HELSINKI UNIVERSITY OF TECHNOLOGY
Department of Computer Science and Engineering
Laboratory for Theoretical Computer Science

Risto Hakala

# Linear Cryptanalysis of Two Stream Ciphers

Master's thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Technology

Espoo, December 5, 2007

Supervisor: Prof. Kaisa Nyberg
Instructor: Prof. Kaisa Nyberg

HELSINKI UNIVERSITY OF TECHNOLOGY
Department of Computer Science and Engineering

ABSTRACT OF THE MASTER'S THESIS

Author: Risto Hakala

Date: December 5, 2007
Pages: viii + 52

Title of thesis: Linear Cryptanalysis of Two Stream Ciphers

Professorship: Theoretical Computer Science
Code: T-79

Supervisor: Prof. Kaisa Nyberg

Instructor: Prof. Kaisa Nyberg

Stream ciphers are symmetric encryption primitives, which are used to ensure confidentiality of messages in digital communications. Compared to block ciphers, stream ciphers are often more efficient and allow a more compact implementation—they are suitable especially for telecommunication applications. The security of stream ciphers has not been on the same level with the most secure block ciphers, however. This is why the design and analysis of stream ciphers has started to receive even more attention. In this thesis, a survey of linear cryptanalysis of shift register-based stream ciphers is performed, and attacks on two recently developed stream ciphers, SOBER-128 and Shannon, are presented. The attacks are linear distinguishing attacks, which aim at distinguishing a keystream from a truly random sequence using linear cryptanalytic techniques. A linear distinguishing attack is based on a linear transformation, which is able to detect statistical bias in the keystream. The transformation that is used in the distinguishing attack is formed by approximating the nonlinear parts in the cipher with linear functions. In order to find a good transformation, we present a new technique for analyzing linear approximations of T-functions efficiently. This technique is used to construct the distinguishing attack on SOBER-128, in which case the attack also gives information about the secret constant that is used in SOBER-128. The distinguishing attack on Shannon is based on a multidimensional linear transformation. A notable benefit is gained from using a multidimensional transformation instead of a one-dimensional transformation.

Keywords: stream ciphers, linear cryptanalysis, distinguishing attacks, linear approximations, multiple linear approximations, T-functions, SOBER-128, Shannon

| TEKNILLINEN KORKEAKOULU<br>Tietotekniikan osasto | | DIPLOMITYÖN TIIVISTELMÄ |
|---|---|---|
| Tekijä | Risto Hakala | Päiväys 5. joulukuuta 2007 |
| | | Sivuja viii + 52 |
| Työn nimi | Kahden jonosalausmenetelmän lineaarinen kryptoanalyysi | |
| Professuuri | Tietojenkäsittelyteoria | Koodi T-79 |
| Työn valvoja | Prof. Kaisa Nyberg | |
| Työn ohjaaja | Prof. Kaisa Nyberg | |

Jonosalausmentelmät ovat symmetrisiä salausprimitiivejä, joilla pyritään takaamaan viestien luottamuksellisuus digitaalisessa tietoliikenteessä. Lohkosalausmenetelmiin verrattuna jonosalausmenetelmät ovat usein suoritustehokkaampia ja pienikokoisempia toteutukseltaan—ne soveltuvat erityisesti langattoman tietoliikenteen sovelluksiin. Kehitettyjen jonosalaimien turvallisuus ei ole kuitenkaan ollut samalla tasolla parhaiden lohkosalaimien kanssa. Siksi jonosalaimien suunnitteluun ja analyysiin on alettu kiinnittää entistä enemmän huomiota. Tässä diplomityössä luodaan katsaus siirtorekisteripohjaisten jonosalausmenetelmien lineaariseen kryptoanalyysiin ja esitetään hyökkäykset kahdelle hiljattain kehitetylle jonosalausmenetelmälle, jotka ovat SOBER-128 ja Shannon. Hyökkäykset ovat lineaarista kryptoanalyysia hyödyntäviä erotteluhyökkäyksiä, joilla pyritään erottamaan jonosalaimen tuottama avainjono satunnaisesta lukujonosta. Lineaarinen erotteluanalyysi perustuu syötelukujonoon käytettävään lineaarimuunnokseen, joka pystyy erottelemaan tilastollisia poikkeamia avainjonosta. Erotteluanalyysissä käytetty muunnos muodostetaan approksimoimalla salauksen epälineaarisia osia lineaarisilla funktioilla. Hyvän muunnoksen löytämiseksi työssä on esitetty uusi tekniikka, jolla voidaan analysoida T-funktioiden lineaarisia approksimaatioita tehokkaasti. Tätä käytetään hyökkäyksen muodostamiseen SOBER-128:lle, jonka tapauksessa voidaan myös saada informaatiota rakenteessa olevasta salaisesta vakiosta. Shannon-jonosalaimen erotteluanalyysi perustuu moniulotteiseen lineaarimuunnokseen, jonka avulla saavutetaan selvä etu verrattuna yksiulotteista muunnosta käyttävään hyökkäykseen.

Avainsanat

jonosalausmenetelmät, lineaarinen kryptoanalyysi, erotteluhyökkäykset, lineaariset approksimaatiot, useat lineaariset approksimaatiot, T-funktiot, SOBER-128, Shannon

# Acknowledgements

This work has been done at the Laboratory for Theoretical Computer Science, Helsinki University of Technology, and has been part of the Ad-Hoc Networks Project and the Stream Cipher Project funded by the Finnish Defence Forces and the Scientific Advisory Board for Defence, respectively.

I would like to thank Prof. Kaisa Nyberg for guidance and support on research, and for valuable comments and suggestions regarding this thesis.

I would also like to express my gratitude to my family for their constant encouragement and support throughout my studies.

Risto Hakala

December 5, 2007

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

The proliferation of computers and communication systems during the last decades has brought an increasing demand for methods to protect information in digital form. The academic disciplines of computer security, information security, and information assurance all share the common goals of ensuring security and reliability of information systems. *Cryptology* is the science that examines and provides methods to assure the security of information systems.

The research in cryptology is divided into *cryptography* and *cryptanalysis*. Cryptography studies the design of cryptographic systems used to preserve security of information systems. Modern cryptographic systems commonly aim at providing a number of security services, such as confidentiality, integrity, authenticity, and non-repudiation. Confidentiality stands for ensuring that information cannot be accessed by unauthorized entities or processes. Integrity is the assurance that information stays consistent, correct, and accessible. Authenticity means verifying the author of the information, and non-repudiation refers to the concept of ensuring that a contract cannot be later denied by either of the parties involved. The ability to provide these security services is assessed within cryptanalysis. The objective of cryptanalysis is to attempt to circumvent the security of the system being examined. The techniques in cryptography can be further divided into symmetric and asymmetric techniques.

The topic of this thesis is cryptanalysis of *stream ciphers*, which are symmetric encryption primitives that have recently attracted much attention in the cryptographic community. In the following section, we give a short introduction to design and analysis of stream ciphers. Then we give an outline of this thesis.

## 1.1 Stream Ciphers

Stream ciphers are a class of symmetric encryption primitives intended to ensure confidentiality of messages in cryptograhic systems. They are widely used in practice, especially in telecommunication applications. *Block ciphers* form another class of symmetric encryption algorithms. To paraphrase Rueppel [1986] about the distinction between block and stream ciphers, stream ciphers operate on invidual symbols with a time-varying transformation instead of on entire blocks with a fixed transformation.

Stream ciphers try to imitate the behaviour of a theoretically unbreakable cipher, the *one-time pad*. The one-time pad uses a sequence of truly random bits as the secret key. The plaintext message is added bit-by-bit to the key to produce the ciphertext. The remarkable feature about one-time pads is its security: Shannon [1949] showed that an adversary cannot gain information about the message from the ciphertext given infinite computing power. The disadvantage of the one-time pad is key management, i.e., generation, exchange, and storage of the key. Stream ciphers try to overcome these difficulties by generating a sequence of pseudo-random bits, called the *keystream*, from a short secret key. Hence, the security of a stream cipher is largely dependent on how random the keystream can be made to appear. Statistical cryptanalytic attacks on stream ciphers often aim at detecting statistical bias in the keystream. This sometimes allows gaining useful information about the cipher, such as its internal state. At worst, it can lead to full key recovery [see, e.g., Fluhrer et al., 2001].

So far, block ciphers have been the most studied class of ciphers. The publication of the Data Encryption Standard (DES) [FIPS PUB 46] in 1977 was an important step for development of cryptanalysis. The analysis of DES has resulted in many important cryptanalytic techniques, which have been applied to both block ciphers and stream ciphers. Probably the most significant inventions have been differential [Biham and Shamir, 1990] and linear [Matsui and Yamagishi, 1993, Matsui, 1994] cryptanalysis. Even though these statistical techniques were introduced for block ciphers, they have been applied on stream ciphers also. Another block cipher standard, the Advanced Encryption Standard (AES) [FIPS PUB 197], was published in 2001. The cipher was developed by Daemen and Rijmen, and it has been extensively studied and is currently used widely in many applications.

Despite the success of block ciphers, there is a need for stream ciphers, since stream ciphers seem to have some advantages over block ciphers. Stream ciphers are often much more efficient and allow a more compact implementation; however, their security has not been on the same level with the most secure block ciphers. Since there does not seem to be any specific reason for this, design and analysis of stream ciphers have started to receive more attention from the cryptographic community. Two recent European projects that have had influence in this direction are the NESSIE and

eSTREAM projects. NESSIE was a project within the Information Society Technologies Programme of the European Commission from 2000–2003. Its main objective was to put forward a portfolio of strong cryptographic primitives, including stream ciphers. However, weaknesses were found in all stream cipher submissions, and therefore no stream cipher made it to the final portfolio. One of the submissions was SOBER-128, which is also analyzed in this thesis. After NESSIE came to an end, the eSTREAM project was initiated by the European Network of Excellece for Cryptology, ECRYPT, to identify new stream ciphers that might be suitable for widespread adoption. The project is planned to be completed in 2008.

## 1.2 Outline of the Thesis

The subject of this thesis is linear distinguishing attacks on stream ciphers. Distinguishing attacks are attacks on encryption primitives, in which an adversary is able to distinguish the keystream generated by the stream cipher from a sequence of truly random bits. In linear distinguishing attacks, linear cryptanalytic techniques are used to formulate the distinguishing attack. These attacks are based on application of linear transformation on the keystream such that statistical bias is revealed from it. A linear distinguishing attack is presented on two stream ciphers, which are SOBER-128 and Shannon. The outline of the thesis is as follows.

Chapter 2 discusses stream ciphers. We give an overview of different stream cipher types, common building blocks for stream ciphers, and classical stream cipher designs. Stream ciphers are usually classified as asynchronous or synchronous stream ciphers. We describe these stream cipher types and discuss their advantages and disadvantages. The building blocks to be covered include Boolean functions, S-boxes, $\omega$-narrow T-functions, and two types of shift registers. Finally, a classical stream cipher design, the nonlinear filter generator, is discussed.

Chapter 3 discusses application of linear cryptanalytic methods in stream cipher cryptanalysis. We give an overview of different classes of cryptanalytic attacks, general distinguishing attacks, linear distinguishing attacks, and their application to nonlinear filter generators. We consider linear distinguishing attacks, which make use of one-dimensional or multidimensional linear transformations of the keystream. These transformations are formed using biased approximations of nonlinear parts of the cipher with linear functions. Computational techniques for facilitating this process are also discussed. We present a technique for analyzing linear properties of T-functions. This technique is used to help cryptanalysis of SOBER-128.

