFSR of the stream cipher. The complexity of recovering the state bits of FSR i is 2^{L_i} operations, where an operation consists of the analysis of the N keystream bits. Hence, the complexity of recovering all state bits of the stream cipher is $\sum_{i=1}^{k} 2^{L_i}$ and the complexity of an attack to recover some state bits is $2^{L_{min}}$, where $L_{min} = \min\{L_1, L_2, ..., L_k\}$. This complexity is much smaller than the complexity of a *brute-force* attack or exhaustive search, which is $2^{L_1+L_2+..+L_k}$. The required number of samples in correlation attack depends on ϵ_{min} and is given in [77] to be

$$N \in \Omega\left(\epsilon_{min}^{-2}\right) \tag{6.1}$$

or

$$N \propto \epsilon_{min}^{-2}. \tag{6.2}$$

Therefore, an attack of a stream cipher needs a time of [77]

$$T \in O\left(2^{L_{max}} \times \epsilon_{min}^{-2}\right). \tag{6.3}$$

where L_{max} is the bit size of largest FSR and each operation consists of an examination of the equality of the keystream bit and the output of FSR.

6.2 Categorization of Measured Power Difference Values

To apply SPA to an LFSR/NLFSR based on the measured power consumption of the LFSR/NLFSR, we should map the measured power differences (an analog variable measured in watts) to a categorized power difference, represented as PD^g where $PD^g \in \{-1, 0, +1\}$ for an individual LFSR or NLFSR. In an ideal circumstance the attacker would determine PD^g to be the theoretical PD value with 100% likelihood (as is assumed in Chapters 3 and 4). In practical applications, the power differences measured on clock edges cannot be mapped perfectly to the theoretical power differences due to factors such as power consumption sources that are not flip-flops (eg., the combinational logic in the feedback and output functions), clock skew, and static power consumption. In most attacks based on power analysis, to overcome these types of inaccuracies, differential power analysis is used where the power consumption measurements are repeated and the average of the observed data is used to filter out noise. However, for stream ciphers, such as Grain, this approach is limited in applicability and can only be applied by making use of power traces from numerous resynchronizations [46, 78]. The correlation attack based on side channel analysis developed in this chapter is applicable even with high rate of error in the mapping of measured power differences to cipher data.

As described in Chapter 5, if we assume that the bits generated by the LFSR and/or NLFSR are random such that the probability of 0 and 1 are both equally likely, then the probabilities of $PD = -1$, $PD = 0$ and $PD = +1$ for each LFSR/NLFSR are .25, .5 and .25, respectively. In order to map the measured power differences to a PD^g value, we can sort all the measured power differences and then categorize the largest positive 25% of them to $PD^g = +1$. Similarly, we can categorize the smallest (i.e., most negative) 25% of the measured power differences to $PD^g = -1$. The rest of the measured power differences should be categorized to $PD^g = 0$. This method of categorization can be applied to a system with one LFSR or NLFSR and was studied in Section 5.3.

In order to categorize the measured power differences of stream ciphers with two FSRs such as Grain, we can use a similar method. For the Hamming distance power model, we know that the overall power consumption of stream ciphers with multiple FSRs, such as Grain, is approximated by the summation of power consumption of the LFSRs and the NLFSRs (assuming power consumed in other parts of the circuit is negligible) and measuring the power at the triggering edge of the clock embodies the power consumption of the D flip-flops of the LFSRs and NLFSRs. For Grain, if we assume the power consumption of the circuit at time t (at the triggering edge) is the summation of the power consumption of the LFSR and NLFSR (which is also proportional to the Hamming distance of their consecutive states), then we can conclude the overall dynamic power dissipation of the circuit at the triggering edge of the clock is proportional to $HD_t^{LFSR} + HD_t^{NLFSR}$. Hence, we can define the theoretical power difference of the circuit as

$$\begin{aligned} PD_t^{Grain} &= [HD_{t+1}^{LFSR} + HD_{t+1}^{NLFSR}] \\ &\quad -[HD_t^{LFSR} + HD_t^{NLFSR}] \\ &= PD_t^{LFSR} + PD_t^{NLFSR}. \end{aligned} \tag{6.4}$$

Assuming that the probability of generating 0 and 1 in the LFSR and NLFSR are both equal to $\frac{1}{2}$, based on (6.4), the probabilities of occurrence of possible PD values are easily calculated and listed in Table 6.1. For example, if PD of Grain is +2, both PD values of LFSR and NLFSR should be +1. This will happen with the probability of .25 for each of them. Then, the probability of $PD = +2$ for Grain is $\frac{1}{4} \times \frac{1}{4} = .0625$.

Similar to the proposed method for one FSR systems, to categorize measured PD values of Grain, we sort the measured power differences. Then the 6.25% largest positive measured power differences should be categorized as $PD^g = +2$ and the 6.25% most negative measured power differences should be categorized as $PD^g = -2$. Correspondingly, the next 25% of the most positive and negative measured power differences should be categorized as $PD^g = +1$ and $PD^g = -1$, respectively. The remaining values are categorized as $PD^g = 0$.

6.3 Practical Categorization for Grain

Here, we investigate the practical issues by simulating circuits in CMOS. The circuits are prototyped in TSMC 180 nm standard cell CMOS technology. We have used Cadence Virtuoso Spectre Circuit Simulator version 5.10.41 to obtain the power con-

Table 6.1: Probability of occurrence of PD values in stream ciphers with two LFSR/NLFSRs such as Grain.

PD value	Probability
-2	$\frac{1}{16} = .0625$
-1	$\frac{1}{4} = .25$
0	$\frac{3}{8} = .375$
$+1$	$\frac{1}{4} = .25$
$+2$	$\frac{1}{16} = .0625$

sumption of the circuits. The power supply of all circuits is 1.8 volts. The simulations have been done at room temperature and default noise. In our implementations, we have used the classical D flip-flop shown in Figure 5.1 (a).

Obtaining experimental results from a CMOS implementation of Grain, we have collected 11700 power samples simulated through the Cadence tools. Applying the categorization method of Section 6.2, we found that experimentally only 46.56% of measured power differences were categorized correctly. Although this value is low (that is, well below 100%), it is significantly different than the 27.34% probability of correctly categorizing the measured power differences by random. This is calculated as follows. Let $P(PD^g = i)$ represent the probability of a measured PD value to be categorized (randomly) as i, $P(PD = i)$ represent the probability that the theoretical PD is equal to i, and $P(PD^g = i|PD = i)$ represent the conditional probability. The

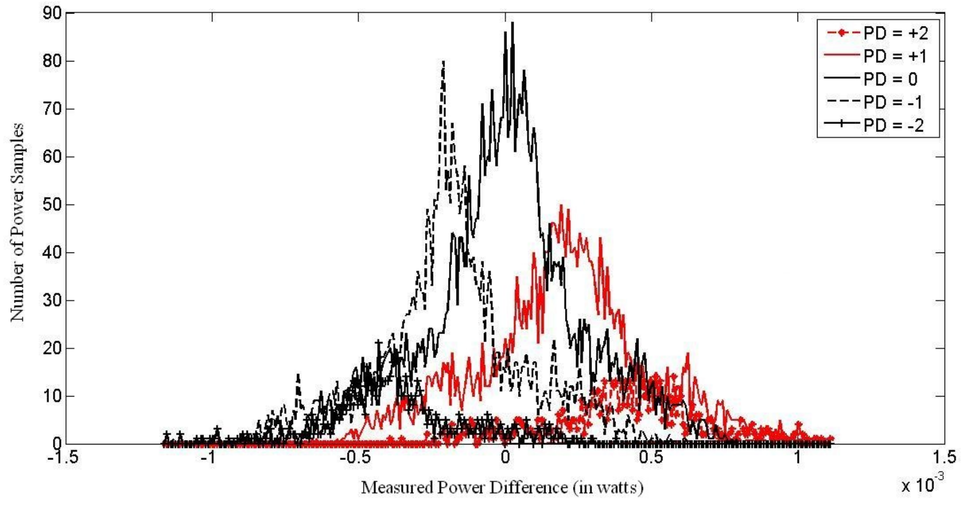


Figure 6.2: Measured power difference categorization results for Grain given theoretical PD values.

probability of correctness is given by

$$\begin{aligned}
P_{corr} &= \sum_{i=\{-2,\ldots,+2\}} P(PD^g = i | PD = i) \cdot P(PD = i) \\
&= \sum_{i=\{-2,\ldots,+2\}} P(PD^g = i) \cdot P(PD = i) \\
&= (\frac{1}{16})^2 + (\frac{1}{4})^2 + (\frac{3}{8})^2 + (\frac{1}{4})^2 + (\frac{1}{16})^2 \\
&= .2734,
\end{aligned}$$

where $P(PD = i) = P(PD^g = i)$ which can be obtained from Table 6.1 and where the conditional probability becomes unconditional due to the randomness assumption of the categorization.

To further illustrate the nature of incorrect categorization, we present the following discussion. Based on our experimental results, in Figure 6.2, we have presented a graph which shows measured power differences and their corresponding theoretical PD values. As shown in the graph, there are many ranges of measured power differences where there are power samples corresponding to multiple theoretical PD values. For example, there are a total of 646 power samples that have measured power differences falling in the range from 0 to $+4.5 \times 10^{-5}$. Of these, 400 have theoretical values of $PD = 0$ and 130 have theoretical values of $PD = +1$. (The remaining 116 power samples have other PD values.) Since the measured power differences in this range are close to zero, we might reasonably expect that the correct categorization would be $PD^g = 0$, but to categorize in this way, will clearly incorrectly categorize a number of values for which $PD = +1$ (as well, as other PD values). Clearly, any proposed method to attack stream ciphers based on simple power analysis should mitigate the problem of incorrect categorization.

6.4 Divide-and-Conquer Method

The theoretical attacks on stream ciphers presented in Chapter 3 and Chapter 4, assume that power measurements can be used to determine with certainty the theoretical power difference, PD, and thereby relate power measurements to the cipher data. In a practical attack, this expectation is naive, since there are many factors which make the mapping between the measured power difference and theoretical PD inaccurate.

This issue is discussed thoroughly in Chapter 5, which examines the implications of simple power analysis applied to a simulated CMOS implementation of an individual LFSR and NLFSR. However, since the techniques of Chapter 5 focus on extracting information from only one FSR (LFSR or NLFSR), it is not immediately practical to apply these to ciphers using multiple LFSRs/NLFSRs such as Grain.

6.4.1 Using Power Measurements in a Brute Force Attack

Assume a stream cipher includes k FSRs and a nonlinear combining function. We assume the overall power consumption of the circuit is determined by the summation of the power consumption of each LFSR/NLFSR. Hence, the overall measured power difference of the circuit is the summation of measured power differences of each individual FSR. Consequently, the possible PD values for the complete circuit satisfy $-k \leq PD \leq k$.

Let the summation of the size of all FSRs be L_{tot} so that $L_{tot} = L_1 + L_2 + ... + L_k$, where L_i is the size of FSR i. In a brute-force attack, we try all possible bit configurations for the LFSRs and NLFSRs (i.e., all values for L_{tot} bits). Then we should check whether our guess is correct, typically using known plaintext/ciphertext data with the resulting complexity of this method being $2^{L_{tot}}$. However, as we shall explain, power consumption measurements can be used in place of the plaintext/ciphertext data used in the check of each guess.

Assume that we have collected N power samples and categorized the measured power differences. In applying brute force based on simple power analysis, after each guess, we should calculate PD values of the cipher based on the guessed bit values of the FSRs for N clock cycles and compare with the categorized measured power differences. In an ideal environment (where there would be no mapping errors from measured PD to PD^g), if the categorized values match the PD values corresponding to the guess, we know our guess for the bits is correct. In a practical implementation, we will have some number of errors during categorization of measured power differences. If the number of errors between guessed PD values and categorized measured power differences, PD^g, is large (as would be expected for random mappings), we can conclude that some number of the guessed bit values for the cipher FSRs are wrong.

Based on our experiments on Grain, when the guess is correct, the probability of categorized measured power differences being correct should be .4656 and, hence, PD^g values are in error with a probability of .5344. In comparison, PD^g values for incorrect guesses, should have the probability of error of about $1 - .2734 = .7266$. This difference can be used to accurately distinguish the correct guess of FSR bits from incorrect guesses, assuming that N is large enough.

Other approaches can also be used to distinguish the correct guess. Consider Table 6.2 which lists the probabilities for different combinations of categorized measured power differences and PD. The table shows that, for some PD values, some measured power differences will not happen or have very low probabilities of occurrence. For example, when the theoretical $PD = -2$, the probability of categorized measured power differences being $+2$ was found to be 0 and, for theoretical power differences of $+2$, the probability of $PD^g = -2$ was also found to be 0. These (virtually) impossible mappings could help us to distinguish incorrect guesses since they will (virtually) never occur with a correct guess but may occur with an incorrect guess.