Chapter 4 presents a linear distinguishing attack on SOBER-128 [Hawkes et al., 2003], which is a synchronous stream cipher. The keystream generator of SOBER-128 is a nonlinear filter generator with a linear feedback shift

register. The distinguisher for the attack is constructed by first approximating the nonlinear parts of the filter function with linear functions such that an approximate relation involving keystream variables and shift register state variables is formed. The recurrence relation of the shift register is then used to cancel out the state variables such that an approximation involving only keystream variables is obtained. We observe that the statistical bias of the relation changes according to a secret key-dependent constant. This fact is used to gain information about the constant based on a number of linear approximations. To our current estimates, it takes on the average $2^{113.5}$ keystream terms to get one bit of information of the secret constant and $2^{124.6}$ terms to get four bits of information.

Chapter 5 presents a linear distinguishing attack on Shannon [Hawkes et al., 2007], which is a synchronous stream cipher like SOBER-128. Shannon is a nonlinear filter generator with a nonlinear feedback shift register. Our attack on Shannon uses multiple linear transformations on the keystream simultaneously. We construct the attack similarly as for SOBER-128, but in addition to forming a linear approximation for the nonlinear filter, we also form an approximation for the nonlinear recurrence relation of the shift register. This is used for obtaining a linear approximation involving keystream variables. The attack requires about $2^{106.996}$ keystream terms to succeed.

Chapter 6 summarizes the contributions of the thesis and draws some conclusions. Possible directions for future research are also discussed.

# Chapter 2

# Stream Ciphers

Stream ciphers are symmetric encryption primitives that are widely used to preserve confidentiality of wireless communication. For example, A5 is used to encrypt speech in GSM networks, SNOW has been standardized for use in 3G communications, E0 is used in the Bluetooth protocol, and RC4 is used (and unfortunately also misused) in the WEP protocol. In comparison to block ciphers, stream ciphers operate on invidual symbols with a time-varying transformation instead of on entire blocks with a fixed transformation. This distinction is not always clear-cut, since—in certain modes of operation—a block cipher can be used in such a way that it acts as a stream cipher.

Stream ciphers try to imitate the behaviour of a theoretically unbreakable cipher, the *one-time pad* [Shannon, 1949]. The one-time pad uses a sequence of random bits as the key. To produce the ciphertext, the key is combined with the plaintext in a bit-by-bit fashion using the exclusive-or ($\oplus$) operation. Let the binary vectors $P = (p_0, \ldots, p_{N-1})$ and $K = (k_0, \ldots, k_{N-1})$ denote the plaintext and the key, respectively. The ciphertext $C = (c_0, \ldots, c_{N-1})$ is given as

$$c_t = p_t \oplus k_t, \quad t = 0, \ldots, N-1.$$

The one-time pad provides *perfect secrecy* [Shannon, 1949], if the key is perfectly random, kept secret, and used only once. Perfect secrecy means that an adversary is unable to obtain any information of the plaintext from the ciphertext, even if he or she has infinite computing power. Since the key is a sequence of random bits, the plaintext is statistically independent of the ciphertext, i.e., $\Pr[p_t \mid c_t] = \Pr[p_t]$, for $t = 0, \ldots, N-1$. Thus, the ciphertext is completely meaningless without any knowledge of the key. Despite this advantage, the one-time pad is not a suitable choice for many applications. Since the key must be at least the same length as the message, it is difficult to distribute it to the correct parties. Careful treatment is also required to prevent the key from being reused or revealed to an adversary. Stream

ciphers try to overcome these disadvantages by using a short key to produce a sequence of pseudo-random bits, called the keystream.

Stream ciphers have advantages over block ciphers in certain applications. Since stream ciphers operate on invidual symbols, there is no need for padding. Indeed, stream ciphers are often used in applications, where the length of the plaintext is not known before encryption. Another advantage compared to block ciphers is that there is very little error propagation. These advantages make stream ciphers a good choice for securing wireless communication. Stream ciphers provide more flexibility in other applications also. For example, it is possible to produce the keystream separately of the plaintext. In software-oriented stream ciphers, the symbol size is often chosen to correspond the word size of the CPU of the system. This allows a more efficient use of available operations in the CPU. In hardware environments, bit-oriented stream ciphers outdo block ciphers in throughput.

An overview of stream ciphers is given in this chapter. Stream ciphers are usually classified into two types: *asynchronous* and *synchronous stream ciphers*. In Section 2.1, we discuss properties of these stream cipher types. Common building blocks for stream ciphers are discussed in Section 2.2. A classical stream cipher design, the nonlinear filter generator, is discussed in Section 2.3. Since both of the stream ciphers in this thesis are synchronous stream ciphers and nonlinear filter generators, the emphasis is put on related subjects.

## 2.1 Types of Stream Ciphers

In this thesis, a stream cipher is viewed as a function $\mathcal{A}$ that takes a plaintext message $P$, a key $K$, and an initialization vector $IV$ as inputs. The output of the function is the ciphertext $C$. In other words, the interface of the stream cipher $\mathcal{A}$ can be written as

$$C = \mathcal{A}(P, K, IV).$$

It is common that stream ciphers operate in two phases: the *setup phase* and the *encryption/decryption phase*.

1. **Setup phase** The setup phase initializes the cipher using the key $K$ and the initialization vector $IV$. The aim is to produce a random looking initial state, denoted $\sigma_0$, by mixing the keybits and the initialization vector bits together. This is achieved by executing the cipher a predefined number of rounds.

2. **Encryption/decryption phase** In the encryption/decryption phase, a stream cipher generates a keystream symbol $z_t$ at each time step $t \geq 0$ based on the internal state $\sigma_t$ and the key $K$. If the stream cipher is used for encryption, the keystream symbol $z_t$ is combined

with the plaintext symbol $p_t$ to produce the ciphertext $c_t$; when used for decryption, $z_t$ is combined with the ciphertext $c_t$ to produce the plaintext $p_t$. The state of the stream cipher is then updated to $\sigma_{t+1}$ based on the current state $\sigma_t$, the key $K$, and sometimes the ciphertext $c_t$.

Stream ciphers are often classified based on how the internal state $\sigma_t$ is updated. If the state is updated independently of the ciphertext $c_t$, the stream cipher is classified as synchronous. In contrast, if the state is updated based on previous ciphertext symbols $c_{t-k}, \ldots, c_{t-1}$, the cipher is called asynchronous or self-synchronizing. In addition, there exists some designs that fall into both categories. For example, the eSTREAM candidate Phelix is one such recent design. The differences between the update functions of synchronous and self-synchronizing ciphers have some relevance on how the cipher operates in practice. A closer look to these stream cipher types is taken in the following sections.

### 2.1.1 Synchronous Stream Ciphers

Synchronous stream ciphers produce a keystream independently of the ciphertext. A synchronous stream cipher can be described as a finite state machine that has an internal state and an update function. In addition, synchronous stream ciphers have a keystream function that is used to produce the keystream, and an output function that is used to combine the keystream with the plaintext.

- **Internal state** The internal state of a synchronous stream cipher at time $t$ is the vector $\sigma_t = (\sigma_t^{(0)}, \ldots, \sigma_t^{(l-1)})$ of $l$ invidual components $\sigma_t^{(i)}$, $i = 0, \ldots, l-1$.

- **State update function** The state update function $G$ produces the next state $\sigma_{t+1}$ from the current state $\sigma_t$ and the key $K$:

$$\sigma_{t+1} = G(\sigma_t, K).$$

- **Keystream function** The keystream function $F$ produces a new keystream symbol $z_t$ from the key $K$ and the internal state $\sigma_t$:

$$z_t = F(\sigma_t, K).$$

- **Output function** The output function $H$ is an injective function that combines a plaintext symbol $p_t$ and a keystream symbol $z_t$, and outputs a ciphertext symbol $c_t$:

$$c_t = H(p_t, z_t).$$

The function $H$ has to be injective for a fixed $z_t$ so that it is possible to determine the plaintext $p_t$ for which $H(p_t, z_t) = c_t$ based on the ciphertext $c_t$ and the keystream $z_t$.

The output function $H$ is usually chosen to be the bitwise exlusive or ($\oplus$), in which case the ciphertext symbol is produced as $c_t = p_t \oplus z_t$. Such synchronous stream ciphers are often refered to as *additive synchronous stream ciphers*. The exclusive-or is its own inverse. Therefore, the encryption and decryption processes are the same with additive synchronous stream ciphers.

Since each plaintext symbol is encrypted independently of other plaintext symbols, a corruption of a ciphertext symbol in case of a transmission error does not affect the decryption of other ciphertext symbols. In other words, synchronous stream ciphers have no error propagation. This might appear as a desirable property; however, it also means that it is harder to detect transmission errors. For an attacker, it is easier to make controlled changes to parts of the ciphertext knowing fully how they affect the corresponding plaintext. Also, to decrypt the ciphertext with a synchronous stream cipher the keystream has to be completely in sync with the corresponding ciphertext. Resynchronization of the keystream and the ciphertext is often achieved by using markers, which contain information of the position in the ciphertext. This technique allows the keystream to be resynchronized after the next marker position. An advantage of synchronous stream ciphers is that the keystream can be created separately of the plaintext and used later for encryption.

### 2.1.2 Self-Synchronizing Stream Ciphers

A self-synchronizing stream cipher produces a keystream depending on a fixed number of previous ciphertext symbols. Denote by $l$ the number of previous ciphertext symbols that each keystream symbol depends on. The operation of a self-synchronizing stream cipher can be described by the following equations:

$$\begin{aligned} \sigma_t &= (c_{t-l}, \ldots, c_{t-1}), \\ z_t &= F(\sigma_t, K), \\ c_t &= H(p_t, z_t). \end{aligned}$$

As synchronous stream ciphers, self-synchronizing stream ciphers have a setup phase and an encryption/decryption phase. In the setup phase, the $IV$ is used to initialize the initial state $\sigma_0$. In the encryption/decryption phase, previous $l$ ciphertext symbols $c_{t-l}, \ldots, c_{t-1}$ are used directly as the state $\sigma_t$. Otherwise, the encryption/decryption phase works as with synchronous ciphers.

Since the internal state $\sigma_t$ is defined by the $l$ previous ciphertext symbols $c_{t-l}, \ldots, c_{t-1}$, a corruption of a single ciphertext symbol affects the

decryption of the next $l$ ciphertext symbols also. On the other hand, this also means that the keystream is resynchronized after receiving $l$ ciphertext symbols without errors. Hence, self-synchronizing ciphers are able to resume the correct decryption automatically. Self-synchronizing stream ciphers have at least the following drawbacks. Since the state of the cipher depends on some previous ciphertext symbols, the attacker always knows all values that determine the next state—even worse, the attacker can control a part of the values that define the next state in a chosen plaintext attack. These facts make it difficult to assess the security of a self-synchronizing stream cipher in comparison to a synchronizing stream cipher.

## 2.2 Building Blocks for Stream Ciphers

In this section, we discuss a few widely used building blocks in stream ciphers. Two subjects are considered: *shift registers* and *Boolean functions*. With Boolean functions, the emphasis is put on a class of Boolean functions, called *T-functions*.