In another approach, using the provided probabilities of the last two columns in Table 6.2, we can compare the probability of correctness for specific PD values. For example, if, for all $PD = +2$ based on our guess, the probability of the corresponding $PD^g = +2$ is close to .0625, we can conclude that our guess is incorrect and, if that probability is close to .3076, we can conclude the guess is correct.

Table 6.2: Experimental probability of measured power difference categorization in Grain used for brute force attack.

Theoretical PD^{cipher} (x)	$PD^{g,cipher}$ of cipher (y)	Probability of $PD^{g,cipher} = y$ given theoretical $PD^{cipher} = x$	Probability of $PD^{g,cipher} = y$
$+2$	$+2$	.3076	.0625
$+2$	$+1$	.5211	.250
$+2$	0	.1475	.375
$+2$	-1	.0239	.250
$+2$	-2	0	.0625
$+1$	$+2$	.1196	.0625
$+1$	$+1$	.4553	.250
$+1$	0	.2987	.375
$+1$	-1	.1196	.250
$+1$	-2	.0057	.0625
0	$+2$	.0342	.0625
0	$+1$	.2010	.250
...	...	...	...
-1	-1	.4610	.250
...	...	...	...
-2	$+2$	0	.0625
...	...	...	...
-2	-2	.2649	.0625

The complexity of applying this method of a brute-force attack to Grain is 2^{160} since we expect that we will need to try guesses for all possible values of the LFSR and the NLFSR. The high complexity of this attack makes it impractical for serious cyptanalytic threat to a stream cipher. However, in the next section we propose a method to reduce the complexity of the attack, thereby potentially compromising the security of practical stream cipher implementations.

6.4.2 SPA Attack Using Divide-and-Conquer on Stream Ciphers with Multiple FSRs

To decrease the complexity of the attack we can apply a divide-and-conquer approach by guessing each LFSR/NLFSR independently and then checking whether the guessed bits for the FSR are correct or not. Note that this divide-and-conquer approach is similar to classical correlation attacks applied to nonlinear combination keystream generators [59].

6.4.2.1 General Attack

One approach to attacking an FSR-based stream cipher, using a divide-and-conquer approach, takes advantage of the impossibility of certain PD^g values for the cipher occurring given one of the FSR PD values. Table 6.3 illustrates such error scenarios for different numbers, k, of FSRs in the system, when all PD^g values matched with the actual values of PD with the probability of 100%. For example, for $k = 2$, if one FSR is assumed to have values of $PD = +1$ at various points in time, then if this assumption is correct, no value of $PD^g = -2$ or -1 for the overall cipher must

Table 6.3: Error scenarios for categorized PD

# of FSRs	Guessed PD^{FSR} (one FSR)	Impossible $PD^{g,cipher}$ (overall cipher)
2	$+1$	-2, -1
	0	-2, $+2$
	-1	$+1$, $+2$
3	$+1$	-3, -2
	0	-3, $+3$
	-1	$+2$, $+3$
4	$+1$	-4, -3
	0	-4, $+4$
	-1	$+3$, $+4$
k	$+1$	$-k$, $-(k-1)$
	0	$-k$, $+k$
	-1	$+(k-1)$, $+k$

occur at any of these points in time. If it does, it is reasonable to assume that the assumption is incorrect and the assigned PD^g value to the FSR is incorrect. (Of course, in practice, inaccuracies in categorization of measured power differences may make there be a small non-zero probability that such an error will occur.)

In a divide-and-conquer approach, we can guess the bit values of one FSR, determine the corresponding PD values for the FSR and examine whether impossible PD^g values occur. From this we can determine the error probability as the fraction of impossible PD^g values of the cipher (given by the 3rd column of the Table 6.3).

From this error probability, we can determine whether our guess for the bits of the FSR was correct. For example, for $k = 2$, for all FSR values with $PD = +1$ based on the guess of the FSR bits, the error probability for the cipher $PD^g = -2$ is expected to be about 0.0625 if the categorization is random relative to the PD values based on the guessed FSR bits (that is, corresponding to an incorrect guess of the FSR bits), while the error probability should in theory be 0 (or, in practice, very close to 0) when the correct bits of the FSR are guessed.

Using these concepts, an algorithm to attack a stream cipher with multiple FSRs is proposed as follows:

1. Measure the power consumption of the circuit and calculate a number, N, of measured power difference (MPD) values.
2. Categorize the measured power differences of the circuit to obtain PD^g values.
3. For each of the k FSRs of the cipher, repeat the following until the correct guess is determined.
    (a) Try a new guess of the bit values of the FSR.
    (b) Calculate N consecutive theoretical PD values of the FSR based on the guessed bits.
    (c) Compute the error probability based on the calculated PD values of the FSR and the PD^g values of whole circuit.
    (d) Compare the computed error probability with the probability of categorizing measured power differences randomly. If the probability is close to

the probability of randomly categorized values, we can conclude the guess is incorrect; if the probability is 0 (or very close to it), the guess is correct.

4. Return the correct guess for each FSR.

The computational complexity of this algorithm is determined by the need to systematically guess bit values for each FSR individually. This will take the longest for the largest FSR and therefore, the complexity is $2^{L_1}+2^{L_2}+...+2^{L_k} < k2^{L_{max}}$ power trace analyses, where L_{max} is the bit size of the largest FSR in the cipher. A power trace analysis consists of the assessment of the N power differences for each guess of an FSR value and is therefore comprised of the steps (a)-(d) of step 3 in the algorithm. Example stream ciphers which are vulnerable to divide-and-conquer attack are the Geffe generator [62] and E0 [6] and other stream ciphers based on combining FSRs. In the next section, we apply this divide-and-conquer attack approach to Grain in a practical environment where inaccurate categorizations of measured power differences can occur.

6.4.2.2 Applying the Attack to Grain

In this section, we examine the experimental results of applying a divide-and-conquer approach to a practical implementation of Grain. Having obtained experimental data for an implementation of Grain, we outline a slightly different approach to checking our guess for the FSR bits than is used in the previous section, where the impossible scenarios of Table 6.3 were used to distinguish correct and incorrect guesses. More generally, we can use the probability of any or all PD^g values of the cipher given PD values of one FSR, not just the impossible scenarios, where this probability is zero.

Grain consists of an LFSR and an NLFSR (i.e., $k = 2$). In Table 6.4, we present the probability that, when a PD value of one of Grain's FSRs is x, the categorized power difference for the whole Grain circuit takes on a value of y. The provided data in the table is based on 11700 power samples of Grain determined from simulations for 180 nm CMOS. Note that the first column of Table 6.4 represents the PD value associated with only one FSR; this is different than the data in Table 6.2 whose first column represents the PD value of the entire cipher. Also, note that, since these results now reflect practical power measurements, the error scenarios given in Table 6.3 are not impossible to observe since practical inaccuracies occur during the categorization process. However, the error scenarios occur with very small probabilities, eg. the probability of observing $PD^{g,cipher} = +2$ given the $PD^{FSR} = -1$ is only 1.05%, well below the 6.25% of values expected to be $PD^{cipher} = +2$.

After collecting power samples from the circuit, we have to systematically guess the bit values of the FSR. Based on the guessed bits, we should determine guessed theoretical PD values of one FSR and compare them with the categorized power difference, PD^g, values of the overall cipher determined from the measured power consumption. If the resulting experimentally determined conditional probabilities are similar to the fifth column of Table 6.4, we know that our guess for the bit values of the FSR is incorrect, but, if the probabilities are close to the third column, we can assume the guessed bits are correct. The fourth column, representing the theoretical conditional probabilities (i.e., where no mapping errors occur in categorization), is presented for comparison. In order to complete the attack, the cryptanalyst must individually exhaustively try all values for both the NLFSR and the LFSR, which will take, at most, $2^{80} + 2^{80} = 2^{81}$ analyses of the available power samples.

Table 6.4: Probability of measured power difference categorization in Grain used for divide-and-conquer attack.

PD^{FSR} (x)	$PD^{g,cipher}$ (y)	$P(PD^{g,cipher} = y \| PD^{FSR} = x)$ (experimental)	$P(PD^{g,cipher} = y \| PD^{FSR} = x)$ (theoretical)	$P(PD^{g,cipher} = y)$ (theoretical)
+1	+2	.1440	.250	.0625
+1	+1	.4075	.500	.250
+1	0	.3216	.250	.375
+1	−1	.1111	0	.250
+1	−2	.0158	0	.0625
0	+2	.0475	0	.0625
0	+1	.2407	.250	.250
0	0	.4215	.500	.375
0	−1	.2398	.250	.250
0	−2	.0505	0	.0625
−1	+2	.0105	0	.0625
−1	+1	.1096	0	.250
−1	0	.3352	.250	.375
−1	−1	.4109	.500	.250
−1	−2	.1338	.250	.0625

In order to succeed, the attack needs enough power samples, N, to accurately calculate probabilities and the probability of success in the attack increases by increasing the number of collected measured power differences. In the next section we study the relationship between the probability of success and the required number of power samples.

Given the complexity of the attack, the proposed attack is more effective than a brute force approach at guessing the 160 bits of the keystream generator state. However, in Grain the unknown state is initialized through the combination of an 80-bit key and a known 64-bit IV. Hence, the proposed attack does not improve on the complexity of an exhaustive key search attack on Grain. Since the key size of Grain is only 80 bits, it is reasonable to expect that Grain can be attacked for a publicly known IV value, by trying 2^{80} different guesses for the key and verifying the correct guess by matching a couple hundred bits of actual keystream to the keystream generated based on the guessed bits. The results on Grain are therefore for illustration purposes only and circumstances for which the attack will be more successful than other approaches are discussed in Section 6.4.1.

6.4.2.3 Analysis of Attack on Grain

In order to determine the practical applicability of the attack on Grain, it is necessary to determine the number of required power samples, N. In order to analyze this, consider the case presented in the Table 6.4 corresponding to a guessed theoretical PD value of an FSR of +1, resulting in the overall categorized PD^g value for the cipher of +1. Experimental analysis shown in the table indicates that, if our guess for the bits of the FSR is correct, the probability of the cipher PD^g value being +1 when the FSR $PD = +1$ is about .4 (experimentally, .4075 and, theoretically, .5), while, if the guess for the FSR bits is incorrect, the probability is .25. Similarly, for cases where the guessed bits of the FSR imply a PD value of -1 for the FSR, the cipher PD^g value should be -1 with a probability of about .4 (experimentally, .4109 and, theoretically, .5) if the guess is correct, with $PD^g = -1$ with probability of .25 when the guess is incorrect. Since for an FSR, on average, 50% of the PD values are $+1$ or -1, we can make use of about half of the measured power samples and the resulting PD^g to verify these two conditional probabilities.

For simplicity, assume that out of N power samples, exactly half correspond to cases of $PD = +1$ or $PD = -1$ for the guessed bits of the targeted FSR in the divide-and-conquer attack. (The exact number is a random variable following the binomial distribution with a mean of $\frac{N}{2}$, but variations in this number do not make a significant difference to the analysis, so the assumption is reasonable.) Let M represent the number of guessed PD values of $+1$ or -1 for which the cipher PD^g value is the same. The probability of occurrence for a particular value of M is given by the binomial distribution:

$$P_{eq}(M) = \binom{\frac{N}{2}}{M} \times \Lambda^M \times (1-\Lambda)^{\frac{N}{2}-M}, \tag{6.5}$$

where Λ represents the conditional probability of PD^g for the overall cipher given the guessed PD value of the FSR and $\Lambda = .4$ when the FSR guess is correct and $\Lambda = .25$ when the FSR guess is incorrect.

For the correct guess, the number of occurrences for which the guessed FSR PD is the same as the cipher's PD^g should be, on average, $.4 \times \frac{N}{2}$. For the incorrect guess, the number of such occurrences will average $.25 \times \frac{N}{2}$. So, in order to distinguish between a correct guess for the FSR bits and an incorrect guess, we can set a threshold, γ, such that, if the value of M is greater than or equal to the threshold, it is assumed that the correct guess has been made. Otherwise, the incorrect guess has been made. In this case, the probability of correctly distinguishing the correct guess is given by calculating the probability that $M \geq \gamma$ for the probability distribution associated with the correct guess of the FSR value. We can refer to this as a case of a *true accept* and its probability is given by

$$P_{ta} = \sum_{M=\gamma}^{\frac{N}{2}} \binom{\frac{N}{2}}{M} \times (.4)^M \times (.6)^{\frac{N}{2}-M}, \tag{6.6}$$

which is simply derived from (6.5) with the value $\Lambda = .4$. The probability of erroneously accepting an incorrect guess as correct is given by calculating the probability that $M \geq \gamma$ for the probability distribution associated with the incorrect guess of the FSR value. We can refer to this as a case of *false accept* and its probability is given by

$$P_{fa} = \sum_{M=\gamma}^{\frac{N}{2}} \binom{\frac{N}{2}}{M} \times (.25)^M \times (.75)^{\frac{N}{2}-M}, \tag{6.7}$$

which is simply derived from (6.5) with the value $\Lambda = .25$.