### 2.2.1 Boolean Functions

We use $\mathbb{F}_2^n$ to denote the $n$-dimensional vector space formed by binary vectors $x = (x_0, \ldots, x_{n-1})$ of $n$ coordinates $x_i \in \mathbb{F}_2$, $i = 0, \ldots, n-1$. A function $f\colon \mathbb{F}_2^n \to \mathbb{F}_2$ is called a Boolean function. In this section, we introduce basic concepts related to Boolean functions and their usage in this thesis. Boolean functions are generally represented using a *truth table* or an *algebraic normal form*.

- **Truth table** A truth table of a Boolean function $f$ of $n$ variables is a table in which all possible input vectors for $f$ are listed together with the corresponding output value.
- **Algebraic normal form** An algebraic normal form is a method for representing every Boolean function in a standard form. Any Boolean function $f$ of $n$ variables has a unique polynomial representation in $\mathbb{F}_2[x_0, \ldots, x_{n-1}]/\langle x_0^2 + x_0, \ldots, x_{n-1}^2 + x_{n-1}\rangle$:

  $$f(x) = \sum_{u \in \mathbb{F}_2^n} a_u x^u, \quad a_u \in \mathbb{F}_2,$$

  where we denote $x^u = \prod_{i=0}^{n-1} x_i^{u_i}$. This is called the algebraic normal form of $f$.

#### S-boxes

A vector-valued Boolean function $f\colon \mathbb{F}_2^n \to \mathbb{F}_2^k$ is commonly called as a *substitution box* (S-box). An S-box can be considered to consist of $k$ Boolean

functions $f_i$, $i = 0, \ldots, k-1$, called the coordinate functions of $f$. By coordinate functions every S-box can be expressed in the following way:

$$\begin{aligned} f(x)_0 &= f_0(x_0, \ldots, x_{n-1}) \\ f(x)_1 &= f_1(x_0, \ldots, x_{n-1}) \\ &\vdots \\ f(x)_{k-1} &= f_{k-1}(x_0, \ldots, x_{n-1}). \end{aligned}$$

S-boxes are fundamental building blocks in contemporary cryptography and typically the only source of nonlinearity in ciphers. Several cryptanalytic methods exploiting linearity properties have been developed, most prominently differential and linear cryptanalysis and their variations [see, e.g., Pasalic, 2003].

### T-functions

T-functions [Klimov and Shamir, 2003, 2004, 2005] are a class of vector-valued Boolean functions, where the $j$th output bit is uniquely determined by the first $j$ bits of each input word. They are highly efficient and have been claimed to have desirable cryptographical properties [Klimov and Shamir, 2003]. Many stream ciphers employ T-functions in their structure. For example, the eSTREAM candidates ABC, VEST, TSC, and Mir-1 are built upon different types of T-functions. SOBER-128 also employs T-functions in its structure. Therefore, it is useful to examine general properties of these functions before considering a specific T-function more closely. We use the following terminology and notation to discuss T-functions in this thesis.

Let $n$, $m$ and $d$ be positive integers, and let $f$ be a multivariate mapping that has $m$ $n$-bit input words and $d$ $n$-bit output words. The multivariate input is represented by the $m \times n$ matrix $x = (x_{i,j})$ over $\mathbb{F}_2$ with the input words organized as the rows of the matrix. We use $x^{(i-1)}$ to denote the $i$th row vector of $x$ and $x_{j-1}$ to denote the $j$th column vector of $x$. The output of $f$ is defined similarly as a matrix in $\mathbb{F}_2^{d \times n}$. Multivariate T-functions are defined as follows:

**Definition 1.** A function $f\colon \mathbb{F}_2^{m \times n} \to \mathbb{F}_2^{d \times n}$ is called a *T-function*, if the $j$th column $f(x)_{j-1}$ of the output depends only on the first $j$ columns $x_0, \ldots, x_{j-1}$ of the input.

We use the notation $f(x) = (f_0(x), \ldots, f_{n-1}(x))$ to refer to the coordinate functions $f_j$ of $f$. T-functions may be evaluated recursively using a parametric expression. First, let us define *parametric functions* [Klimov and Shamir, 2003] as follows:

**Definition 2.** A *parametric function* is a function $g(x_1, \ldots, x_a; \alpha_1, \ldots, \alpha_b)$, whose arguments are separated by a semicolon into inputs $x_i$ and parameters $\alpha_j$.

For $j = 1, \ldots, n-1$, let $\alpha_j$ be a parametric function that depends on the first $j$ input columns $x_0, \ldots, x_{j-1}$ and has the functions $\alpha_0, \ldots, \alpha_{j-1}$ as parameters. Note that the parameters $\alpha_0, \ldots, \alpha_{j-1}$ are obviously not needed, since every parameter can be determined completely with the previous columns. We will include them in the expression, however, because they allow us to simplify certain T-functions later. The $\alpha$ function is a special type of T-function, which is generally called a *parameter* [Klimov and Shamir, 2005]. By using parameters, every T-function can be expressed as a parametric function in the following way:

$$\begin{aligned} f(x)_0 &= f_0(x_0; \alpha_0) \\ f(x)_1 &= f_1(x_1; \alpha_1(x_0; \alpha_0)) \\ f(x)_2 &= f_2(x_2; \alpha_2(x_1, x_0; \alpha_1, \alpha_0)) \\ &\vdots \\ f(x)_{n-1} &= f_{n-1}(x_{n-1}; \alpha_{n-1}(x_{n-2}, \ldots, x_0; \alpha_{n-2}, \ldots, \alpha_0)). \end{aligned}$$

where the parameters $\alpha_0, \ldots, \alpha_{n-1}$ are defined according to $f$. Not all T-functions, however, need all previous input columns and parameters during evaluation. By treating T-functions as well as their parameters as parametric functions, some T-functions may be defined with a parameter $\alpha_j(x_{j-1}; \alpha_{j-1})$ that depends only on the previous column $x_{j-1}$ and the previous parameter $\alpha_{j-1}$. Therefore, we adapt the concept of $\omega$-narrow T-functions introduced by Daum [2005], but extend the definition to include noninteger values of $\omega$ also.

**Definition 3.** Let $f\colon \mathbb{F}_2^{m \times n} \to \mathbb{F}_2^{d \times n}$ be a T-function. The T-function $f$ is called *$\omega$-narrow*, if there exists parameters $\alpha_j\colon \mathbb{F}_2^m \times V \to V$, $j = 1, \ldots, n-1$, such that $\omega = \log_2|V|$ and $\alpha_j$ depends on the previous input column $x_{j-1}$ and parameter $\alpha_{j-1}$ so that $f$ can be recursively evaluated as

$$\begin{aligned} f(x)_0 &= f_0(x_0; \alpha_0) \\ f(x)_1 &= f_1(x_1; \alpha_1(x_0; \alpha_0)) \\ f(x)_2 &= f_2(x_2; \alpha_2(x_1; \alpha_1)) \\ &\vdots \\ f(x)_{n-1} &= f_{n-1}(x_{n-1}; \alpha_{n-1}(x_{n-2}; \alpha_{n-2})). \end{aligned}$$

The *narrowness* of $f$ is the smallest $\omega$ such that $f$ is still $\omega$-narrow.

Therefore, the process of evaluating a $\omega$-narrow T-function may be viewed as a Markov chain, where—given an input—the $j$th parameter and input column determines the $(j+1)$th parameter.

According to Klimov and Shamir [2005], all Boolean operations and most of the arithmetic operations available on modern computers are T-functions.

In particular, bitwise and ($\wedge$), or ($\vee$), exclusive-or ($\oplus$) and complementation ($\bar{\cdot}$), and addition ($\boxplus$), subtraction (binary $\boxminus$), negation (unary $\boxminus$) and multiplication ($\boxdot$) modulo $2^n$ are univariate or bivariate T-functions. These operations are called *primitive operations* [Klimov and Shamir, 2003]. Note that right shifts ($\gg$) or rotations ($\lll$ and $\ggg$) are not T-functions. Left shifts ($\ll$) are allowed, however, since $x \ll k$ equals $x \boxdot 2^k$. The composition of two T-functions is a T-function, and hence, an arbitrary composition of primitive operations is also a T-function. Maximov and Johansson [2005] use the term *pseudo-linear functions modulo* $2^n$ to refer to the following T-functions: compositions of Boolean operations, and additions, subtractions and negations of $n$-bit integers.

It is obvious that the parameter $\alpha_j$ in the recursive definition of addition, subtraction and negation modulo $2^n$ may be determined completely by the $j$th column $x_{j-1}$ and the $j$th parameter $\alpha_{j-1}$. For example, addition modulo $2^n$ with $m$ inputs can be evaluated with $f_j(x_j; \alpha_j) = x_{0,j} \oplus \cdots \oplus x_{m-1,j} \oplus \alpha_j(x_{j-1}; \alpha_{j-1})$, where the parameters are computed with the function

$$\alpha_j(x_{j-1}; \alpha_{j-1}) = \lfloor (w_{\mathrm{H}}(x_{j-1}) + \alpha_{j-1})/2 \rfloor, \quad j = 1, \ldots, n-1,$$

where $\alpha_0 := 0$, and $w_{\mathrm{H}}(x)$ denotes the Hamming weight of $x$, i.e., the number of nonzero components of $x$. In this case, the parameter $\alpha_{j-1}$ may be viewed as the carry from the previous round. The maximum carry value for addition modulo $2^n$ with $m$ inputs is $m-1$, which is obviously also the maximum parameter value. Thus, the range of the parameter is $V = \{0, \ldots, m-1\}$, and the narrowness of addition modulo $2^n$ with $m$ inputs is $\log_2 m$. Apart from multiplication, the narrowness of primitive operations is not dependent on the length $n$ of the input words. In this thesis, we concentrate only on T-functions that have this property.

### 2.2.2 Shift Registers

In cryptography, shift registers are used to generate pseudo-random sequences from a seed value. The majority of current stream ciphers use shift registers as basic building blocks. In this section, we introduce two shift register types: *linear feedback shift registers* (LFSRs) and *nonlinear feedback shift registers* (NLFSRs). Stream ciphers are often constructed using LFSRs, since much of their mathematical properties can be readily determined. They are also efficient and easy to implement in hardware.

#### Linear Feedback Shift Registers

An LFSR consists of a state and a linear recurrence relation that defines how the state is updated at each time step $t \geq 0$. The state consists of $r$ memory cells, each of which holds one symbol. A symbol is an element from the finite field $\mathbb{F}_q$, where $q = p^k$ for prime $p$ and an integer $k$. Denote the state

at time $t \geq 0$ by $S_t = (s_t, \ldots, s_{t+r-1})$, where each $s_{t+i} \in \mathbb{F}_q$, $i = 0, \ldots, r-1$, corresponds the contents of one memory cell. The initial state of the shift register is $S_0 = (s_0, \ldots, s_{r-1})$. An LFSR produces a sequence $s_0, s_1, \ldots$ that satisfies a *linear recurrence relation*

$$s_{t+r} = a_0 s_t + a_1 s_{t+1} + \cdots + a_{r-1} s_{t+r-1}, \quad t \geq 0,$$

where $a_0, \ldots, a_{r-1} \in \mathbb{F}_q$ are the *feedback coefficients*. The sequence $s_0, s_1, \ldots$ is uniquely determined by the linear recurrence relation and by the initial values $s_0, \ldots, s_{r-1}$. Since the recurrence relation is linear, one can form a system of linear equations from $2r$ successive sequence terms such that the unknown feedback coefficients can be uniquely solved from it. Thus, it is possible to determine the linear recurrence relation that generates the sequence. If $r$ is unknown, the Berlekamp-Massey algorithm can be used to recover the shortest linear recurrence relation that will generate the sequence. Given at least $2r$ sequence terms, the algorithm gives a unique recurrence relation; otherwise, a non-unique recurrence relation is given. The original algorithm is due to Berlekamp [1968], and its application to linearly recurrent sequences was noted by Massey [1969]. Examples of recent LFSR-based stream ciphers are SOBER-128 and the eSTREAM candidates Sfinks and WG. The most common ways of describing a linear recurrence relation of an LFSR are using a *feedback polynomial* or a *characteristic polynomial*.