In order for the attack on Grain to be a success, we can set the reasonable requirements of $P_{ta} = 99\%$ and $P_{fa} < 2^{-80}$. Setting these requirements will result in a very high likelihood that, after testing all 2^{80} guesses of the FSR value, the only guess distinguished as being correct is indeed the correct value of the FSR state. We can then use (6.6) and (6.7) to determine the number of power samples, N, to satisfy these requirements. We can do so most easily by approximating the binomial distributions implied by (6.5) as a Gaussian distribution, where for an incorrect FSR guess, the mean and variance are given by:

$$\begin{aligned} \mu_0 &= .25 \cdot n/2 \\ \sigma_0^2 &= (\tfrac{N}{2}) \times .25 \times (1 - .25) = .1875 \times \tfrac{N}{2}. \end{aligned} \tag{6.8}$$

For the correct FSR guess, the mean and variance of the Gaussian approximation are given by:

$$\begin{aligned} \mu_1 &= .4 \cdot \tfrac{N}{2} \\ \sigma_1^2 &= (\tfrac{N}{2}) \times .4 \times (1 - .4) = .24 \times \tfrac{N}{2}. \end{aligned} \tag{6.9}$$

Based on the constraints on P_{ta} and P_{fa}, the threshold must be set to at least $10.22\sigma_0$ above μ_0 and at least $2.33\sigma_1$ below μ_1, resulting in the following constraint:

$$10.22\sigma_0 + 2.33\sigma_1 \leq \mu_1 - \mu_0 \tag{6.10}$$

Substituting (6.8) and (6.9) into (6.10) results in $N \geq 2750$. Hence, we must use about 3000 power samples from the cipher to determine 3000 values of PD^g which may be used to verify which guess of the FSR state is correct. Hence, we may conclude that using 2^{81} tests on about 3000 power samples is sufficient to attack the cipher, which is substantially less effort than a brute force search on the full Grain state of 160 bits.

Note that we have only used the probabilities of having $PD^g_{cipher} = +1$ or -1, conditioned on $PD_{FSR} = +1$ or -1, respectively, and it would also be possible to make use of all cases in Table 6.4 to possibly reduce the number of required power samples further. Using a chi-square goodness-of-fit test, one could analyze all cases to determine whether the experimental conditional probabilities of column 3 corresponding to a correct guess of the FSR is a better fit than the probabilities of column 5 associated with an incorrect guess of the FSR.

6.4.3 General Applicability of the Attack to Multiple FSR Ciphers

As discussed in Section 6.4.2.2, the effectiveness of the attack on Grain is limited by the fact that the size of key is the same as the register sizes used in the cipher structure. Hence, a divide-and-conquer approach that targets guessing one register at a time does not improve on exhaustively searching through the key. We have focused on Grain in this thesis to illustrate the concepts on an existing cipher proposal. However, this attack has broad applicability to any stream cipher constructed using multiple FSRs and in many cases, the divide-and-conquer approach may indeed prove to be substantially more effective than either a brute force search through all possible keystream generator states or an exhaustive key search.

For example, consider the applicability to a nonlinear combination generator comprised of k FSRs (which can be either LFSRs or NLFSRs), where the sizes of the registers are L_1, L_2, ..., L_k. The total number of state bits in the keystream generator is given by $L_{tot} = L_1 + L_2 + ... + L_k$. In this case, a brute force attack on the state of the generator would require a complexity of $2^{L_{tot}}$ and a basic time-memory tradoff attack [19] would require a complexity of $2^{L_{tot}/2}$ in time and memory. However, an SPA-based divide-and-conquer attack would require a time complexity of $2^{L_{min}}$ and, perhaps, a few hundred power samples, where $L_{min} = \min\{L_1, L_2, ..., L_k\}$ to recover some of the state bits. For $L_{min} \ll L_{tot}$, this implies that the SPA attack represents a significant improvement over brute force and time/memory tradeoff attacks. Also, if L_{min} is significantly smaller than the size of the key, then SPA is potentially significantly more successful than exhaustive key search. In the case of Grain, $L_1 = L_2 = 80$,

$L_{min} = 80$, and $L_{tot} = 160$. Hence, the SPA attack is a significant improvement over brute force search of the state space and improvement on a time-memory tradeoff, which requires large memory resources. However, since the key size is equal to L_{min}, there is no improvement over exhaustive key search.

6.5 Summary

In this chapter we have proposed a simple power analysis attack, applicable to stream ciphers based on multiple LFSRs and/or NLFSRs. The proposed method is based on divide-and-conquer approach and is similar to correlation attack in the mathematical cryptanalysis of stream ciphers. The practical implications of the attack are demonstrated by examining simulated power trace measurements from a CMOS implementation of the Grain stream cipher. These results show that a large category of FSR based stream ciphers are practically vulnerable to simple power analysis and care must be taken to design implementations which do not leak power consumption information.

Chapter 7

Using Fast Correlation Attack for Simple Power Analysis

As described in Chapter 6, Siegenthaler [59] shows that if there exists a measure of correlation between the keystream sequence and the output of the FSRs in an stream cipher, it is possible to determine the initial state of each FSR, independently. Thereby he reduced the cryptanalytic attack to a divide and conquer attack with approximate complexity of $\sum_i 2^{L_i}$. The correlation attack significantly reduces the complexity of the cryptanalysis of stream ciphers but it requires an exhaustive search over entire states of each FSR. To avoid an exhaustive search, the *fast correlation attack* was proposed in [79].

The main goal of the fast correlation attack is to convert the cryptographic problem into a decoding problem. In the fast correlation attack the number of required keystream bits is more than the number of required keystream bits in the correlation attack, and a precomputation step should be done before the attack. In the next section, we briefly review fast correlation attack and then we will develop it to be applicable to simple power analysis.

7.1 Preliminaries: Fast Correlation Attack

A typical methodology for producing random-like sequences from LFSRs in a stream cipher is to combine LFSR's bits using a nonlinear function, f. As shown in [80, 81] there is always a correlation between the output of the nonlinear function, f, and a linear combination of its inputs. In other words, if f has n inputs and is characterized by $(n-1)$ resilient functions [1] (but not n resilient functions), then there is a correlation which can be shown as

$$P(z = c_1 \cdot u_1 \oplus c_2 \cdot u_2 \oplus c_3 \cdot u_3 \cdots \oplus c_n \cdot u_n) \neq .5, \tag{7.1}$$

where u_i and z are the inputs and output of f, respectively, and c_i are fixed values $(0 < i \leq n)$. Increasing the resiliency of f will decrease the nonlinearity of the function, resulting in a higher vulnerability to algebraic attacks [60].

Instead of an exhaustive search over state bits of the LFSR as originally suggested in the correlation attack, fast correlation attacks are based on using certain parity check equations created from the feedback polynomial of the LFSR. All fast correlation attacks use two phases [79, 84, 85]. In the first phase, the attacker finds a set of parity check equations stemming from the LFSR's feedback. This phase is generally done as a precomputational process. In the second phase, the attacker

[1] An n-input m-output t-resilient function is a function runs through every possible output m-tuple an equal number of times when t arbitrary inputs are fixed and the remaining $n-t$ inputs run through all the 2^{n-t} input tuples once [82, 83].

uses the generated parity check equations in a fast decoding algorithm to recover the transmitted codeword and the initial state of an LFSR.

Assume the stream cipher contains an LFSR with the bit length of L. The probability of equality of $s_t(i)$ and z_t is shown as $P(s_t(i) = z_t) = \frac{1}{2} + \epsilon$. In well designed stream ciphers, $|\epsilon|$ is very small (smaller than .01) [77]. Using the correlation attack, the attacker can guess $s_t(i)$ with the probability of $\frac{1}{2} + \epsilon$. To increase this probability, we can use certain parity check equations. Let the feedback of the LFSR be defined as $s_t(L) = a_0 \cdot s_t(0) \oplus a_1 \cdot s_t(1) \oplus \cdots a_{L-1} \cdot s_t(L-1)$, and assume that it includes d nonzero terms or $\sum_{i=0}^{L-1} a_i = d$. Therefore, we can write

$$\begin{aligned}
s_t(L) &= a_0 \cdot s_t(0) \oplus a_1 \cdot s_t(1) \oplus \cdots a_{L-1} \cdot s_t(L-1) \\
s_t(L+1) &= a_0 \cdot s_t(1) \oplus a_1 \cdot s_t(2) \oplus \cdots a_{L-1} \cdot s_t(L) \\
s_t(L+2) &= a_0 \cdot s_t(2) \oplus a_1 \cdot s_t(3) \oplus \cdots a_{L-1} \cdot s_t(L+1) \\
&\vdots \\
s_t(2L-1) &= a_0 \cdot s_t(L-1) \oplus a_1 \cdot s_t(L) \oplus \cdots a_{L-1} \cdot s_t(2L-2).
\end{aligned} \tag{7.2}$$

For any arbitrary bit of the LFSR, $s_t(i)$, $0 \leq i < L$, we can generate d number of parity check equations using (7.2).

It is also possible to write another set of equations which are generated from $P(x^q) = P^q(x)$ (where $q = 2^j$ and $P(x)$ is the feedback polynomial for the LFSR and $P(x) = \bigoplus_{i=0}^{L-1} a_i \cdot x^i$). This increases the number of parity check equations and hence improves the probability of a successful attack. For example, assume a finite field with the primitive polynomial of $x^4 \oplus x^3 \oplus 1 = 0$. The corresponding feedback is $P(x) = x^3 \oplus 1$, or $s_t(4) = s_t(3) \oplus s_t(0)$. For the first set of equations for $s_t(10)$, we can write

$$\begin{aligned} s_t(10) &= s_t(5) \oplus s_t(6) \\ s_t(11) &= s_t(10) \oplus s_t(7) \\ s_t(14) &= s_t(13) \oplus s_t(10) \end{aligned}$$

or

$$\begin{aligned} s_t(10) &= s_t(5) \oplus s_t(6) \\ s_t(10) &= s_t(11) \oplus s_t(7) \\ s_t(10) &= s_t(13) \oplus s_t(14). \end{aligned}$$

For the second set of equations when $q = 2^1$, we can write

$$\begin{aligned} P(x^2) &= P^2(x) \\ &= x^8 \oplus x^6 \oplus 1 \\ &= \left(x^4 \oplus x^3 \oplus 1\right)^2 \\ &= 0 \end{aligned}$$

Since, $x^8 \oplus x^6 \oplus 1 = 0$ is corresponding to $s_t(8) \oplus s_t(6) \oplus s_t(0) = 0$, we can write the following set of equations for $s_t(10)$

$$\begin{aligned} s_t(10) &= s_t(8) \oplus s_t(2) \\ s_t(12) &= s_t(10) \oplus s_t(4) \\ s_t(18) &= s_t(16) \oplus s_t(10) \end{aligned}$$

or

$$\begin{aligned}
s_t(10) &= s_t(8) \oplus s_t(2) \\
s_t(10) &= s_t(12) \oplus s_t(4) \\
s_t(10) &= s_t(16) \oplus s_t(18).
\end{aligned}$$

Using $P(x^q) = P^q(x)$ for different q is dependent on the number of observable keystream bits, N. Using these proposed techniques, for any arbitrary bit of the LFSR, we can write $m = (d+1)\log(\frac{N}{2L})$ equations (where log uses base 2) [79]. In the precomputation step, the attacker sets up these parity check equations and calculates the probability of the correctness of a guess for each $s_t(i)$.

In the second phase, the attacker employs a decoding algorithm to find the hidden state bits of the LFSR. In the primary paper [79], there are two algorithms proposed for this purpose, called algorithm A and algorithm B. In algorithm A, from the calculated probabilities of correctness for different $s_t(i)$, the attacker chooses L bits which have the highest probabilities. From the selected $s_t(i)$, the state bits of the LFSR are calculated and its correctness is checked using the observed keystream of the cipher. This process continues until the right values for $s_t(i)$ and the state bits of the LFSR are found. In algorithm B, the attacker selects a threshold probability and, for each $s_t(i)$, the probability of the correctness which are smaller than the threshold causes the values of $s_t(i)$ to flip. This process will continue until the correct values of the LFSR state bits are discovered. The efficiencies of both algorithms are studied in [79] and some implementation results are summarized there.

The improvement in fast correlation attack can be done in both phases, finding parity check equations and using the calculated probabilities to find the correct $s_t(i)$.

The efficiency of the first phase has potential for improvement through the discovering of more or better parity check equations (that is low weight parity check equations).

As described in [77, 85], the required number of keystream bits to attack an LFSR in a stream cipher is:

$$N \propto \left(\frac{1}{2|\epsilon|}\right)^{\frac{2(d-2)}{d-1}} 2^{\frac{L}{d-1}}, \tag{7.3}$$

where d is the weight of the LFSR or the number of nonzero terms of the feedback of the LFSR, i.e. $d = \sum_{i=0}^{L-1} a_i$. The time complexity of the precomputation step (or first phase) where the attacker is looking for the parity check equations is [77, 85]

$$T_{pre} \in O\left(\frac{N^{d-2}}{(d-2)!}\right). \tag{7.4}$$

This complexity can be reduced by using a time-memory trade off technique, proposed in [86]. Finally, the time complexity of phase two of the attack is roughly [77, 85]:

$$T_a \in O\left(\left(\frac{1}{2|\epsilon|}\right)^{\frac{2d(d-2)}{d-1}} 2^{\frac{L}{d-1}}\right). \tag{7.5}$$

The complexities presented in equations (7.3), (7.4) and (7.5) may differ in various papers based on the algorithms employed. The best proposed attack is dependent on various input parameters. These parameters include correlation (ϵ), received keystream length (N), LFSR size (L), and the form of feedback polynomial. Other factors of consideration are the cost of precomputation time, memory and platform of computations. Even given these factors, it is still difficult to determine the time complexity of many proposed algorithms. For specific algorithms, a theoretical derivation can be made on the complexity as a function of the correlation. For others, there are mainly just simulation results [84].

7.2 Using Fast Correlation Attack in Simple Power Analysis

Chapter 6 summarized applications for the correlation attack in simple power analysis. Similar to the correlation attack, the proposed simple power analysis needs an exhaustive search over the LFSR or NLFSR state bits. This fact makes an attack infeasible for stream ciphers with long FSRs, such as Grain. In a fast correlation attack, decoding techniques are used to avoid exhaustive search and reduce the complexity of the attack. The obtained parity check equations stem from the feedback relation of the LFSR. In this section, similar to fast correlation attack, decoding techniques are implemented to reduce the complexity of the proposed technique in Chapter 6, applied to stream ciphers based on the nonlinear combination of the LFSRs.

Similar to correlation attack based on power analysis, the attacker should measure the power consumption of the cryptographic core and calculate the measured power differences, MPD, at the edges of the clock. The second step is to map the measured power differences of the LFSR, MPD, to discrete PD values, $\{-1, 0, +1\}$, denoted as PD^g. Such methods are discussed in Section 5.3. The categorization or mapping the analogue MPD to discrete PD^g, usually has some deviation from the theoretical PD of the LFSR. Incorrect PD^g values makes the implementation of the attack difficult, because of the errors introduced to the ultimate system of equations. The errors are analogous to the differences between an LFSR bit values and keystream bit values as considered in the correlation attack.

To apply fast correlation attack in simple power analysis, one must show at first the same relationships which are applicable to $s_t(i)$ of an LFSR, are also applicable to PD of the LFSR. The use of proposed decoding techniques (in a fast correlation attack) on PD (instead of $s_t(i)$) will result in L consecutive PD of the LFSR. After this, the attacker can use either the proposed technique in Section 4.1 or use the mathematical approach. The mathematical approach involves setting up a system of linear equations and using appropriate tools, such as Sage, to solve it and recover state bits of the LFSR.

Let the feedback of the LFSR be defined as

$$s_t(L) = a_0 \cdot s_t(0) \oplus a_1 \cdot s_t(1) \oplus \cdots a_{L-1} \cdot s_t(L-1). \tag{7.6}$$

If we substitute L at the left side with L, $L-1$, 0 and -1, the following four equations are obtained.

$$\begin{aligned} s_t(L) &= a_0 \cdot s_t(0) \oplus a_1 \cdot s_t(1) \oplus \cdots \oplus a_{L-1} \cdot s_t(L-1) \\ s_t(L-1) &= a_0 \cdot s_t(-1) \oplus a_1 \cdot s_t(0) \oplus \cdots \oplus a_{L-1} \cdot s_t(L-2) \\ s_t(0) &= a_0 \cdot s_t(-L) \oplus a_1 \cdot s_t(1-L) \oplus \cdots \oplus a_{L-1} \cdot s_t(-1) \\ s_t(-1) &= a_0 \cdot s_t(-L-1) \oplus a_1 \cdot s_t(-L) \oplus \cdots \oplus a_{L-1} \cdot s_t(-2). \end{aligned} \tag{7.7}$$

Adding the four equations over GF(2) and using the (2.11) notation gives us:

$$|PD_t| = a_0 \cdot |PD_{t-L}| \oplus a_1 \cdot |PD_{t-L+1}| \oplus \cdots a_{L-1} \cdot |PD_{t-1}|. \tag{7.8}$$

Hence, the same feedback relation applicable to the bits of the LFSR is applicable to $|PD_t|$. Similarly, we can extend other relationships such as $P(x^q) = P^q(x)$ (where $q = 2^j$ and $P(x) = \bigoplus_{i=0}^{L} a_i \cdot x^i$) to $|PD|$ values. Extending $s_t(i)$ relationships to $|PD_t|$, allows the use of the same parity check equations and decoding techniques for $|PD_t|$, that would be applicable in fast correlation attack.

In the first phase of the simple power analysis using fast correlation, the attacker sets up parity check equations for $|PD_t|$ values. It is possible that the attacker uses one of the techniques proposed in [84, 86, 87, 88, 89, 90, 91, 92, 93, 94] to find low weight parity check equations. This phase can be done as a precomputation step.

After collecting the parity check equations, an algorithm such as Algorithm A, Algorithm B [79] or other proposed algorithms in [86, 92, 95, 96, 97] can be used to find the correct values of $|PD_t|$ from $|PD_t^g|$. In the original algorithms proposed for the second phase of the fast correlation attack, the attacker uses z_t or the keystream bits as the guessed values of $s_t(i)$. However, in simple power analysis, we use $|PD_t^g|$, as the guessed value for $|PD_t|$. Executing the algorithm, gives us actual $|PD_t|$ values. Once the actual $|PD_t|$ values are determined, we can set up a system of equations and using appropriate tools or use the described method in Section 4.1 to calculate state bits S_t of the LFSR.

In simple power analysis, the probability of equality of PD_t and PD_t^g is defined as the probability of correctness of PD^g or $P_{cr} = P(PD_t = PD_t^g)$. Typically, this probability is much higher than the probability of correlation of z_t and $s_t(i)$.[2] Therefore, ϵ, in simple power analysis is defined as:

$$\epsilon = P_{cr} - \frac{1}{2}. \tag{7.9}$$

Hence, we can use the same equations provided in Section 7.1 to calculate the required number of power samples, (7.3), the precompuation time complexity of the attack, (7.4), and its time complexity, (7.5).

For example, consider an LFSR studied in Chapter 2 for Grain. The bit length

[2] In practical stream ciphers, ϵ for mathematical approaches are smaller than .01 [77]

of the LFSR is 80, and the feedback is $s_t(80) = s_t(67) \oplus s_t(57) \oplus s_t(42) \oplus s_t(29) \oplus s_t(18) \oplus s_t(0)$. Therefore, the feedback weight of the LFSR is $d = 6$. Assume the probability of categorizing the $|PD^g|$ values of the LFSR correctly is the same as the probability of categorizing the $|PD^g|$ values of the NLFSR correctly, studied in Chapter 5. This probability is $P_{cr} = 0.840$. Therefore, ϵ is .34. Substituting these values in equations (7.3) and (7.5), gives the time complexity of the attack to be $T_a \approx 2^{21.4}$ and the required number of power samples is $N \approx 2^{17}$. The time complexity of precomputation step is $T_{pre} \approx 2^{63.4}$. The complexity of this method (excluding the precomputation step) is significantly smaller than the proposed method in Section 5.6 which has the time complexity of 2^{45} or 2^{24}, depending on the number of observed power samples. Also this complexity is smaller than the complexity of the proposed method in Section 5.7 which is 2^{35}. However, the required number of power samples, in this method is more than the proposed method in the previous sections. The required number of power samples for the proposed methods in Section 5.6 are 6400 and 80000 (which are approximately $2^{12.6}$ and $2^{16.3}$), while the proposed method in Section 5.7 needs only 200 power samples.

Another example of direct application of this method is the Toyocrypt [43, 98]. The Toyocrypt consists of a 128 bit LFSR and a filter function. The feedback of the LFSR is initialized at the beginning of the encryption and may change for different applications (depending on IV or a secondary key). If we assume the probability of categorizing $|PD^g|$ values of the LFSR correctly is 84% ($P_{cr} = 0.840$), then ϵ is .34. For different feedbacks, equations (7.3) and (7.5), gives the different computation time and required number of power samples. For $d = 5$, the time complexity of the attack is $2^{36.2}$ and the required number of power samples is 2^{33}. The precomputation process

d	Time complexity of the attack (T_a)	Required number of power samples (N)	The complexity of precomputation phase (T_{pre})
5	$2^{36.2}$	2^{33}	$2^{97.5}$
6	2^{31}	$2^{26.4}$	2^{101}
7	$2^{30.3}$	$2^{22.2}$	$2^{104.1}$

Table 7.1: The complexity of the attack on Toyocrypt for different d ($\epsilon = .34$)

needs $2^{97.5}$ bit operations. For $d = 6$, the time complexity of the attack is 2^{31}, the required number of power samples is $2^{26.4}$ and the complexity of the precomputation process is 2^{101} bit operations. These values are summarized in the Table 7.1

It should be noted that the precomputation step needs to be done only once. When the parity check equations for an ϵ are calculated, they can be used for any other circuit with the greater ϵ.

7.3 Application of Fast Correlation Attack to Grain

In this section, we investigate the practical issue of the fast correlation attack in power analysis by applying our attack to a Grain circuit. This section will use the same simulation results, which were studied in Chapter 6 for correlation attack on Grain-v1. The implemented circuit is prototyped in TSMC 180 nm standard cell CMOS technology. Cadence Virtuoso Spectre Circuit Simulator version 5.10.41 is used to obtain the real-time power consumption of the circuit. The power supply of the circuit is 1.8 volt. The simulations were done in room temperature default noise. A total of 11700 power samples of Grain were collected.

To attack Grain, at first we should categorize the measured PD values. Since, the attacker can only measure MPD of the whole circuit, i.e. MPD^{Grain}, a method to realize the PD values of the LFSR from MPD^{Grain} must be contemplated. Then, by the applying fast correlation attack to $|PD^g|$ of the LFSR, the LFSR state bits can be uncovered. Finally, we offer a technique to calculate the NLFSR state bits using algebraic methods.

7.3.1 Categorizing the MPD of Grain

As described in [8, 14, 15], Grain contains an LFSR and an NLFSR. The power consumption of the circuit includes the power consumption of the LFSR, NLFSR and the power consumption of the combining function. As studied in previous chapters, at the triggering edges of the clock, we can ignore the power consumption of the combining functions and the feedbacks of the FSRs. For an ideal power consumption model for an FSR, the theoretical PD can be $\{-1, 0, +1\}$. For a circuit with two FSRs such as Grain, theoretical power differences of the cipher circuit can be $\{-2, -1, 0, +1, +2\}$. To categorize the analogue MPD values of the Grain circuit to the discrete $\{-2, -1, 0, +1, +2\}$ values, we use the same method as described in Section 6.2. In this method, we assume the power consumption of the circuit at time t (at the triggering edge) is the summation of the power consumption of the LFSR and the NLFSR (which is also proportional to the Hamming distance of their consecutive states). Therefore, we can conclude the overall dynamic power dissipation of the circuit at the triggering edge of the clock is proportional to $HD_t^{LFSR} + HD_t^{NLFSR}$.

The resulting equation (6.4) gives

$$PD_t^{Grain} = PD_t^{LFSR} + PD_t^{NLFSR}. \tag{7.10}$$

The first requirement is to hypothesize the PD of the LFSR from the measured power consumption of the circuit. There are two methods that can be used to obtain the power difference of the LFSR. The first method hypothesizes PD^g of the LFSR from categorized PD of Grain, $PD^{g,Grain}$. The second method guesses PD^g of the LFSR directly from measured PD values of the circuit.

In the first method, after collecting enough power samples to calculate the MPD values of Grain, the measured power differences are sorted in ascending order. The results are separated using Table 6.1. The 6.25% largest positive measured power differences are categorized as $PD^{g,Grain} = +2$. Conversely, the 6.25% most negative measured power differences are categorized as $PD^{g,Grain} = -2$. Correspondingly, the next 25% of the most positive and negative measured power differences should be categorized as $PD^{g,Grain} = +1$ and $PD^{g,Grain} = -1$, respectively. The remaining values are categorized as $PD^{g,Grain} = 0$. Experimental results have shown the probability of correctly categorizing an MPD in this method is 46.65%.