- **Feedback polynomial** Describes the linear recurrence relation as a polynomial $Q$ over $\mathbb{F}_q$.

$$Q(x) = 1 - a_{r-1}x - \cdots - a_1 x^{r-1} - a_0 x^r.$$

- **Characteristic polynomial** The characteristic polynomial $Q^*$ is the reciprocal of the feedback polynomial $Q$ over $\mathbb{F}_q$.

$$Q^*(x) = x^r Q(x^{-1}) = -a_0 - a_1 x - \cdots - a_{r-1} x^{r-1} + x^r.$$

**Nonlinear Feedback Shift Registers**

Several recent stream cipher proposals use NLFSRs as building blocks. For example, Shannon and the eSTREAM candidates Grain, Trivium, Dragon, and NLS make use of NLFSRs in their structure. The sequence $s_0, s_1, \ldots$ generated by an NLFSR satisfies a nonlinear recurrence relation instead of a linear one. The advantage of NLFSRs over LFSRs is that there exists no short linear recurrence relation that is always satisfied. This property has been used to attack stream ciphers based on LFSRs. It is also used in our analysis of SOBER-128. In our analysis of Shannon, the nonlinear recurrence relation of the NLFSR is replaced with a linear one, which holds with certain probability. Even though NLFSRs have been a widely studied [see, e.g., Menezes et al., 1997], their properties are not nearly as understood as the properties of LFSRs. For example, construction of NLFSRs with guaranteed long periods remains an open problem.

## 2.3 Stream Cipher Designs

Stream ciphers are often constructed using shift registers. LFSRs are a popular choice, because they can be easily analyzed mathematically and implemented in hardware. While shift registers are efficient in hardware, they are not as efficient in software implementations. RC4 is an example of a software-oriented stream cipher that does not use shift registers, but operations that are efficient especially in software. Since LFSRs alone do not provide sufficient security, various schemes have been proposed to make LFSR-based stream ciphers more secure. Some widely used designs are *nonlinear combining functions*, *clock-controlled generators*, and *nonlinear filter generators*. Sometimes, NLFSRs are used instead of LFSRs to make certain attacks infeasible. In the following section, we give a description of nonlinear filter generators, since Shannon and SOBER-128 are both of this type.

### 2.3.1 Nonlinear Filter Generators

A nonlinear filter generator consists of a shift register and a nonlinear function that is commonly refered to as the *nonlinear filter* (NLF). Let $S_t$ denote the state of shift register at time $t \geq 0$. The state update function of the shift register works as the state update function $G$ of the stream cipher. The NLF is used as the keystream function $F$ of the stream cipher. It produces one keystream symbol $z_t$ based on the shift register state $S_t$ and the key $K$ at each time instance $t \geq 0$. The purpose of the NLF is to hide linearity of the sequence generated by the shift register.

NLFs in filter generators are essentially vector-valued Boolean functions, i.e., S-boxes. In certain ciphers, such as SOBER-128, some parts of the NLF can be viewed as T-functions. This fact is used in the analysis of SOBER-128 in this thesis.

# Chapter 3

# Linear Cryptanalysis of Stream Ciphers

Cryptanalysis is the study of methods aimed at compromising cryptosystems. Cryptanalytic methods are used to evaluate the security of a cryptosystem against certain security criteria. In cryptanalysis of symmetric encryption primitives, an adversary typically strives to obtain information of the secret key that is used in the primitive. An adversary may also have other objectives than key recovery. For example, the ability to recover the initial state of a stream cipher would be disastrous even though the secret key would not be revealed in the attack.

The security of a stream cipher is largely dependent on how random the keystream can be made to appear. To analyze this property, statistical and algebraic distinguishing attacks have been developed. *Distinguishing attacks* on stream ciphers are attacks, in which the attacker is able to tell whether a sequence has been generated by the cipher or not. The difference between distinguishing attacks and general statistical tests is that a distinguishing attack is usually formulated using the knowledge of how the keystream generator has been constructed. Statistical distinguishing attacks make often use of *linear* [Matsui and Yamagishi, 1993, Matsui, 1994] or *differential* [Biham and Shamir, 1990] cryptanalytic techniques. Linear and differential cryptanalysis are two of the most powerful statistical techniques for cryptanalysis of symmetric ciphers proposed to date, and resistance against these attacks is held as one of the most important standard design goals for current ciphers. Linear cryptanalysis studies biased *linear approximate relations* over the components of the cipher, whereas differential cryptanalysis is based on *differential propagation* through the components. Distinguishing attacks on stream ciphers can be constructed based on biased linear approximate relations involving keystream terms only. In such case, the distinguishing attack is performed by testing the approximate relation with empirical data. This type of an attack is called a *linear distinguishing attack*.

This chapter gives an overview of linear distinguishing attacks on nonlinear filter generators. The techniques in this chapter are used to analyze Shannon and SOBER-128. In Section 3.1, we discuss classification of cryptanalytic attacks in general. An introduction to distinguishing attacks and distinguishers is given in Section 3.2. In Section 3.3, we give an overview of linear distinguishing attacks and introduce the main terminology. Section 3.4 deals exclusively with linear distinguishers for filter generators. Important computational techniques are discussed in Section 3.5. As before, we use $\mathbb{F}_{2^n}$ to denote the finite field with $2^n$ elements, i.e., the integers modulo $2^n$. The integers in $\{0, \ldots, 2^n - 1\}$ are identified with the vectors $x = (x_0, \ldots, x_{n-1})$ in $\mathbb{F}_2^n$ using the natural correspondence $x \leftrightarrow \sum_{j=0}^{n-1} x_j 2^j$. For the vectors $u = (u_0, \ldots, u_{n-1}) \in \mathbb{F}_2^n$ and $x = (x_0, \ldots, x_{n-1}) \in \mathbb{F}_2^n$, we let $u \cdot x$ denote the standard inner product $u \cdot x = u_0 x_0 \oplus \cdots \oplus u_{n-1} x_{n-1} \in \mathbb{F}_2$.

## 3.1 Classification of Attacks

Cryptanalytic attacks are characterized according to how much threat they pose to a cryptosystem. We give a classification of attacks on symmetric encryption primitives in the following sections. Typically, the importance of an attack is done by considering

1. what knowledge and capabilities are needed as a prerequisite,
2. how much secret information is revealed, and
3. how much effort is required to perform the attack.

### 3.1.1 Attack Scenarios

Cryptanalysis of encryption primitives can be performed under a number of asumptions about how much can be observed about the primitive. The most common assumption is known as *Kerckhoffs' principle*, according to which the encryption algorithm is known to the attacker. Other common assumptions are:

- **Ciphertext-only** The attacker has access to a collection of ciphertexts.
- **Known-plaintext** The attacker has a set of plaintexts, for which he knows the corresponding ciphertexts.
- **Chosen-plaintext** The attacker can choose any plaintext and obtain the corresponding ciphertext.
- **Adaptive chosen-plaintext** A chosen-plaintext attack, in which the attacker chooses plaintexts based on the previously obtained ciphertexts.

For the latter three attack scenarios, there exists corresponding attacks, in which the assumptions are made for ciphertext instead of plaintext. In addition to these attack scenarios, other scenarios, such as *chosen-IV* and *known-IV*, are also used.

### 3.1.2 Success of the Attack

The main objective in the cryptanalysis of encryption primitives is recovery of the secret key, since this makes it possible to decrypt any messages encrypted with the same key. There exists also several other attacks, which might give important information to the adversary without revealing the entire secret key. Knudsen [1999] classified various attacks on block ciphers according to the amount and quality of previously unknown information that the attack reveals:

- **Total break** The attacker recovers the secret key.
- **Global deduction** The attacker finds an algorithm equivalent to encryption and decryption without learning the secret key.
- **Instance deduction** The attacker is able to produce previously unknown plaintexts (or ciphertexts).
- **Information deduction** The attacker is able to gain previously unknown (Shannon) information about the secret key, the $IV$, the plaintexts or the ciphertexts.
- **Distinguishing algorithm** The attacker is able to detect statistical anomalies that should not be present in the cipher should by applying an algorithm.

This classification is hierarchial, i.e., total break allows global deduction, global deduction allows instance deduction, and so on. In attacks on stream ciphers, one could classify internal state recovery as instance deduction and initial state recovery as global deduction. The focus of the thesis is on distinguishing attacks, but information deduction from the secret key is also studied.

### 3.1.3 Complexity of the Attack

Another characterization of attacks is based on the resources they require:

- **Time** The number of primitive operations that are needed to execute the attack.
- **Memory** The amount of storage required to perform the attack.

- **Data** The amount of data (e.g. plaintexts, ciphertexts, or keystream) required for the attack.

For the attacks presented in this thesis, we hold data complexity as the most important indicator for the computational complexity. A distinguishing attack is commonly considered successful, if the keystream can be distinguished from $2^{|K|}$ keystream terms, where $|K|$ is the bit-length of the key $K$.

## 3.2 Distinguishing Attacks

Distinguishing attacks are attacks in which an adversary tries to determine whether a sequence has been produced by a specific cipher or seems to be a random sequence. Distinguishing attacks can be applied to both block ciphers and stream ciphers. They are used to detect statistical anomalies in the primitive; however, they may help in key recovery in some cases. Linear or differential cryptanalytic techniques are widely used to construct a distinguishing attack for a specific cipher. A distinguishing attack with a very high complexity may not pose a threat in itself, but indicates a weakness in the primitive.

A *statistical distinguisher* is essentially an implementation of a *statistical hypothesis test*. It can be viewed as a function that takes a sequence as input and outputs either cipher or random. A distinguisher should be able to give the correct answer at high confidence level. Denote by $\mathcal{D}_U$ the uniform distribution and by $\mathcal{D}_C$ the distribution of a sample sequence generated by the cipher. Let $x_0, \ldots, x_{N-1}$ be a realization of random variables $X_0, \ldots, X_{N-1}$ from an unknown distribution $\mathcal{D}_X$. A distinguisher performs a hypothesis test, where it decides whether the sequence $x_0, \ldots, x_{N-1}$ is a sample from $\mathcal{D}_C$ or $\mathcal{D}_U$. In other words, it tries to determine if the distribution $\mathcal{D}_X$ is more likely to be $\mathcal{D}_C$ or $\mathcal{D}_\mathcal{U}$. Usually, distinguishers can be divided into two classes: *general distinguishers* and *cipher-specific distinguishers*.

- **General distinguisher** A general distinguisher does not consider the internal structure of the cipher. The cipher is viewed as a black box that outputs a keystream, whose randomness properties are examined. General distinguishers are useful especially for cipher designers who want to examine statistical properties of the cipher. Examples of general distinguishers are the NIST statistical test suite, the Diehard tests, and Crypt-X. These distinguishers include several statistical tests, which evaluate randomness properties of sequences.

- **Cipher-Specific Distinguisher** A cipher-specific distinguisher uses knowledge of the internal structure of the cipher to decide whether the sequence has been generated by the cipher or is a random sequence.

The distinguishers in this class operate in two phases: the input sequence is transformed in some way and the resulting sequence is then fed into the statistical inference part, which makes the final decision.