The next step is to generate the power difference values of the LFSR, $PD^{g,LFSR}$, from PD^g of Grain. For an ideal power consumption model of Grain, at different values of PD^{Grain}, the possible values of PD^{LFSR} and their probabilities (i.e., $P\left(PD^{LFSR} = i | PD^{Grain} = j\right)$, $-1 \leq i \leq +1$ and $-2 \leq j \leq +2$) are summarized in Table 7.2.

As shown in Table 7.2, only for two values of PD^{Grain} ($PD^{Grain} = +2$ and $PD^{Grain} = -2$), the PD of the LFSR can be inferred with the probability of 100%.

Table 7.2: Possible PD values for LFSR/NLFSR, for different PD^{Grain} and their probabilities.

PD^{Grain}	Possible values for PD^{FSR}	Probability $P\left(PD^{FSR} \mid PD^{Grain}\right)$
-2	-1	1
-1	-1	.5
	0	.5
	-1	.25
0	0	.5
	$+1$	.25
$+1$	$+1$	.5
	0	.5
$+2$	$+1$	1

For the rest of PD^{Grain} values ($PD^{Grain} \in \{-1, 0, +1\}$), a method to map PD^g of Grain to PD^g of the LFSR is required. The PD^g of the LFSR at time t is a function of the categorized PD^{Grain} of the same time (i.e. $PD_t^{g,Grain}$). If $PD_t^{g,Grain} = +2$ or $PD_t^{g,Grain} = +1$, we determine $PD_t^{g,LFSR} = +1$, and if $PD_t^{g,Grain} = -2$ or $PD_t^{g,Grain} = -1$, we determine $PD_t^{g,LFSR} = -1$. For $PD_t^{g,Grain} = 0$, we set $PD_t^{g,LFSR}$ to 0.

This method of categorizing PD^g for the LFSR is imperfect and as such some PD_t^{LFSR} will be categorized incorrectly. Assuming all $PD^{g,Grain}$ values are categorized correctly, the probability of correctly categorizing a $PD_t^{g,LFSR}$ is calculated as follow

$$\begin{aligned}
& P_{cr}\left(|PD^{g,LFSR}|\right) \\
= \; & P\left(PD^{LFSR} = +1 \; or \; -1 \quad | \; |PD^{Grain}| = 2\right) \cdot P\left(|PD^{Grain}| = 2\right) \\
& +P\left(PD^{LFSR} = +1 \; or \; -1 \quad | \; |PD^{Grain}| = 1\right) \cdot P\left(|PD^{Grain}| = 1\right) \\
& +P\left(PD^{LFSR} = 0||PD^{Grain}| = 0\right) \cdot P\left(|PD^{Grain}| = 0\right) \\
= \; & 1 \times 0.0625 + 1 \times 0.0625 + .5 \times .25 + .5 \times .25 + .5 \times .375 \\
= \; & .5625
\end{aligned}$$

To use the conventional fast correlation attack, the guessed $|PD^g|$ should be 0 or 1. Hence, we use the absolute values of $PD^{g,LFSR}$, i.e. $|PD^{g,LFSR}|$. This method is applied to the implemented Grain-v1 circuit. The simulation results show 55.04% of the $|PD^{g,LFSR}|$ are guessed correctly, by first categorizing the inaccurate Grain PD values. As expected this is a little less than the calculated theoretical value of 56.25% which assumed no noise in the categorization process. Hence, $|PD|$ of the LFSR can be guessed with the probability of 0.5504, which is 0.0504 better than the probability of random guessing.

The second method of guessing $|PD|$ of the LFSR, is directly from MPD of the circuit. After measuring the power consumption of the circuit and obtaining the MPD values of the circuit, we sort them. For the 25% largest positive measured power differences, the corresponding $|PD_t^{g,LFSR}|$ should be categorized to $+1$. For the 25% largest negative MPD_t, the corresponding $|PD_t^{g,LFSR}|$ should be categorized to -1. The rest of $|PD_t^{g,LFSR}|$ should be categorized to 0. We have applied this method of categorization to the measured power consumption of the circuit. The experimental results shows that the probability of correctly categorizing $|PD^{g,LFSR}|$ in this method is 55.70% which is slightly more than the previous method.

7.3.2 Deriving the LFSR State Bits of Grain

In the previous section, we describe two methods to guess the $|PD|$ values of the LFSR from measured power consumption of the circuit. To obtain state bits of the LFSR, S_t, at first we calculate L consecutive $|PD|$ values of the LFSR. To calculate L consecutive $|PD^{LFSR}|$, we use fast correlation attack technique, described in Section 7.1 and 7.2. Then, we set up a system of equations to obtain $s_t(i)$, $0 \leq i < L$, or state bits of the LFSR. As studied in Section 7.3.1, the probability of guessing $|PD^{LFSR}|$, are .5504 and .5570 for two the proposed methods. The attacker, through the use of proposed techniques to collect enough parity check equations for $|PD^{g,LFSR}|$ and the proposed algorithms in [79, 92, 95, 96, 97] (substituting z_t with $|PD^{g,LFSR}|$ in the algorithms), computes the actual $|PD^{LFSR}|$. The timing complexity of obtaining PD^{LFSR} can be calculated using (7.5). The value of ϵ obtained from equation (7.9), for the first method of categorization is 0.0504 and for the second method is 0.057.

The bit length of LFSR of Grain is 80 ($L = 80$) and d for the LFSR feedback of Grain is 6. Substituting these values in (7.3), gives the required number of power samples, which are on the order of $2^{21.3}$ and 2^{21} for the first and second method of guessing $|PD^{g,LFSR}|$, respectively. The time complexities of calculating $|PD^{LFSR}|$, are $2^{47.8}$ and 2^{46} bit operations, respectively. The time complexities of the first phase or precomputation steps are 2^{81} and 2^{80} bit operations.

7.3.3 Deriving the NLFSR State Bits of Grain

There are two methods for exploring the NLFSR bits which we will consider. The first method is applicable to Grain-v0. It is fully explained in [17] by Berbain, Gilbert and Maximov. The paper discusses a simple approach to calculate NLFSR bits after obtaining LFSR state bits. For the second method, the focus is on using power differences and keystream bits to calculate the NLFSR state bits. It is applicable to Grain-v1.

7.3.3.1 Deriving the NLFSR State Bits of Grain-v0

In [17], the authors employed a second order fast correlation attack to Grain-v0 resulting in recovery of the LFSR state bits (Their approach was purely mathematical and did not make use of power measurements.). The time complexity of the attack was 2^{43} with 2^{42} bits of memory and 2^{38} keystream bits. The LFSR state bits serve as input to calculate the NLFSR state bits.

The output combining function of Grain-v0 is defined as

$$z_t = b_t(0) \oplus b_t(63) \cdot p_t \oplus q_t, \tag{7.11}$$

where $b_t(i)$ represents the i-th bit of the NLFSR and p_t and q_t are functions of LFSR bits $s_t(i)$ at time t,

$$\begin{aligned} p_t &= 1 \oplus s_t(64) \oplus s_t(46)\left(s_t(3) \oplus s_t(25) \oplus s_t(64)\right) \qquad (7.12) \\ q_t &= s_t(25) \oplus s_t(3) \cdot s_t(46)\left(s_t(25) \oplus s_t(64)\right) \oplus s_t(64)\left(s_t(3) \oplus s_t(46)\right). \end{aligned}$$

When the initial state of the LFSR has been recovered, each output keystream bit (z_t) can be expressed by a linear relation of the NLFSR state bits (B_t). Using equation (7.11), for each z_t one of the following four equations (depending on the initial state of the LFSR), can be written

$$\begin{aligned} z_t &= b_t(0) && \text{for } p_t = 0 \text{ and } q_t = 0 \\ z_t &= b_t(0) \oplus 1 && \text{for } p_t = 0 \text{ and } q_t = 1 \\ z_t &= b_t(0) \oplus b_t(63) && \text{for } p_t = 1 \text{ and } q_t = 0 \qquad (7.13) \\ z_t &= b_t(0) \oplus b_t(63) \oplus 1 && \text{for } p_t = 1 \text{ and } q_t = 1. \end{aligned}$$

Since, p and q are balanced functions, each equation has the same probability of occurrence.

For any z_t, if one of the two first equations is applicable ($z_t = b_t(0)$ or $z_t = b_t(0) \oplus 1$), the attacker can recover the corresponding bit of the NLFSR, $b_t(0)$, instantly. For any z_t which involves two bits ($z_t = b_t(0) \oplus b_t(63)$ or $z_t = b_t(0) \oplus b_t(63) \oplus 1$), the attacker should consider both the equations for z_t and z_{t+63}. With probability $\frac{1}{2}$, the equation for z_{t+63} provides the value of $b_t(63)$ for the attacker, instantly (i.e. z_{t+63} is of the form of $z_{t+63} = b_{t+63}(63)$ or $z_{t+63} = b_{t+63}(63) \oplus 1$.). If the corresponding equation for z_{t+63} provides the value of $b_t(63)$, the attacker can substitute it in the corresponding equation of z_t and obtain $b_t(0)$. With probability $\frac{1}{2}$, the equation for z_{t+63} also involves two bits. Then the equation of $z_{t+2\times 63}$ should be considered, which can also either involve only one bit or two bits and we have to consider more equations to solve. Consequently the probability that a chain is of length n is $\frac{1}{2^{n+1}}$ and the probability that a chain is of length strictly larger than n is $\frac{1}{2^{n+1}}$.

Lets N be the number of available keystream bits and equal $n \times 63$. The probability that the length of a chain becomes larger than N is $\frac{1}{2^{n+1}}$. The probability of all chains being smaller than 10, or 630 keystream bits being enough for the attack is more than 96% which is calculated as $\left(1 - \frac{1}{2^{n+1}}\right)^{80}$, where $n = 10$. However, if one or two chains become longer than 10, the NLFSR state bits, are still recoverable by the means of a brute force approach.

7.3.3.2 Deriving the NLFSR State Bits of Grain-v1

In 2007, the designers of Grain made a change in the combining output function of Grain-v0, to make it immune to the proposed attacks such as [17]. The second method of recovering the NLFSR state bits relies on the use of both keystream and simple power analysis techniques. As explained in Section 2.2, the output combining function is

$$\begin{aligned} z_t &= b_t(0) \oplus s_t(25) \oplus b_t(63) \oplus s_t(64) \cdot s_t(3) \oplus s_t(64) \cdot s_t(46) \\ &\oplus s_t(46) \cdot s_t(25) \cdot s_t(3) \oplus s_t(64) \cdot s_t(46) \cdot s_t(3) \\ &\oplus b_t(63) \cdot s_t(46) \cdot s_t(3) \oplus b_t(63) \cdot s_t(64) \cdot s_t(46). \end{aligned} \tag{7.14}$$

If we know the LFSR state bits (S_t) and keystream bits (z_t), the nonlinear relation between NLFSR state bits (B_t) will be removed and a linear relation between NLFSR state bits can be written. For example, if $s_t(64) \cdot s_t(46) = 1$, we can write:

$$\begin{aligned} b_t(63) &= z_t \oplus b_t(0) \oplus s_t(25) \oplus b_t(63) \oplus s_t(64) \cdot s_t(3) \\ &\oplus s_t(64) \cdot s_t(46) \oplus s_t(46) \cdot s_t(25) \cdot s_t(3) \\ &\oplus s_t(64) \cdot s_t(46) \cdot s_t(3) \oplus b_t(63) \cdot s_t(46) \cdot s_t(3), \end{aligned} \tag{7.15}$$

where z_t, $s_t(3)$, $s_t(25)$, $s_t(46)$ and $s_t(64)$ are known values. Hence, the first step is to write the linear relationships between $b_t(i)$ (where $0 \leq i < 80$) for different t, based on the available keystream bits, z_t, and LFSR state bits, S_t. It should be noted that any known z_t gives us a new linear relationship between NLFR state bits.

The second step is to calculate the $PD^{g,NLFSR}$. After categorizing MPD of Grain and obtaining the PD of the LFSR, the calculation of the PD^g of the NLFSR, results from subtracting PD^{LFSR} from PD^{Grain}. Therefore, the PD^{NLFSR} is calculated as

$$PD_t^{g,NLFSR} = PD_t^{g,Grain} - PD_t^{LFSR}. \tag{7.16}$$

The PD of the NLFSR gives us the relation

$$PD_t^{NLFSR} = [b_t(L) \oplus b_t(L-1)] - [b_t(0) \oplus b_{t-1}(0)]. \tag{7.17}$$

The rest of attack is similar to the proposed attack in Section 5.7, which is applying simple power analyzing to LFSR based stream cipher in noisy environments. Similar to Section 5.7, we can rewrite (7.17) for $PD_t^{g,NLFSR}$:

$$PD_t^{g,NLFSR} = g_t - h_t, \tag{7.18}$$

where

$$\begin{aligned} g_t &= b_t(L) \oplus b_{t-1}(L) \\ h_t &= b_t(0) \oplus b_{t-1}(0). \end{aligned} \tag{7.19}$$

$PD_t^{NLFSR} \neq -1$		$PD_t^{NLFSR} \neq +1$	
$PD_t^{g,Grain}$	PD_t^{LFSR}	$PD_t^{g,Grain}$	PD_t^{LFSR}
$+2$	-1	-2	$+1$
$+2$	0	-2	0
-1	$+1$	$+1$	-1

Table 7.3: The target $PD_t^{g,NLFSR}$

The possible values of g and h for different $PD_t^{g,NLFSR}$ are summarized in Table 5.5. Identical to the proposed method in Section 5.7 the goal is to find target $PD_t^{g,NLFSR}$ with their corresponding actual PD_t^{NLFSR} not equal to $+1$, $PD_t^{NLFSR} \neq +1$ (or not equal to -1, $PD_t^{NLFSR} \neq -1$). Due to high rate of error in $PD_t^{g,NLFSR}$, the proposed method in Section 5.7 is not directly applicable for $PD_t^{g,NLFSR}$. In other words, if $PD_t^{g,NLFSR} = +1$ (or $PD_t^{g,NLFSR} = -1$) it does not imply the actual PD_t^{NLFSR} is definitely not equal -1 (or $+1$).

Studying the experimental results of the circuit shows that if categorized $PD_t^{g,Grain} = +2$ and the calculated PD of LFSR is 0 or -1, then the probability of $PD_t^{NLFSR} = -1$ is negligible. Similarly, if categorized $PD_t^{g,Grain} = -2$ and the calculated PD of LFSR is 0 or $+1$, then the probability of $PD_t^{NLFSR} = +1$ is negligible. Also, when $PD_t^{g,Grain} = +1$ (or $PD_t^{g,Grain} = -1$) and the calculated PD of LFSR is -1 (or $+1$), then the probability of $PD_t^{NLFSR} = -1$ (or $PD_t^{NLFSR} = +1$) is negligible. We summarize the target $PD_t^{g,NLFSR}$ in Table 7.3. The experimental results show the probability of occurrence of a target $PD_t^{g,NLFSR}$ is .054.

After collecting enough power samples, we have to find target $PD^{g,NLFSR}$ which imply $PD_t^{NLFSR} \neq +1$ or $PD_t^{NLFSR} \neq -1$. For each target PD, we can write a second order equation based on g and h. The conversion of equations from g and h to $b_t(i)$, where $0 \leq i < 80$ and t can be any value is possible, using equations (7.19). For $b_t(i)$, when $t + i \geq L$ we can use equation (7.15) or the provided linear equations in the first step (based on known z_t and LFSR state bits, S_t) to convert it to a linear relation of $b_t(i)$, where $0 \leq i < 80$, for a specific t. Therefore, we have a system of equations in which the unknown variables are $b_t(i)$ where $0 \leq i < 80$ for a specific t and the degrees of the system of equations are two.

The last step is to solve the nonlinear system of equations, through the use of the conventional algebraic methods such as relinearization or XL algorithms. Converting the system of nonlinear equations to a linear system of equations (using the XL method), the maximum number of unknown variables is

$$\binom{80}{2} + 80 = 3240 \approx 2^{11.66}. \tag{7.20}$$

Using the proper tools such as Sage [63] and Gaussian elimination, the complexity of calculating $b_t(i)$ for $0 \leq i < 80$ for the initial state ($t = 0$) is cubic in the number of known variables and this will take no more than 2^{35} bit operations.

In order to solve a system of nonlinear equations using the XL method, provided that after linearization there will be enough linearly independent equations, more than 80 equations are required. A system with 100 equations or target PD and 80 unknown variables is solvable with high probability [73]. In order to set up 100 nonlinear equations, 100 target $PD_t^{g,NLFSR}$ values are needed. Since, the probability of occurrence of a target $PD_t^{g,NLFSR}$ is 0.054, the collection of 100 target power difference requires on average we need $100 \times .054^{-1}$ or 1852 power samples. Hence, on average with 1852 power samples and keystream bits and the complexity of 2^{35}, the NLFSR state bits may be determined.

Since, Grain consists of an LFSR and an NLFSR and a nonlinear combining function, recovering the state bits of both FSRs results in the recovering state bits of the stream cipher. As described above, determining the LFSR state bits needs 2^{46} bit operations (following a precomputation of 2^{80} bit operations) and calculating NLFSR state bits needs 2^{35} bit operations.

7.4 Summary

In this chapter, we have improved the method of recovering LFSR state bits using simple power analysis. This method is applicable when the measured power consumption of the circuit is not accurate. In other words, when the attacker can not map the measured power consumption of the circuit to the actual state bits of the circuit, this method is very useful. In previous chapters, we developed a correlation attack and use it in simple power analysis, however the complexity of the attack was still high. Here, we have combined the simple power analysis technique and fast correlation attack to reduce the time complexity of the attack. The proposed technique is applicable on LFSR based stream ciphers.

In order to demonstrate the applicability of the attack, the simulated power consumption of a CMOS implemented circuit of Grain-v0 and Grain-v1 were collected. Then, we considered the proposed technique based on the information derived from the collected power simulations. We have shown that Grain is susceptible to the proposed attack. The timing complexity of recovering LFSR state bits with our method is 2^{46} (following a precomputation step of complexity 2^{80}) and the timing complexity of recovering NLFSR state bits is 2^{35}. It should be considered that the precomputation phase in cryptanalysis needs to be done only once and the results can be used for any other attack. This complexity is much smaller than exhaustive search and correlation attack, described in the previous section. The timing complexity of exhaustive search to obtain state bits of Grain is 2^{160} and the timing complexity of correlation attack is 2^{80}.

The proposed complexities to recover the state bits of Grain is calculated for the standard LFSR defined in [8, 14, 15]. d in the defined LFSR in [8, 14, 15] is 6. If the number of the tabs in the LFSR was 7, the number of required power samples for a successful attack would be $2^{18.6}$ and the timing complexity of the attack would be 2^{50}, following a precomputation step of complexity $2^{85.9}$. These values when the $d = 5$ are $2^{24.7}$, $2^{71.5}$ and $2^{43.5}$ for the number of required power samples, the timing complexity of the attack and the timing complexity of the precomputation step, respectively.

Chapter 8

Conclusion

8.1 Summary of Research

In this thesis, we have studied the cryptanalysis of FSR based stream ciphers using power analysis methods. Several new attacks on stream ciphers based on power analysis are presented. Theoretical and simulation results concerning their performance are presented.

We summarize the contributions of our research as follows. We extended the conventional simple power analysis from single to multiple LFSR based stream ciphers. Through the further development of conventional simple power analysis, we have been enabled to make it applicable on NLFSR bases stream ciphers. These attacks have been examined by application to E0, LILI-128 and Grain in an ideal environment, where there is perfect correlation between measured power consumption and state bits of the cipher.

As well, we have examined the effect of noise and/or inaccurate power measurements in simple power analysis attack. No previous research has studied these practical issues for power analysis attacks applied to stream ciphers. We developed the simple power analysis methods to be practical with inaccurate measurements or power consumption data, proposing an approach to recover the state bits of stream ciphers including an LFSR or NLFSR with inaccurate power information. The efficiency of these methods have been investigated by application to an 80-bit NLFSR simulated as a CMOS circuit.

Another main contribution of this research is to apply the classical mathematical methods of attacking LFSR based stream ciphers in the context of simple power analysis. For this purpose, we have studied algebraic, correlation and fast correlation attacks. We have developed an algebraic attack for simple power analysis, to propose a new attack on LFSR based stream ciphers. This attack is practical with inaccurate measurement data. The application of this attack has been studied using 80 bit LFSRs. It has been shown that this attack needs the timing complexity of 2^{35} bit operations to calculate the state bits of the 80 bits LFSR based on CMOS simulated data.

Further, we have developed a correlation attack for side channel analysis and applied it to Grain. By application, we have verified that Grain is vulnerable to side channel analysis attack with the complexity of 2^{80} which is substantially less than the brute force recovering of the cipher state. To reduce the complexity of the attack, we have employed the fast correlation attack. We have shown that fast correlation techniques can be employed in power analysis to determine LFSR state bits of a stream cipher. Subsequently, we have applied the fast correlation techniques in power analysis of Grain. It is shown that Grain is susceptible to the proposed attack even with noisy environments and inaccurate measurements.

This research shows the stream ciphers are more susceptible to power analysis attacks than they were thought before. In practical hardware implementations of the stream ciphers, the power analysis of the circuit must be considered as a potential threat and the hardware cores must designed in a way to stop leaking of power information data.

8.2 Future Work

Although, we have begun work on this area, there are many issues that still need to be considered in the application of power analysis to stream ciphers. To apply fast correlation attack for simple power analysis we have used the proposed techniques in fast correlation attack to reduce the timing complexity of the attack. These techniques are mostly mathematical and based on finding more parity check equations to increase the probability of guessing PD values correctly. Future research could use the power consumption information to increase the probability of guessing PD values. For that purpose, we can use the proposed techniques in Chapter 5, such as robust threshold, sequence consistency and RE/FE equivalence techniques. Using these techniques along with proposed mathematical approaches can increase the efficiency of the attack and reduce the timing complexity and required number of power samples.

Other work that remains to be done in future is applying the proposed algebraic method on block ciphers. Not much research has been done to apply algebraic side channel attacks in noisy environments on block ciphers. The proposed algebraic technique in this dissertation for noisy environments is applicable on block ciphers and can helps the attacker to use it for practical implementations.

Another complement for this research is finding countermeasure techniques for simple power analysis. This research shows the necessity of designing and developing countermeasure circuits beside the implemented cryptographic cores and care must be taken to design implementations which do not leak power consumption information.

Bibliography

[1] I. N. Bronshtein and K. A. Semendyayev, *Handbook of mathematics (3rd ed.)*. London, UK, UK: Springer-Verlag, 1997.

[2] A. J. Menezes, S. A. Vanstone, and P. C. V. Oorschot, *Handbook of Applied Cryptography*. Boca Raton, FL, USA: CRC Press, Inc., 1st ed., 1996.

[3] H. C. A. V. Tilborg, *Fundamentals of Cryptology: A Professional Reference and Interactive Tutorial*. Norwell, MA, USA: Kluwer Academic Publishers, 1st ed., 1999.

[4] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Trans. Inf. Theor.*, vol. 22, pp. 644–654, Sept. 1976.

[5] European Telecommunications Standards Institute, "Digital cellular telecommunications system (phase 2+); technical realization of the short message service (sms); point-to-point (pp)." (TS 100 901, GSM 03.40 version 7.3.0 Release 1998), November 1999.

[6] B. Sig, *Specifications of the Bluetooth System - Version 1.1 B*, 2001.

[7] C. Paar and J. Pelzl, *Understanding Cryptography: A Textbook for Students and Practitioners.* Springer-Verlag New York Inc, 2010.

[8] M. Hell, T. Johansson, and W. Meier, "Grain - a stream cipher for constrained environments," *Int. J. Wire. Mob. Comput.*, vol. 2, pp. 86–93, may 2007.

[9] C. D. Canniere and B. Preneel, "Trivium specifications," vol. 2006, 2005. `url` `http://www.ecrypt.eu.org/stream/`.

[10] B. Tsaban and U. Vishne, "Efficient linear feedback shift registers with maximal period," *Finite Fields and Their Applications*, vol. 8, no. 2, pp. 256–267, 2002.

[11] A. Klein, *Stream Ciphers.* Dresden, SA, Germany: Springer-Verlag, 2013.

[12] F. Masoodi, S. Alam, and M. U. Bokhari, "An analysis of linear feedback shift registers in stream ciphers," *International Journal of Computer Applications*, vol. 46, no. 17, pp. 46–49, 2012. Published by Foundation of Computer Science, New York, USA.

[13] "ECRYPT. eSTREAM: ECRYPT Stream Cipher Project, IST-2002-507932," 2005. `url` `http://www.ecrypt.eu.org/stream/`.

[14] M. Hell, T. Johansson, and W. Meier, "Grain - a stream cipher for constrained environments," 2005. `url` `http://www.ecrypt.eu.org/stream/`.

[15] M. Hell, T. Johansson, A. Maximov, and W. Meier, "The Grain family of stream ciphers," in *New Stream Cipher Designs - The eSTREAM Finalists* (M. J. B. Robshaw and O. Billet, eds.), vol. 4986 of *Lecture Notes in Computer Science*, pp. 179–190, Springer, 2008.

[16] A. Maximov, “Cryptanalysis of the Grain family of stream ciphers,” in *Proceedings of the 2006 ACM Symposium on Information, computer and communications security*, ASIACCS ’06, (New York, NY, USA), pp. 283–288, ACM, 2006.