A cipher-specific distinguisher can be constructed by finding a relation that results in biased samples. The relation is usually achieved by examining the inner structure of the cipher. In linear cryptanalysis, one uses linear functions to approximate all nonlinear parts in the cipher, in which case the final relation will be a biased linear relation. The goal is often to find a relation that holds with as high probability as possible. Biased linear relations are usually called *linear distinguishers*. Cipher-specific distinguishers based on general statistical tests also have been developed [see, e.g., Vaudenay, 1996].

## 3.3 Linear Distinguishing Attacks

Linear cryptanalysis [Matsui and Yamagishi, 1993, Matsui, 1994] is a general form of statistical cryptanalysis based on finding linear approximate relations over the nonlinear components of the cipher. The idea of linear cryptanalysis for block ciphers was introduced by Matsui and Yamagishi [1993] in an attack on FEAL. The technique was then refined and applied on DES by Matsui [1994]. Let $f_K\colon \mathbb{F}_2^n \to \mathbb{F}_2^n$ be a Boolean function that depends on a key $K \in \mathbb{F}_2^n$, and let $u, v, w \in \mathbb{F}_2^n$ be vectors. Linear cryptanalysis for block ciphers is a known-plaintext attack, in which one examines linear approximate relations of form

$$v \cdot f_K(x) \oplus u \cdot x = w \cdot K,$$

that hold with certain probability. The intention is usually to find constants $u, v, w \in \mathbb{F}_2^n$ such that this relation holds (or does not hold) with high probability. This allows finding out whether $w \cdot K$ equals 0 or 1, when enough sample pairs $(x, f_K(x))$ have been given. Thus, we get one bit of information of $K \in \mathbb{F}_2^n$.

Several generalizations of linear cryptanalysis of block ciphers have been presented since its introduction. For example, Kaliski and Robshaw [1994] gave a form of linear cryptanalysis using multiple linear approximate relations concurrently. This has also been examined by Biryukov et al. [2004] and by Baignères et al. [2004]. Linear statistical distinguishers on stream ciphers were introduced by Golić [1995]. This work is based on an algorithm for finding nonbalanced linear functions of the keystream, which is called *linear sequential circuit approximation* and was also introduced by Golić [1993]. Linear distinguishing attacks can be considered to be a technique of linear cryptanalysis, since one uses linear approximate relations to form linear distinguishers. If the distribution of the nonbalanced linear functions of the keystream are key-dependent, then one gets information of the key

similarly as with block ciphers. Otherwise, linear distinguishing attacks can be used to assess randomness properties of the stream cipher. Linear cryptanalysis has been succesfully applied to distinguish the output keystream from a truly random sequence [see, e.g., Coppersmith et al., 2002, Watanabe et al., 2004, Nyberg and Wallén, 2006]. In this thesis, the focus is on linear cryptanalysis as the method for distinguishing output sequences of nonlinear filter generators.

This concept of linear approximations is formalized in Section 3.3.1. Linear approximations of iterated vector-valued Boolean functions is discussed in Section 3.3.2.

### 3.3.1 Linear Approximations

Let $n$ and $m$ be positive integers. In this thesis, we consider a component of the cipher to be a mapping $f\colon \mathbb{F}_2^{m\times n} \to \mathbb{F}_2^n$, i.e., a mapping that takes $m$ $n$-bit input words and maps them to a single $n$-bit output word. A component can also be written as a univariate mapping, where the input words have been concatenated such that the input is a single $mn$-bit word. A constant vector or matrix that is used to select what input (output) bits will be used in a linear approximate relation of a function is called a *linear input (output) mask* of the function. A *linear approximation* of a functional dependency $f\colon \mathbb{F}_2^{m\times n} \to \mathbb{F}_2^n$ is a relation of the form

$$v \cdot f(x) = \bigoplus_{i=0}^{m-1} u^{(i)} \cdot x^{(i)},$$

where the row vectors $u^{(0)}, \ldots, u^{(m-1)} \in \mathbb{F}_2^n$ are the linear input masks for the input words and $v \in \mathbb{F}_2^n$ is the linear output mask. The linear input mask for $f$ is the matrix $u = (u_{i,j}) \in \mathbb{F}_2^{m\times n}$ with $u^{(0)}, \ldots, u^{(m-1)}$ as the rows. The efficiency of a linear approximation of $f$ is measured by its *correlation*

$$c_f(v,u) = 2\Pr\left[v \cdot f(x) = \bigoplus_{i=0}^{m-1} u^{(i)} \cdot x^{(i)}\right] - 1,$$

where the probability is taken over the uniformly distributed $x \in \mathbb{F}_2^{m\times n}$. We use $\epsilon_f(v,u) = c_f(v,u)/2$ to denote the *bias* of a linear approximation of $f$. The linear approximation of $f$ with the input mask $u$ and the output mask $v$ is denoted with the tuple $(v,u) \in \mathbb{F}_2^n \times \mathbb{F}_2^{m\times n}$. A comma is used for separating the output mask to the left and the input mask(s) to the right. Given a linear mask $u \in \mathbb{F}_2^n$ and an element $a \in \mathbb{F}_2^n$, we denote by $ua$ the linear mask, which satisfies the equality

$$ua \cdot x = u \cdot ax, \quad \text{for all } x \in \mathbb{F}_2^n,$$

where the products $ua$ and $ax$ are taken in $\mathbb{F}_{2^n}$.

### 3.3.2 Linear Chains

In linear cryptanalysis, the aim is generally to find linear approximations of iterated mappings with high bias. Let $f = f_{N-1} \circ \cdots \circ f_0$ be an iterated mapping, where each $f_i$ is a function between vector spaces over $\mathbb{F}_2$, $f_i \colon \mathbb{F}_2^{n_i} \to \mathbb{F}_2^{n_{i+1}}$, $i = 0, \ldots, N-1$. Denote by $c_{f_i}(u_{i+1}, u_i)$ the correlation of a linear approximation of $f_i$ with the output mask $u_{i+1} \in \mathbb{F}_2^{n_{i+1}}$ and the input mask $u_i \in \mathbb{F}_2^{n_i}$. A *linear chain* is a linear approximation of $f$ such that the correlation is determined from invidual linear approximations $(u_{i+1}, u_i)$ of $f_i$. The correlation $c_f$ of a linear chain is defined to be

$$c_f = \prod_{i=0}^{N-1} c_{f_i}(u_{i+1}, u_i).$$

This is actually an estimate of the true correlation as we will show next. Let $g \colon \mathbb{F}_2^{n_0} \to \mathbb{F}_2^{n_1}$ and $h \colon \mathbb{F}_2^{n_1} \to \mathbb{F}_2^{n_2}$ be Boolean functions, and let $u \in \mathbb{F}_2^{n_0}$ and $v \in \mathbb{F}_2^{n_2}$ be linear masks. Using a framework based on the Walsh-Hadamard transform, Daemen et al. [1995] showed that the correlation of a linear approximation $(v, u)$ of $h \circ g$ is

$$c_{h \circ g}(v, u) = \sum_{w \in \mathbb{F}_2^{n_1}} c_h(v, w) c_g(w, u). \tag{3.1}$$

Denote $u = u_0$ and $v = u_N$. For iterated mappings $f = f_{N-1} \circ \cdots \circ f_0$, it follows that

$$c_f(v, u) = \sum_{u_1, \ldots, u_{N-1}} \prod_{i=0}^{N-1} c_{f_i}(u_{i+1}, u_i).$$

If the sum is dominated by a single linear chain with the masks $u_0, \ldots, u_N$, one can estimate that

$$c_f(u_N, u_0) \approx \prod_{i=0}^{N-1} c_{f_i}(u_{i+1}, u_i). \tag{3.2}$$

This estimate should be interpreted carefully, since several linear chains contribute to the same linear approximations, some with negative and some with positive correlation. Also, if the iterated mapping depends on a constant, then the correlation may change with the constant. We give a few examples of this further in the thesis. One can get information from a secret constant, if it is known how different constant values affect the correlation. This is used in our analysis of SOBER-128. In addition to (3.1), there exist other explicit formulas for the correlation of some linear approximations [see, e.g., Nyberg, 2001]. Further in this thesis, we show how one can derive an explicit expression for $c_f(v, u)$ of certain linear approximations $(v, u)$, when $f$ is a T-function.

The estimate (3.2) can also be concluded from the *Piling-Up Lemma* [Matsui, 1994]. Suppose that $X_0, \ldots, X_{N-1}$ are independent binary random variables such that $\Pr[X_i = 0] = \frac{1}{2} + \epsilon_i$, $i = 0, \ldots, N-1$. Denote $\Pr[X_0 \oplus \cdots \oplus X_{N-1} = 0] = \frac{1}{2} + \epsilon$. The Piling-Up Lemma states that

$$\epsilon = 2^{N-1} \prod_{i=0}^{N-1} \epsilon_i. \tag{3.3}$$

Now assume that each linear approximation $(u_{i+1}, u_i)$ of $f_i$ is statistically independent of other approximations and denote $X_i = u_{i+1} \cdot f_i(x) \oplus u_i \cdot x$. It follows that $\epsilon_i = \epsilon_{f_i}(u_{i+1}, u_i)$, and using the Piling-Up Lemma we get

$$\epsilon = 2^{N-1} \prod_{i=0}^{N-1} \frac{1}{2} c_{f_i}(u_{i+1}, u_i) = \frac{1}{2} \prod_{i=0}^{N-1} c_{f_i}(u_{i+1}, u_i).$$

The estimate (3.2) follows by denoting $\epsilon = c_f(u_{N+1}, u_1)/2$.

## 3.4 Linear Distinguishers for Filter Generators

In this section, we discuss linear distinguishers for a nonlinear filter generator that consists of a shift register and an NLF $F$. Suppose that the shift register has $r$ memory cells with elements from the finite field $\mathbb{F}_{2^n}$. Let $S_t = (s_t, \ldots, s_{t+r-1}) \in \mathbb{F}_{2^n}^r$ denote the state of the shift register at time $t \geq 0$, and suppose that the NLF $F$ is a function of the keystream state $S_t$ and a secret key $K$ such that $F\colon \mathbb{F}_{2^n}^r \times \mathbb{F}_{2^n} \to \mathbb{F}_{2^n}$. The output of the generator at time $t \geq 0$ is denoted by $z_t = F(S_t, K)$. In basic linear distinguishing attacks on a filter generator, one studies linear approximate relations of the form

$$\bigoplus_{j \in J} v_j \cdot z_{t+j} = 0, \quad t \geq 0, \tag{3.4}$$

where $v_j \in \mathbb{F}_2^n$ is the linear mask used in the approximation of the output word $z_{t+j} \in \mathbb{F}_2^n$, and $J$ is the index set that defines which output words are included in the approximation. The probability that (3.4) holds is a conditional probability taken over uniform $S_{t+j} \in \mathbb{F}_{2^n}^r$, for all $j \in J$, given a constant $K \in \mathbb{F}_{2^n}$. For simplicity, we will first discuss linear distinguishers, where the output $z_t$ is independent of the value of $K$. This assumption leads to distinguishers that do not give information of $K$. A linear distinguisher for a filter generator operates in two phases. In this thesis, we refer to these phases as the *transformation phase* and the *statistical inference phase*.