[17] C. Berbain, H. Gilbert, and A. Maximov, “Cryptanalysis of Grain,” in *Proceedings of the 13th international conference on Fast Software Encryption*, FSE’06, (Berlin, Heidelberg), pp. 15–29, Springer-Verlag, 2006.

[18] S. Khazaei, M. Hassanzadeh, and M. Kiaei, “Distinguishing attack on Grain,” 2005. `url` `http://www.ecrypt.eu.org/stream/`.

[19] A. Biryukov and A. Shamir, “Cryptanalytic Time/Memory/Data Tradeoffs for stream ciphers,” in *Proceedings of the 6th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology*, ASIACRYPT ’00, (London, UK, UK), pp. 1–13, Springer-Verlag, 2000.

[20] T. Bjorstad, “Cryptanalysis of Grain using Time / Memory / Data Tradeoffs v1.0,” 2008. `url` `http://www.ecrypt.eu.org/stream/papersdir/2008/012.pdf`.

[21] C. De Cannière, O. Küçük, and B. Preneel, “Analysis of Grain’s initialization algorithm,” in *Proceedings of the Cryptology in Africa 1st international conference on Progress in cryptology*, AFRICACRYPT’08, (Berlin, Heidelberg), pp. 276–289, Springer-Verlag, 2008.

[22] F. Armknecht, "Algebraic attacks on stream ciphers.." ECCOMAS - 4th European Congress on Computational Methods in Applied Sciences and Engineering, technical paper to the given talk, 2004.

[23] Y. Shaked and A. Wool, "Cryptanalysis of the bluetooth E0 cipher using obdd's," in *Proceedings of the 9th international conference on Information Security*, ISC'06, (Berlin, Heidelberg), pp. 187–202, Springer-Verlag, 2006.

[24] A. Clark, E. Dawson, J. Fuller, J. D. Golic, H.-J. Lee, W. Millan, S.-J. Moon, and L. Simpson, "The LILI-II keystream generator," in *Proceedings of the 7th Australian Conference on Information Security and Privacy*, ACISP '02, (London, UK, UK), pp. 25–39, Springer-Verlag, 2002.

[25] MATLAB, *version 7.10.0 (R2010a)*. Natick, Massachusetts: The MathWorks Inc., 2010.

[26] X. Huang, W. Huang, X. Liu, C. Wang, Z. jing Wang, and T. Wang, "Reconstructing the nonlinear filter function of LILI-128 stream cipher based on complexity," *CoRR*, vol. abs/cs/0702128, 2007.

[27] P. P. Deepthi and P. Sathidevi, "Hardware stream cipher based on LFSR and modular division circuit," *International Journal of Electrical and Computer Engineering*, pp. 791–799, 2008.

[28] P. C. Kocher, "Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems," in *Proceedings of the 16th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '96, (London, UK, UK), pp. 104–113, Springer-Verlag, 1996.

[29] J. Kelsey, B. Schneier, D. Wagner, and C. Hall, “Side channel cryptanalysis of product ciphers,” *J. Comput. Secur.*, vol. 8, pp. 141–158, Aug. 2000.

[30] N. P. Smart, “Physical side-channel attacks on cryptographic systems.,” *Software Focus*, vol. 1, no. 2, pp. 6–13, 2000.

[31] S. Ghosh, D. Mukhopadhyay, and D. Roychowdhury, “Petrel: Power and timing attack resistant elliptic curve scalar multiplier based on programmable arithmetic unit,” *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 58, no. 8, pp. 1798–1812, 2011.

[32] D. Brumley and D. Boneh, “Remote timing attacks are practical,” *Comput. Netw.*, vol. 48, pp. 701–716, Aug. 2005.

[33] E. D. Win and B. Preneel, “Elliptic curve public-key cryptosystems - an introduction,” in *State of the Art in Applied Cryptography, Course on Computer Security and Industrial Cryptography - Revised Lectures*, (London, UK, UK), pp. 131–141, Springer-Verlag, 1998.

[34] J. Fan, X. Guo, E. De Mulder, P. Schaumont, B. Preneel, and I. Verbauwhede, “State-of-the-art of secure ECC implementations: a survey on known side-channel attacks and countermeasures,” in *2010 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pp. 76–87, 2010.

[35] S. Chari, J. R. Rao, and P. Rohatgi, “Template attacks,” in *Revised Papers from the 4th International Workshop on Cryptographic Hardware and Embedded Systems*, CHES ’02, (London, UK, UK), pp. 13–28, Springer-Verlag, 2003.

[36] N. Hanley, M. Tunstall, and W. P. Marnane, "Information security applications," ch. Unknown Plaintext Template Attacks, pp. 148–162, Berlin, Heidelberg: Springer-Verlag, 2009.

[37] S. Mangard, E. Oswald, and T. Popp, *Power Analysis Attacks: Revealing the Secrets of Smart Cards (Advances in Information Security)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2007.

[38] A. Amara, F. Amiel, and T. Ea, "FPGA vs. ASIC for low power applications," *Microelectronics Journal*, vol. 37, pp. 669–677, Aug. 2006.

[39] A. Moradi, M. Salmasizadeh, M. T. Manzuri Shalmani, and T. Eisenbarth, "Vulnerability modeling of cryptographic hardware to power analysis attacks," *Integr. VLSI J.*, vol. 42, pp. 468–478, Sept. 2009.

[40] S.-M. S. Kang and Y. Leblebici, *CMOS Digital Integrated Circuits Analysis & Design*. New York, NY, USA: McGraw-Hill, Inc., 3 ed., 2003.

[41] S. Burman, D. Mukhopadhyay, and K. Veezhinathan, "LFSR based stream ciphers are vulnerable to power attacks," in *Proceedings of the cryptology 8th international conference on Progress in cryptology*, INDOCRYPT'07, (Berlin, Heidelberg), pp. 384–392, Springer-Verlag, 2007.

[42] B. Qu, D. Gu, Z. Guo, and J. Liu, "Differential power analysis of stream ciphers with LFSRs," *Computers and Mathematics with Applications*, vol. 65, no. 9, pp. 1291 – 1299, 2013. Advanced Information Security.

[43] M. Mihaljevic and H. Imai, "Cryptanalysis of Toyocrypt-HS1 stream cipher," *IEICE Transactions on Fundamentals*, vol. E85-A, pp. 66–73, 2002.

[44] P. C. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '99, (London, UK, UK), pp. 388–397, Springer-Verlag, 1999.

[45] B. P. J. Lano, N. Mentens and I. Verbauwhede, "Power analysis of synchronous stream ciphers with resynchronization mechanism," in *Proceedings of the SASC 2004 - The State of the Art of Stream Ciphers*, (Brugge, Belgium), pp. 327–333, Springer-Verlag, October 2004. Available at http://www.ecrypt.eu.org/stvl/sasc/record.html.

[46] W. Fischer, B. M. Gammel, O. Kniffler, and J. Velten, "Differential power analysis of stream ciphers," in *Proceedings of the 7th Cryptographers' track at the RSA conference on Topics in Cryptology*, CT-RSA'07, (Berlin, Heidelberg), pp. 257–270, Springer-Verlag, 2006.

[47] Y. Han, X. Zou, Z. Liu, and Y. cheng Chen, "Efficient DPA attacks on AES hardware implementations.," *IJCNS*, vol. 1, no. 1, pp. 68–73, 2008.

[48] J. Ambrose, N. Aldon, A. Ignjatovic, and S. Parameswaran, "Anatomy of differential power analysis for AES," in *Symbolic and Numeric Algorithms for Scientific Computing, 2008. SYNASC '08. 10th International Symposium on*, pp. 459–466, 2008.

[49] M. E. Hellman, "A cryptanalytic Time-Memory Tradeoff," *IEEE Transactions on Information Theory*, vol. 26, no. 4, pp. 401–406, 1980.

[50] P. Oechslin, “Making a faster cryptanalytic Time-Memory Tradeoff,” in *Advances in Cryptology - CRYPTO 2003* (D. Boneh, ed.), vol. 2729 of *Lecture Notes in Computer Science*, pp. 617–630, Springer Berlin Heidelberg, 2003.

[51] M. Albrecht, “Algebraic attacks on the Courtois Toy cipher,” *Cryptologia*, vol. 32, pp. 220–276, July 2008.

[52] A. Kipnis and A. Shamir, “Cryptanalysis of the HFE public key cryptosystem by relinearization,” in *Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO ’99, (London, UK, UK), pp. 19–30, Springer-Verlag, 1999.

[53] N. T. Courtois, “Higher order correlation attacks, XL algorithm and cryptanalysis of Toyocrypt,” in *Proceedings of the 5th international conference on Information security and cryptology*, ICISC’02, (Berlin, Heidelberg), pp. 182–199, Springer-Verlag, 2003.

[54] N. T. Courtois, “Cryptanalysis of Sfinks,” in *Proceedings of the 8th international conference on Information Security and Cryptology*, ICISC’05, (Berlin, Heidelberg), pp. 261–269, Springer-Verlag, 2006.

[55] N. T. Courtois and W. Meier, “Algebraic attacks on stream ciphers with linear feedback,” in *Proceedings of the 22nd international conference on Theory and applications of cryptographic techniques*, EUROCRYPT’03, (Berlin, Heidelberg), pp. 345–359, Springer-Verlag, 2003.

[56] N. T. Courtois, "Algebraic attacks on combiners with memory and several outputs," in *Proceedings of the 7th international conference on Information Security and Cryptology*, ICISC'04, (Berlin, Heidelberg), pp. 3–20, Springer-Verlag, 2005.

[57] C. Cid and S. Murphy, *Algebraic Aspects of the Advanced Encryption Standard.* Springer Publishing Company, Incorporated, 1st ed., 2010.

[58] M. S. Mohamed, J. Ding, J. Buchmann, and F. Werner, "Algebraic attack on the MQQ public key cryptosystem," in *Proceedings of the 8th International Conference on Cryptology and Network Security*, CANS '09, (Berlin, Heidelberg), pp. 392–401, Springer-Verlag, 2009.

[59] T. Siegenthaler, "Decrypting a class of stream ciphers using ciphertext only," *IEEE Trans. Comput.*, vol. 34, pp. 81–85, Jan. 1985.

[60] T. Siegenthaler, "Correlation-immunity of nonlinear combining functions for cryptographic applications (corresp.)," *IEEE Trans. Inf. Theor.*, vol. 30, pp. 776–780, Sept. 2006.

[61] A. A. Zadeh and H. M. Heys, "Applicability of simple power analysis to stream ciphers constructed using multiple LFSRs," in *Proceedings of the 25th Annual Canadian Conference on Electrical and Computer Engineering (CCECE)*, pp. 1–6, IEEE, 2012.

[62] P. Geffe, "How to protect data with ciphers that are really hard to break," *Electronics*, vol. 46, pp. 99–101, Jan. 1973.

[63] W. A. Stein *et al.*, *Sage Mathematics Software (Version 5.6)*. The Sage Development Team, 2013. `http://www.sagemath.org`.

[64] C. L. Chen, "Linear dependencies in linear feedback shift registers," *IEEE Trans. Comput.*, vol. 35, pp. 1086–1088, Dec. 1986.

[65] A. A. Zadeh and H. M. Heys, "Theoretical simple power analysis of the Grain stream cipher," in *Proceedings of the 26th Annual Canadian Conference on Electrical and Computer Engineering (CCECE)*, pp. 1–5, IEEE, 2013.

[66] A. A. Zadeh and H. M. Heys, "Simple power analysis applied to nonlinear feedback shift registers," *IET Information Security (In Press)*, 2013.

[67] M. Renauld, F.-X. Standaert, and N. Veyrat-Charvillon, "Algebraic side-channel attacks on the AES: Why time also matters in DPA," in *CHES* (C. Clavier and K. Gaj, eds.), vol. 5747 of *Lecture Notes in Computer Science*, pp. 97–111, Springer, 2009.

[68] C. Carlet, J.-C. Faugre, C. Goyet, and G. Renault, "Analysis of the algebraic side channel attack," *Journal of Cryptographic Engineering*, vol. 2, no. 1, pp. 45–62, 2012.

[69] M. Mohamed, S. Bulygin, M. Zohner, A. Heuser, M. Walter, and J. Buchmann, "Improved algebraic side-channel attack on AES," *Journal of Cryptographic Engineering*, pp. 1–18, 2013.

[70] J. Haastad, "Some optimal inapproximability results," *J. ACM*, vol. 48, pp. 798–859, July 2001.

[71] R. Crowston, G. Gutin, M. Jones, E. J. Kim, and I. Z. Ruzsa, “Systems of linear equations over $\mathbb{F}_2$ and problems parameterized above average,” in *Proceedings of the 12th Scandinavian conference on Algorithm Theory*, SWAT’10, (Berlin, Heidelberg), pp. 164–175, Springer-Verlag, 2010.