1. **Transformation phase** In the transformation phase, one applies a transformation to the input sequence $z_0, z_1, \ldots$ to get a new sequence

$\hat{z}_0, \hat{z}_1, \ldots$. With linear distinguishers for filter generators, the following linear transformation is used:

$$\hat{z}_t = \bigoplus_{j \in J} v_j \cdot z_{t+j}, \quad t \geq 0,$$

which is the same transformation as in the linear approximation (3.4).

2. **Statistical inference phase** In the statistical inference phase, a statistical hypothesis test is performed to the sequence $\hat{z}_0, \hat{z}_1, \ldots, \hat{z}_{N-1}$ in order to decide whether the input sequence $\hat{z}_0, \hat{z}_1, \ldots, \hat{z}_{N-1}$ is from the cipher or appears to be a random sequence. If the sequence $z_0, z_1, \ldots$ is from the filter generator, then the sequence $\hat{z}_0, \hat{z}_1, \ldots, \hat{z}_{N-1}$ has—at least in theory—a bias that is close to the bias of the linear approximate relation (3.4). The hypothesis test makes the decision based on a test statistic, which is usually a function of the biases. In this thesis, we use the *log-likelihood ratio* statistic so that the number $N$ of input sequence terms required to make the decision at high confidence level is inversely proportional to the square of the bias $\epsilon$ of (3.4), i.e., $\mathcal{O}(\epsilon^{-2})$. This is a common choice for evaluating how efficient a linear distinguishing attack is. The exact number of needed samples depends on how accurate we want the distinguisher to be.

The focus of this thesis is on the transformation phase, i.e., on constructing efficient transformations for the keystream sequence $z_0, z_1, \ldots$ and estimating its data-complexity. In order to construct an efficient distinguisher, one needs to find a linear approximation for $z_0, z_1, \ldots$ such that the approximation has a large bias $|\epsilon|$. Linear approximations for the keystream can be formed by constructing a linear approximation for the nonlinear filter $F$ and using a time-invariant relation to cancel out the input variables of the approximation of $F$. In Sections 3.4.1 and 3.4.2, we show how to do this for nonlinear filter generators with linear and nonlinear feedback. In these sections, we assume that the output $z_t$ is independent of the key $K$, i.e., $z_t = F(S_t)$ and $s_{t+r} = G(S_t)$, for $t \geq 0$. In Section 3.4.3, we discuss distinguishers that make use of multiple linear approximations concurrently. In Section 3.4.4, we consider the case, in which the output $z_t$ depends on the key $K$.

### 3.4.1 Filter Generators with Linear Feedback

We use the recurrence relation of an LFSR as the time-invariant relation that is used to cancel out the input variables to an approximation of the NLF. Recall that the linear recurrence relation of an LFSR with the state $S_t = (s_t, \ldots, s_{t+r-1}) \in \mathbb{F}_{2^n}^r$ can be written as

$$a_0 s_t \oplus a_1 s_{t+1} \oplus \cdots \oplus a_{r-1} s_{t+r-1} \oplus a_r s_{t+r} = 0, \quad t \geq 0, \tag{3.5}$$

where $a_0, \ldots, a_{r-1} \in \mathbb{F}_{2^n}$, $a_r = 1$, and the product $a_i s_{t+i}$ is taken in $\mathbb{F}_{2^n}$ for $i = 0, \ldots, r$. We assume that the elements of the state $S_t$ have uniform distribution and are statistically independent for all $t \geq 0$. Let $0 \leq j \leq r$ and denote by

$$v_j \cdot z_{t+j} = \bigoplus_{i=0}^{r-1} u^{(i)} a_j \cdot s_{t+j+i} \tag{3.6}$$

a linear approximation of $z_{t+j} = F(S_{t+j})$ with the output mask $v_j \in \mathbb{F}_2^n$ and the input masks $u^{(0)}a_j, \ldots, u^{(r-1)}a_j \in \mathbb{F}_2^n$. If $a_j = 0$, we can choose $v_j = 0$. Summing up the approximations (3.6) for $j = 0, \ldots, r$ gives

$$\bigoplus_{j=0}^{r} v_t \cdot z_{t+j} = \bigoplus_{j=0}^{r} \bigoplus_{i=0}^{r-1} u^{(i)} a_j \cdot s_{t+j+i}.$$

Since $u^{(i)}a_j \cdot x = u^{(i)} \cdot a_j x$, for all $x \in \mathbb{F}_{2^n}$, it follows that

$$\bigoplus_{j=0}^{r} v_j \cdot z_{t+j} = \bigoplus_{i=0}^{r-1} u^{(i)} \cdot \left[ \bigoplus_{j=0}^{r} a_j s_{t+j+i} \right] = 0. \tag{3.7}$$

The last equivalence holds, since $\bigoplus_{j=0}^{r} a_j s_{t+j+i} = 0$ is the recurrence relation (3.5) at time $t := t + i$. Denote the correlation of the approximation (3.6) by $c_F(v_j, u_j)$, where $u_j = (u^{(0)}a_j, \ldots, u^{(r-1)}a_j)$. The final approximation (3.7) is formed by taking the exclusive-or of the binary random variables $v_j \cdot z_{t+j}$, $j = 0, \ldots, r$. Assuming that these random variables are statistically independent, the correlation $c$ of (3.7) can be estimated with the Piling-Up Lemma (3.3) as

$$c \approx \prod_{j=0}^{r} c_F(v_j, u_j),$$

which is the same value for all $t \geq 0$. The correlation $c$ is stronger the less is the number of nonzero coefficients $a_j$ in the recurrence relation, because $c_F(v_j, u_j) = 1$, for $v_j = 0$ and $u_j = (0, \ldots, 0)$. At most, $r + 1$ nonzero masks $(v_j, u_j)$ are needed. If we denote by $J \subseteq \{0, \ldots, r\}$ the index set that defines which coefficients $a_j$ are nonzero, the linear approximation (3.7) can be written as (3.4).

### 3.4.2 Filter Generators with Nonlinear Feedback

Generating a linear distinguisher for a filter generator with nonlinear feedback is similar as with linear feedback. Cho and Pieprzyk [2006b] called this type of linear distinguishing attacks as *crossword-puzzle attacks*. Denote by $G$ the nonlinear state update function and recall that the recurrence relation of the NLFSR can be written as $s_{t+r} = G(S_t)$, $t \geq 0$. Suppose that—as

in the previous section—we have a linear approximation of $z_{t+j} = F(S_{t+j})$ with the output mask $v_j \in \mathbb{F}_2^n$ and the input masks $u^{(0)}a_j, \ldots, u^{(r-1)}a_j \in \mathbb{F}_2^n$. Since the linear relation $\bigoplus_{j=0}^{r} a_j s_{t+j+i} = 0$ does not hold with nonlinear feedback, we form a linear approximation

$$u^{(i)} \cdot \left[ \bigoplus_{j=0}^{r} a_j s_{t+j+i} \right] = 0, \tag{3.8}$$

for $i = 0, \ldots, r-1$, in order to derive the final approximation $\bigoplus_{j=0}^{r} v_j \cdot z_{t+j} = 0$ as in (3.7). By approximating $G(S_{t+i})$ with $u^{(i)}a_r$ as the output mask and $u^{(i)}a_0, \ldots, u^{(i)}a_{r-1}$ as the input masks we get

$$u^{(i)}a_r \cdot s_{t+r+i} = \bigoplus_{j=0}^{r-1} u^{(i)}a_j \cdot s_{t+j+i}, \tag{3.9}$$

which is equivalent to the approximation (3.8). Hence, by forming suitable linear approximations for the nonlinear state update function, one can construct a linear distinguisher for a filter generator with nonlinear feedback. Since the approximation (3.9) has been used to derive the final approximate relation, the correlation of the distinguisher gets additional terms compared to the correlation in the linear case. Denote the correlation of (3.9) by $c_G(u^{(i)}, \hat{u}^{(i)})$, where $\hat{u}^{(i)} = (u^{(i)}a_0, \ldots, u^{(i)}a_{r-1})$. The final approximation is formed by taking the exclusive-or of the binary random variables $u^{(i)} \cdot \left[ \bigoplus_{j=0}^{r} a_j s_{t+j+i} \right]$, $i = 0, \ldots, r-1$, and $v_j \cdot z_{t+j} \oplus \bigoplus_{i=0}^{r-1} u^{(i)}a_j \cdot s_{t+j+i}$, $j = 0, \ldots, r$. Assuming that these random variables are independent, the correlation $c$ can be estimated with the Piling-Up Lemma (3.3) as

$$c \approx \left[ \prod_{j=0}^{r} c_F(v_j, u_j) \right] \times \left[ \prod_{i=0}^{r-1} c_G(u^{(i)}, \hat{u}^{(i)}) \right].$$

### 3.4.3 Multiple Linear Approximations

It is possible to improve a linear distinguishing attack by using multiple linear approximations concurrently. Let $s$ and $r$ be positive integers such that $s \leq r$. In a distinguishing attack with $s$ linear approximations, one studies a system of approximate relations

$$\begin{cases} \bigoplus_{j=0}^{r-1} v_{0,j} \cdot z_{t+j} = 0, \\ \qquad\qquad \vdots \\ \bigoplus_{j=0}^{r-1} v_{s-1,j} \cdot z_{t+j} = 0, \end{cases}$$

where $z_0, z_1, \ldots$ is a sequence over $\mathbb{F}_2^n$ and $v_{i,j}$ is a linear mask for $j = 0, \ldots, r-1$, $i = 0, \ldots, s-1$. Thus, by applying the transformation of the

distinguisher to the input sequence $z_0, z_1, \ldots$ we obtain a sequence of tuples $\hat{Z}_t = (\hat{z}_{0,t}, \ldots, \hat{z}_{s-1,t})$, $t = 0, \ldots, N-1$, where $\hat{z}_{i,t}$ is defined as

$$\hat{z}_{i,t} = \bigoplus_{j=0}^{r-1} v_{i,j} \cdot z_{t+j},$$

for $i = 0, \ldots, s$. The distribution of the sequence $\hat{Z}_0, \ldots, \hat{Z}_{N-1}$ is studied in the statistical inference phase to decide whether the input sequence $z_0, z_1, \ldots$ is from the cipher or appears to be random. Denote $w = (w_0, \ldots, w_{r-1}) \in (\mathbb{F}_2^n)^r$ and let $c(w)$ be the correlation of a linear approximation $\bigoplus_{j=0}^{r-1} w_j \cdot z_{t+j} = 0$. According to Baignères et al. [2004], if the input sequence is from the cipher, the sequence $\hat{Z}_0, \ldots, \hat{Z}_{N-1}$ should have a variance close to $2^{-s} \sum_w c(w)^2$, where the sum is taken over all nonzero linear combinations $(w_0, \ldots, w_{r-1})$ of the mask tuples $(v_{i,0}, \ldots, v_{i,r-1})$, $i = 0, \ldots, s-1$. If $s = r$, the sum is taken over nonzero $2^r - 1$ mask tuples. Yet again, we use the log-likelihood ratio as the test statistic for comparing the variances, so the number $N$ of required samples to make decision reliably is inversely proportional to the *squared Euclidean imbalance*, i.e., $\mathcal{O}(1/\sum_w c(w)^2)$. This requirement can be a significant improvement over a distinguisher with one linear approximation, which needs $\mathcal{O}(1/c(w)^2)$ samples. If the distribution of the correlations is uniform, we have $\sum_w c(w)^2 = (2^s - 1)c(w)^2$, and hence the distinguisher with $r$ linearly independent masks needs approximately $2^r$ times less samples compared to a distinguisher that relies on one linear approximation. For further details of using multiple statistically dependent linear approximations and the log-likelihood ratio, we refer to Baignères et al. [2004].

### 3.4.4 Constants in the Nonlinear Filter

Suppose that the output $z_t$ of the NLF $F$ is given by $z_t = F(S_t, K)$, where $K \in \mathbb{F}_{2^n}$ is a secret constant. In this case, the linear approximation (3.4) involving keystream variables holds with a probability conditional on $K$. If these probabilities are known for all $K \in \mathbb{F}_{2^n}$, it is possible to gain information about the value of $K$ in the linear distinguishing attack. The transformation phase in such attacks is done as usual to obtain a sequence $\hat{z}_0, \ldots, \hat{z}_{N-1}$. The bias of this sequence is compared to the conditional biases of approximations (3.4) for all $K \in \mathbb{F}_{2^n}$ using the log-likelihood ratio statistic. Given enough empirical data, one can determine the constants $K$, which are likely to have been used in $F$ to generate the original keystream sequence.

It is possible that the conditional bias of (3.4) is the same for multiple constants $K$. One can form an equivalence class for each bias value consisting of those constants that induce the same bias. Another possible classification is to classify constants based on the sign of the bias such that constants with

positive, negative, and zero biases are put in separate classes. Constant classes can be described as relations on the constant bits. In the linear distinguishing attack on SOBER-128, we show that the constants $K = (k_0, \ldots, k_{n-1}) \in \mathbb{F}_2^n$ are divided into two classes according to a linear relation such that the linear approximation (3.4) can be rewritten as

$$\bigoplus_{j \in J} v_j \cdot z_{t+j} = w \cdot K, \quad t \geq 0.$$

This attack is comparable to the basic linear cryptanalytic attack due to Matsui [1994]. In Section 4.2.4, we give an example of a linear approximation of the function $f_K(x) = ((x^{(0)} \boxplus x^{(1)}) \oplus K) \boxplus x^{(2)}$, for which the constant classes are defined by nonlinear relations on $K$. From Definition 1, it is easy to see that the following theorem regarding linear approximations of T-functions holds.

**Theorem 1.** *Let $(v, u)$ be a linear approximation of a T-function $f\colon \mathbb{F}_2^{m \times n} \to \mathbb{F}_2^n$ that contains a constant $K \in \mathbb{F}_2^n$. Denote the number of most significant bits that are zeros in all masks $v, u^{(0)}, \ldots, u^{(m-1)}$ by $n_0$, and use $\mathcal{K}$ to denote the set of all constant classes. The number of constant classes is $1 \leq |\mathcal{K}| \leq 2^{n-n_0}$. Moreover, the constants, whose $n - n_0$ least significant bits are the same, have the same correlation.*

## 3.5 Computational Techniques

It is generally a difficult task to find useful linear approximations $(v, u) \in \mathbb{F}_2^n \times \mathbb{F}_2^{m \times n}$ of an arbitrary Boolean function $f\colon \mathbb{F}_2^{m \times n} \to \mathbb{F}_2^n$. Moreover, finding the best linear approximation is even more difficult. Some computational techniques have been established to make these tasks easier. For example, Wallén [2003] presented a linear time algorithm for computing the correlation of a linear approximation of addition modulo $2^n$. In addition, he presented an optimal algorithm for generating all linear approximations for a given nonzero correlation coefficient. An efficient algorithm for computing the correlation of a linear approximation of addition modulo $2^n$ with several inputs was presented by Nyberg and Wallén [2006]. With respect to differential cryptanalysis, similar results have been established. Lipmaa and Moriai [2001] and Lipmaa [2002] examined exclusive-or differential properties of addition. Additive differential properties of exclusive-or have been examined by Lipmaa [2004].

In this section, we discuss two techniques that make searching for useful linear approximations of the nonlinear parts of the cipher easier. The first topic is linear approximations of the form $(v, 0)$, i.e., approximations with zero as the input mask. Another topic is linear approximations of $\omega$-narrow T-functions. We generalize the technique shown by Nyberg and Wallén

[2006] and present an efficient algorithm for determining the correlation of an $\omega$-narrow T-function with small $\omega$.

### 3.5.1 The Walsh-Hadamard Transform

The *Walsh-Hadamard transform* (WHT) is a transform that is often used to examine properties of Boolean functions. Techniques based on the WHT can also be used to reduce the amount of computational work in some problems. In this thesis, it is used to facilitate the search of useful linear approximations.

**Definition 4.** Given a mapping $f\colon \mathbb{F}_2^{mn} \to \mathbb{R}$ the WHT of $f$ is a real-valued function $\mathcal{F}(f)\colon \mathbb{F}_2^{mn} \to \mathbb{R}$ defined as

$$\mathcal{F}(f)(u) = \sum_{x \in \mathbb{F}_2^{mn}} f(x)(-1)^{u \cdot x}, \quad u \in \mathbb{F}_2^{mn}. \tag{3.10}$$

The WHT is easily inverted. Given the transform $F(u) = \mathcal{F}(f)(u)$ for all $u \in \mathbb{F}_2^{mn}$ the values of $f$ can be determined from the inverse transform

$$f(x) = 2^{-mn} \sum_{u \in \mathbb{F}_2^{mn}} F(u)(-1)^{u \cdot x}, \quad x \in \mathbb{F}_2^{mn}. \tag{3.11}$$

Let $f\colon \mathbb{F}_2^{mn} \to \mathbb{F}_2^n$ be a Boolean function, and consider a linear approximation

$$v \cdot f(x) = 0,$$

where $v \in \mathbb{F}_2^n$ is the linear output mask. Suppose we want to find out the linear approximation $(v, 0)$, $v \neq 0$, with the highest bias. Since there are $2^n - 1$ choices for $v$ and $2^{mn}$ values for $x$, it takes about $2^{n(m+1)}$ steps to find out the bias for each linear mask $v \in \mathbb{F}_2^n$, $v \neq 0$, using the naïve technique. The amount of computational work can be reduced using the *fast Walsh-Hadamard transform* (FWHT) as follows. Denote by $p(y)$ the probability that $f(x) = y$ taken over $x \in \mathbb{F}_2^{mn}$. The correlation of the approximation can be written as

$$\begin{aligned} c_f(v, 0) &= \Pr[v \cdot f(x) = 0] - \Pr[v \cdot f(x) \neq 0] \\ &= \sum_{\substack{y \in \mathbb{F}_2^n \\ v \cdot y = 0}} p(y) - \sum_{\substack{y \in \mathbb{F}_2^n \\ v \cdot y \neq 0}} p(y) = \sum_{y \in \mathbb{F}_2^n} p(y)(-1)^{v \cdot y} \end{aligned}$$

Hence, the correlation $c_f(v, 0)$ is given by the transform $\mathcal{F}(p)$ on point $v \in \mathbb{F}_2^n$. The FWHT of $f$ requires about $mn2^{mn}$ computations in general; so determining $\mathcal{F}(p)(v)$, for all $v \in \mathbb{F}_2^n$, can be done in $n2^n$ computations. It takes $2^{mn}$ steps to determine the value distribution of $f$, and thus $2^{mn} + n2^n$ computations are needed to determine $c_f(v, 0)$, for all $v \in \mathbb{F}_2^n$. Significant

improvements to the computation time are thus achieved. Note that this technique can also be applied with the same complexity to compute the correlation of all linear approximations, which use the same mask $v$ to mask the input and output of $f$. In these cases, the approximation can be rewritten as

$$v \cdot (f(x) \oplus x^{(0)} \oplus \cdots \oplus x^{(m-1)}) = 0,$$

for which we can compute the correlation using WHT.

### 3.5.2 Linear Approximations of T-functions

Let $f\colon \mathbb{F}_2^{m\times n} \to \mathbb{F}_2^n$ be a T-function with narrowness $\omega$ and denote by $x \in \mathbb{F}_2^{m\times n}$ the input matrix that consists of $m$ $n$-bit input words. We denote by $u \in \mathbb{F}_2^{m\times n}$ the input mask and by $v \in \mathbb{F}_2^n$ the output mask for a linear approximation of $f$. We suppose that the narrowness of $f$ is not dependent on $n$ and that the parameter $\alpha_j$ for $f$ is the same function for $j = 1, \ldots, n-1$.

Recursive evaluation of an $\omega$-narrow T-function can be viewed as a Markov chain. We generalize the technique presented in Nyberg and Wallén [2006] and make use of the Markov property to generate a set of substochastic transition matrices for the linear approximation $(v, u)$ of $f$. This set of matrices allows us to compute the correlation $c_f(v, u)$ by $n$ matrix multiplications with small matrices instead of the naïve approach that would always take at least $2^{m+n}$ steps to complete. For $j = 0, \ldots, n-1$, there is a transition matrix (or correlation matrix), whose elements represent probabilities $\Pr[v_j f_j(x_j, \alpha_j) = \bigoplus_{i=0}^{m-1} u_{i,j}x_{i,j},\ \alpha_{j+1} = d \mid \alpha_j = c]$ over uniformly distributed $x_j$ given the linear approximation $(v, u)$ of $f$. These matrices enable us to determine the correlation of a linear approximation by going through the tuples $(f_j, v_j, u_{m-1,j}, \ldots, u_{0,j})$, $j = 0, \ldots, n-1$, in order and making a matrix multiplication in each step with the matrix that represents the current transition. The set of correlation matrices and the two vectors that are used for calculating the correlation for a certain linear approximation is called the *linear representation* [Nyberg and Wallén, 2006] of the correlation. In the next section, we show how these matrices are formed for all linear approximations $(v, u)$ of $f$.

#### Linear Representation

Suppose that $f$ has $k$ different functions $f_j$ and $\alpha_j$ is the same function for $j > 0$. We identify every component $f_j$ with a unique number in $\{0, \ldots, k-1\}$. Each tuple $(f_j, v_j, u_{m-1,j}, \ldots, u_{0,j})$ is written as a character $b_j$ that belongs to the alphabet $\{0, \ldots, k2^{m+1} - 1\}$, where $b_j = f_j 2^{m+1} + v_j 2^m + \sum_{i=0}^{m-1} u_{i,j} 2^i$. We will show that there are at most $k2^{m+1}$ $2^\omega \times 2^\omega$ matrices over rationals, a row vector $L$ and a column vector $C$ such that

$$c_f(v, u) = LA_{b_{n-1}} \cdots A_{b_1} A_{b_0} C$$

for all linear approximations $(v, u)$ of $f$ with $m$ $n$-bit input words. We say that the matrices $L, A_{b_{n-1}}, \ldots, A_{b_1}, A_{b_0}, C$ form a linear representation of the correlation with the dimension $2^\omega$.