[72] S.-W. Zhao and X.-S. Gao, “Note: minimal achievable approximation ratio for MAX-MQ in finite fields,” *Theor. Comput. Sci.*, vol. 410, pp. 2285–2290, May 2009.

[73] N. Courtois, A. Klimov, J. Patarin, and A. Shamir, “Efficient algorithms for solving overdefined systems of multivariate polynomial equations,” in *Proceedings of the 19th international conference on Theory and application of cryptographic techniques*, EUROCRYPT’00, (Berlin, Heidelberg), pp. 392–407, Springer-Verlag, 2000.

[74] M. Mezard and A. Montanari, *Information, Physics, and Computation.* New York, NY, USA: Oxford University Press, Inc., 2009.

[75] D. Coppersmith and S. Winograd, “Matrix multiplication via arithmetic progressions,” in *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, STOC ’87, (New York, NY, USA), pp. 1–6, ACM, 1987.

[76] A. A. Zadeh and H. M. Heys, “Application of simple power analysis to stream ciphers constructed using feedback shift registers,” *Submitted to The Computer Journal*, 2013.

[77] A. Canteaut, "Fast correlation attacks against stream ciphers and related open problems," in *Theory and Practice in Information-Theoretic Security, 2005. IEEE Information Theory Workshop on*, pp. 49–54, Oct.

[78] J. Lano, N. Mentens, B. Preneel, and I. Verbauwhede, "Power analysis of synchronous stream ciphers with resynchronization mechanism," in *ECRYPT Workshop, SASC - The State of the Art of Stream Ciphers*, (Brugge,BE), pp. 327–333, 2004.

[79] W. Meier and O. Staffelbach, "Fast correlation attacks on certain stream ciphers," *Journal of Cryptology*, vol. 1, pp. 159–176, 1989. 10.1007/BF02252874.

[80] G. Z. Xiao and J. L. Massey, "A spectral characterization of correlation immune combining functions," *IEEE Trans. Inf. Theor.*, vol. 34, pp. 569–571, Sept. 2006.

[81] J. D. Golic, "Correlation via linear sequential circuit approximation of combiners with memory," in *Proceedings of the 11th annual international conference on Theory and application of cryptographic techniques*, EUROCRYPT'92, (Berlin, Heidelberg), pp. 113–123, Springer-Verlag, 1993.

[82] X. mo Zhang and Y. Zheng, "Cryptographically resilient functions," *IEEE Transactions on Information Theory*, vol. 43, pp. 1740–1747, 1997.

[83] B. Chor, O. Goldreich, J. Hstad, J. Friedman, S. Rudich, and R. Smolensky, "The bit extraction problem of t-resilient functions (preliminary version)," in *FOCS*, vol. 26, pp. 396–407, IEEE Computer Society, 1985.

[84] M. Agren, C. Londahl, M. Hell, and T. Johansson, "A survey on fast correlation attacks," *Cryptography Commun.*, vol. 4, pp. 173–202, Dec. 2012.

[85] M. Hell, T. Johansson, and L. Brynielsson, "An overview of distinguishing attacks on stream ciphers," *Cryptography and Communications*, vol. 1, no. 1, pp. 71–94, 2009.

[86] P. Chose, A. Joux, and M. Mitton, "Fast correlation attacks: An algorithmic point of view," in *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques: Advances in Cryptology*, EUROCRYPT '02, (London, UK, UK), pp. 209–221, Springer-Verlag, 2002.

[87] M. J. Mihaljevic and J. D. Golic, "A fast iterative algorithm for a shift register initial state reconstruction given the noisy output sequence," in *Proceedings of the international conference on cryptology on Advances in cryptology*, AUSCRYPT '90, (New York, NY, USA), pp. 165–175, Springer-Verlag New York, Inc., 1990.

[88] V. Chepyzhov and B. Smeets, "On a fast correlation attack on certain stream ciphers," in *Proceedings of the 10th annual international conference on Theory and application of cryptographic techniques*, EUROCRYPT'91, (Berlin, Heidelberg), pp. 176–185, Springer-Verlag, 1991.

[89] W. T. Penzhorn and G. J. Kuhn, "Computation of low-weight parity checks for correlation attacks on stream ciphers," in *Proceedings of the 5th IMA Conference on Cryptography and Coding*, (London, UK, UK), pp. 74–83, Springer-Verlag, 1995.

[90] W. Penzhorn, "Correlation attacks on stream ciphers: Computing low-weight parity checks based on error-correcting codes," in *Fast Software Encryption* (D. Gollmann, ed.), vol. 1039 of *Lecture Notes in Computer Science*, pp. 159–172, Springer Berlin, Heidelberg, 1996.

[91] J. Golic, "Computation of low-weight parity-check polynomials," *Electronics Letters*, vol. 32, no. 21, pp. 1981–1982, Oct.

[92] T. Johansson and F. Jonsson, "Improved fast correlation attacks on stream ciphers via convolutional codes," in *Proceedings of the 17th international conference on Theory and application of cryptographic techniques*, EUROCRYPT'99, (Berlin, Heidelberg), pp. 347–362, Springer-Verlag, 1999.

[93] A. Canteaut and M. Trabbia, "Improved fast correlation attacks using parity-check equations of weight 4 and 5," in *Proceedings of the 19th international conference on Theory and application of cryptographic techniques*, EUROCRYPT'00, (Berlin, Heidelberg), pp. 573–588, Springer-Verlag, 2000.

[94] D. Wagner, "A generalized birthday problem," in *Proceedings of the 22nd Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '02, (London, UK, UK), pp. 288–303, Springer-Verlag, 2002.

[95] V. V. Chepyzhov, T. Johansson, and B. Smeets, "A simple algorithm for fast correlation attacks on stream ciphers," in *Proceedings of the 7th International Workshop on Fast Software Encryption*, FSE '00, (London, UK, UK), pp. 181–195, Springer-Verlag, 2001.

[96] T. Johansson and F. Jnsson, “Fast correlation attacks based on turbo code techniques,” in *Advances in Cryptology CRYPTO 99* (M. Wiener, ed.), vol. 1666 of *Lecture Notes in Computer Science*, pp. 790–790, Springer Berlin / Heidelberg, 1999.

[97] M. J. Mihaljevic, M. P. C. Fossorier, and H. Imai, “A low-complexity and high-performance algorithm for the fast correlation attack,” in *Proceedings of the 7th International Workshop on Fast Software Encryption*, FSE ’00, (London, UK, UK), pp. 196–212, Springer-Verlag, 2001.

[98] E. Dawson, A. Clark, H. Gustafson, B. Millan, and L. Simpson, *Evaluation of TOYOCRYPT-HS1.* User manual, 2001.

Appendix A

Complete Table of Probabilities for Sequence Consistency Method

Sequence $\{PD^g_{t-L}, PD^g_t, PD^g_{t+L}\}$	Probability of $PD^g_t = PD_t$ for rising edge	Probability of $PD^g_t = PD_t$ for falling edge
$\{+1, -1, +1\}$	.984	.918
$\{-1, +1, -1\}$	.984	.918
$\{+1, 0, -1\}$	.968	.852
$\{-1, 0, +1\}$	.968	.852
$\{0, +1, -1\}$	.906	.781
$\{0, -1, +1\}$	.906	.781
$\{-1, +1, 0\}$	.906	.781
$\{+1, -1, 0\}$	.906	.781
$\{+1, +1, -1\}$	.475	.438
$\{-1, -1, +1\}$	.475	.438
$\{+1, -1, -1\}$	.475	.438

Table A.1: Probability of $PD^g_t = PD_t$ for sequences of three PD^g values for rising edge ($P_{gr} = .840$) and falling edge ($P_{gr} = .680$) (part 1).

Sequence $\{PD^g_{t-L}, PD^g_t, PD^g_{t+L}\}$	Probability of $PD^g_t = PD_t$ for rising edge	Probability of $PD^g_t = PD_t$ for falling edge
$\{-1, +1, +1\}$	.475	.438
$\{+1, +1, +1\}$	.475	.438
$\{-1, -1, -1\}$	.475	.438
$\{-1, 0, -1\}$	.473	.439
$\{+1, 0, +1\}$	.473	.439
$\{+1, +1, 0\}$	.456	.405
$\{-1, -1, 0\}$	.456	.405
$\{0, +1, +1\}$	.456	.405
$\{0, -1, -1\}$	.456	.405
$\{0, 0, -1\}$	.860	.680
$\{0, 0, +1\}$	.860	.680
$\{0, -1, 0\}$	.860	.680
$\{0, +1, 0\}$	.860	.680
$\{-1, 0, 0\}$	.860	.680
$\{+1, 0, 0\}$	.860	.680

Table A.2: Probability of $PD^g_t = PD_t$ for sequences of three PD^g values for rising edge ($P_{gr} = .840$) and falling edge ($P_{gr} = .680$) (part 2).

Appendix B

Algebraic Attack

B.1 Linearization

Let us assume, the goal is to solve a system of multivariate quadratic equations, with n variables, $x_1, x_2, \cdots, x_n$ and m equations. Obviously, to find at least one unique answer for this system of equations, the number of equations should be more than or equal to the number of unknown variables ($m \geq n$). The idea of linearization is very simple. It is systematically replacing every quadratic term in a system of multivariate quadratic equations by a new variable.

To perform linearization, every quadratic term, $x_i \times x_j$ should be replaced by a new term, y_{ij}. Replacing all quadratic terms transforms the system of quadratic equations into a system of linear equations. Using conventional algorithms such as Gaussian elimination, it is straightforward to solve a system of linear equation. As stated above, there must be as many equations as unknowns variables. Therefore, there is a big restriction on using this method. To have enough equations in the linear system of equations the number of equations, in the nonlinear system of equations must be at least $m \geq \frac{n^2}{2}$.

The relinearization technique works for the MQ problem (or system of multivariate quadratic equations), if the number of equations is at least ϵn^2, where n is the number of variables and $0 \leq \epsilon \leq \frac{1}{2}$ [52]. If there exist a system of multivariate quadratic equations which meets this requirement, the first step to solve such a system is to replacement the monomials with new variables as done in linearization. That is, replace every quadratic term $x_i \times x_j$ by a new variable y_{ij}. Then more equations should be constructed using connections between the new variables. For example, $x_1 \times x_2 \times x_3 \times x_4 = y_{12} \times y_{34} = y_{13} \times y_{24} = y_{14} \times y_{23}$.

Now more equations are produced, but all the new equations have quadratic terms in them. So linearization is executed again to get a system of linear equations. In fact, relinearization is just a method to generate new equations and solve the system of nonlinear equations with fewer available equations.

B.2 XL

XL (which stands for eXtended Linearization) was proposed in [73]. let A be a system of multivariate quadratic equations, $l_i = 0$ $(1 < i \leq m)$ where l_i is a multivariate polynomial $f_i(x_1, x_2, \cdots, x_n) - b_i$. Hence, $l_i = f_i(x_1, x_2, \cdots, x_n) - b_i$ and the system to solve is:

$$A : \begin{cases} l_1(x_1, x_2, \cdots, x_n) = 0 \\ l_2(x_1, x_2, \cdots, x_n) = 0 \\ \cdots \\ l_m(x_1, x_2, \cdots, x_n) = 0 \end{cases} \tag{B.1}$$

The problem is to find at least one solution at the form of $X = (x_1, x_2, \cdots, x_n) \in \{0,1\}^n$. In XL, it is assumed that the system has a unique solution.

The maximum degree of the equations is $K \geq 2$. In the XL algorithm, equations are created in the form $\left(\prod_{j=1}^{k} x_{i,j}\right) \times l_i = 0$, where $x_{i,j} \in \{x_1, x_2, \cdots, x_n\}$. The generated equations from this method are denoted by $x^k l$. Let $D \in \mathbb{N}$ (It should be noted over GF(2) for any variable, $x^2 = x$). We consider all $x^k l$ equations with degree smaller than D and we represent them with I_D. Therefore, I_D will be the linear space generated by all the equations of the form $x^k l$ for $0 \leq k < D - 1$. In XL algorithm, at first all I_D equations are generated. Then, each monomial of I_D is considered as a new unknown variable (the degree of all monomials are smaller than D). If I_D includes enough equations, it is possible to use Gaussian elimination algorithms to solve the system of equations. Following estimation for D is offered in [73].

$$D \geq \frac{n}{\sqrt{m}}. \tag{B.2}$$

After picking D, the list of original variables are selected and a new list of variables are constructed with every possible power less than or equal to $D - 2$. For example, if the list of variables is (x, y, z) and $D = 4$, then the new list will be (x, y, z, xy, xz, yz). Each original equation should be multiplied by each variable from the new list. This operation generates more linearly independent equations. It is not necessarily true that all the new equations are linearly independent, but most of them will be.

As a matter of fact, XL is an algorithm to generate more equation (similar to relinearization method) such that the system is solvable using linearization method.