**Theorem 2.** *Let $f\colon \mathbb{F}_2^{m\times n} \to \mathbb{F}_2^n$ be a T-function with narrowness $\omega$ that has $k$ different functions $f_j$. Let $L = (1, 1, \ldots, 1)$ and $C = (1, 0, \ldots, 0)^T$ be row and column vectors of dimension $2^\omega$ respectively. For each $r \in \{0, \ldots, k2^{m+1} - 1\}$ define a $2^\omega \times 2^\omega$ matrix $A_r$ such that*

$$(A_r)_{d,c} = 2^{1-m}|\{\hat{x} \in \mathbb{F}_2^m \mid \hat{v}f_j(\hat{x}; c) = \hat{u} \cdot \hat{x},\ \alpha(\hat{x}; c) = d\}| - 1,$$

*where $r = f_j 2^{m+1} + \hat{v}2^m + \sum_{i=0}^{m-1} \hat{u}_i 2^i$, $j \in \{0, \ldots, k-1\}$, $\hat{v} \in \mathbb{F}_2$, $\hat{u} \in \mathbb{F}_2^m$, and $c, d \in \{0, \ldots, 2^\omega - 1\}$. Let $(v, u)$ be a linear approximation of $f$. Let $b = b_{n-1} \ldots b_1 b_0$ be the word associated with the approximation. We then have*

$$c_f(v, u) = LA_{b_{n-1}} \cdots A_{b_1} A_{b_0} C.$$

*Proof.* The proof is essentially the same as in Nyberg and Wallén [2006]. Denote by $x = (x_{i,j})$ the input matrix over $\mathbb{F}_2$ that contains $m$ uniformly distributed $n$-bit input words as the rows of the matrix. Set $\beta_0 = 0$ and let

$$\beta_k = \bigoplus_{j=0}^{k-1} (u_j \cdot x_j \oplus v_j f_j(x_j; \alpha_j)),$$

for all $k = 1, \ldots, n$. Let $P(b, j)$ be a column vector of dimension $2^\omega$ and $M(b, j)$ be a $2^\omega \times 2^\omega$ matrix such that

$$\begin{aligned} P(b, j)_c &= \Pr[\beta_j = 0,\ \alpha_j = c] - \Pr[\beta_j \neq 0,\ \alpha_j = c] \quad \text{and} \\ M(b, j)_{d,c} &= \Pr[u_j \cdot x_j \oplus v_j f_j(x_j; \alpha_j) = 0,\ \alpha_{j+1} = d \mid \alpha_j = c] \\ &\quad - \Pr[u_j \cdot x_j \oplus v_j f_j(x_j; \alpha_j) \neq 0,\ \alpha_{j+1} = d \mid \alpha_j = c] \end{aligned}$$

for $j = 0, \ldots, n-1$ with $c, d \in \mathbb{F}_2^\omega$. We then have

$$\sum_{c=0}^{2^\omega - 1} M(b, j)_{d,c} P(b, j)_c = P(b, j+1)_d,$$

and thus

$$P(b, j+1) = M(b, j)P(b, j).$$

Note that

$$P(b, 0)_c = \Pr[\beta_0 = 0,\ \alpha_0 = c] - \Pr[\beta_0 \neq 0,\ \alpha_0 = c] = \begin{cases} 1, & c = 0, \\ 0, & c \neq 0. \end{cases}$$

At the other end we have

$$\begin{aligned} LP(b,n) &= \sum_{c=0}^{2^\omega-1} (\Pr[\beta_n = 0,\ \alpha_n = c] - \Pr[\beta_n \neq 0,\ \alpha_n = c]) \\ &= \Pr[\beta_n = 0] - \Pr[\beta_n \neq 0] \\ &= c_f(v,u) \end{aligned}$$

as desired. Since $A_{b_j} = M(b,j)$ and $C = P(b,0)$, it follows that

$$c_f(v,u) = LA_{b_{n-1}} \cdots A_{b_1} A_{b_0} C \ .$$

□

The correlation of a linear approximation of the T-function $f$ with narrowness $w$ can be thus computed by doing $n$ multiplications of a $2^\omega \times 2^\omega$ matrix and a column vector, and $2^\omega$ additional additions. For a fixed $\omega$, this is a linear-time algorithm, and for a small $\omega$ efficient in practice. The number of $2^\omega \times 2^\omega$ matrices to be stored in memory is $k2^{m+1}$ at most. Hence, the precomputation time grows exponentially with respect to $m$ and linearly with respect to the number of values of the parameter. It is quite obvious that this technique extends to cases, where $\alpha_j$ is not the same function for all $j > 0$. We can also apply it for any $\omega$-narrow T-function $f$, even if the narrowness of $f$ is unknown. This might be helpful in practice, since it is not necessary to find the exact narrowness of $f$. Linear representation leads also to an efficient method for generating all relevant masks for addition modulo $2^n$ with two inputs [Nyberg and Wallén, 2006, Wallén, 2003]. There does not seem to be any simple way for generating masks for more complex T-functions, however.

By allowing only certain linear approximations $(v,u)$ to be used for $f$, the correlation $c_f(v,u)$ can be determined by using a subset of the matrices in the linear representation. In some cases, one can derive an explicit formula for the correlation. For example, consider a linear approximation of $f$, where $f$ does not contain any constants or the constants are fixed. Let $v \in \mathbb{F}_2^n$ be the output mask and $u^{(0)}, \ldots, u^{(m-1)} \in \mathbb{F}_2^n$ be the input masks such that $v = u^{(0)} = \cdots = u^{(m-1)}$ and $w_{\mathrm{H}}(v) > 0$. Set $e = w_{\mathrm{H}}(v)$ and $i_{-1} = -1$, and denote the indices of the set bits of $v$ by $i_0, \ldots, i_{e-1}$ such that $i_0 < \cdots < i_{e-1}$. Denote $\delta_j = i_j - i_{j-1} - 1$, for $j = 0, \ldots, e-1$, so that $\delta_j$ represents the number of zero bits between the set bits $i_j$ and $i_{j-1}$ in $v$, when $0 < j \leq e-1$, and the number of trailing zero bits, when $j = 0$. Let $b = b_{n-1} \ldots b_1 b_0$ be the word associated with the approximation. This word consists of only two alphabets, one for the characters $b_i$, $i \in \{i_0, \ldots, i_{e-1}\}$, and one for the characters $b_i$, $i \in \{0, \ldots, n-1\} \setminus \{i_0, \ldots, i_{e-1}\}$. Hence, using the linear representation of the correlation $c_f(v,u)$ we can write

$$c_f(v,u) = LA_1 A_0^{\delta_{e-1}} \cdots A_1 A_0^{\delta_1} A_1 A_0^{\delta_0} C, \tag{3.12}$$

where $A_1$ is the transition matrix used for the characters $b_{i_0}, \ldots, b_{i_{e-1}}$, and $A_0$ is the transition matrix used for all other characters. The eigen decomposition of $A_0$ allows simplifying the terms $A_0^{\delta_j}$, $j = 0, \ldots, e-1$, so that one can derive a matrix-free expression for the correlation $c_f(v, u)$. This expression is obviously a function of $i_0, \ldots, i_{e-1}$. The leading zero bits in the masks do not affect the correlation, since $f$ is a T-function. This is why the term $A_0^{n-i_{e-1}-1}$ has been omitted from (3.12).

# Chapter 4

# Cryptanalysis of SOBER-128

SOBER-128 is a synchronous stream cipher designed by Hawkes, Paddon, and Rose [2003] of Qualcomm Australia. It is an improved version of SOBER-t32 [Rose and Hawkes, 1999] that was submitted to the NESSIE program. SOBER-128 generates a keystream of 32-bit words based on a 128-bit secret key. Originally, it also contained message authentication functionality, but that has been removed due to vulnerabilities to forgery attacks. According to the homepage of Qualcomm Australia [2006], the first attack by Watanabe and Furuya [2004] turned out to be easy to address; however, the designers themselves later found out a similar forgery attack that still applied to SOBER-128. In our analysis of SOBER-128, we concentrate only on the keystream generator.

The best known attack on the keystream generator part of SOBER-128 is due to Cho and Pieprzyk [2006a]. It uses an application of linear cryptanalysis for LFSR-based nonlinear filter generators as discussed in Chapter 3. First, linear approximate relations over nonlinear functions are derived which involve terms from the LFSR state variables and the keystream. Then a linear time-invariant relation originating from the LFSR recurrence relation is used to cancel out the internal LFSR state variables to obtain an approximate linear relation involving keystream variables only. The linear time-invariant relation involving six LFSR state variables used by Cho and Pieprzyk [2006a] is due to Ekdahl and Johansson [2002]. The resulting linear distinguishing attack requires $2^{103.6}$ terms of the keystream.

Our attack on SOBER-128 is also a linear distinguishing attack. To construct the distinguisher, we use the linear recurrence relation of the LFSR directly to cancel out the LFSR state variables. One part of the NLF of SOBER-128 is a pure T-function involving a secret key-dependent constant. We derive approximate linear relations over the NLF and show how the resulting approximate linear relation of the keystream variables can be used, not only to distinguish the output keystream from a purely random sequence but also to determine one bit of information of the secret constant. However,

it seems that the complexity increases slightly. To our current estimates it takes on the average $2^{113.5}$ terms of the keystream to get one bit of information of the secret constant, and $2^{124.6}$ terms to get four bits of the secret constant. This cryptanalytic technique is not specific to SOBER-128. It can be applied whenever linear approximations are taken over cryptographic functions involving secret constants. One linear approximation divides the constants into two classes depending on whether the bias of the keystream relation is positive or negative. In general, this information of the constants cannot be given in a form of a linear equation of the secret bits as is typically the case in linear cryptanalysis, e.g., in the seminal work of Matsui [1994].

The structure of the chapter is as follows. In Section 4.1, we give a description of the keystream generator part of SOBER-128. For a more detailed specification of SOBER-128, refer to Hawkes et al. [2003]. In Section 4.2, we describe our attack on SOBER-128.

## 4.1 Description of SOBER-128

The structure of the SOBER-128 keystream generator is a combination of an LFSR and an NLF. An illustration of this structure is depicted in Figure 4.1. The LFSR consists of 17 memory cells, each containing an element from $\mathbb{F}_2^{32}$. We use the vector $S_t = (s_t, \ldots, s_{t+16})$ to define the state of the LFSR at time $t \geq 0$. The new state at time $t+1$ is determined with the characteristic polynomial

$$x^{17} + x^{15} + x^4 + a \in \mathbb{F}_{2^{32}}[x], \tag{4.1}$$

where $a \in \mathbb{F}_{2^{32}}$ is a constant. We use polynomials in $\mathbb{F}_{2^8}[y]$ to represent the elements in the field $\mathbb{F}_{2^{32}}$. Likewise, the elements in $\mathbb{F}_{2^8}$ are treated as polynomials over the binary field $\mathbb{F}_2$. If we encode the coefficients of a polynomial over $\mathbb{F}_2$ as a hexadecimal number, the constant $a$ can be represented as $\texttt{0x01}y$ in $\mathbb{F}_{2^8}[y]$. The NLF, denoted by $F_K$, is a function of the LFSR state $S_t$ and a key-dependent constant $K \in \mathbb{F}_2^{32}$. At time $t \geq 0$, the NLF produces a 32-bit keystream word $z_t$ as

$$\begin{aligned} z_t &= F_K(s_t, s_{t+1}, s_{t+6}, s_{t+13}, s_{t+16}) \\ &= g((((g(s_t \boxplus s_{t+16}) \ggg 8) \boxplus s_{t+1}) \oplus K) \boxplus s_{t+6}) \boxplus s_{t+13}. \end{aligned}$$

The function $g\colon \mathbb{F}_2^{32} \to \mathbb{F}_2^{32}$ is defined as

$$g(x) = g_1(x_{31,\ldots,24}) \,\|\, (g_2(x_{31,\ldots,24}) \oplus x_{23,\ldots,0}), \tag{4.2}$$

where $\|$ denotes concatenation of two vectors, $g_1\colon \mathbb{F}_2^8 \to \mathbb{F}_2^8$ is the Skipjack S-box [FIPS PUB 185], and $g_2\colon \mathbb{F}_2^8 \to \mathbb{F}_2^8$ is a tailor-designed ISRC S-box [Dawson et al., 1999]. To simplify our analysis, we define a function

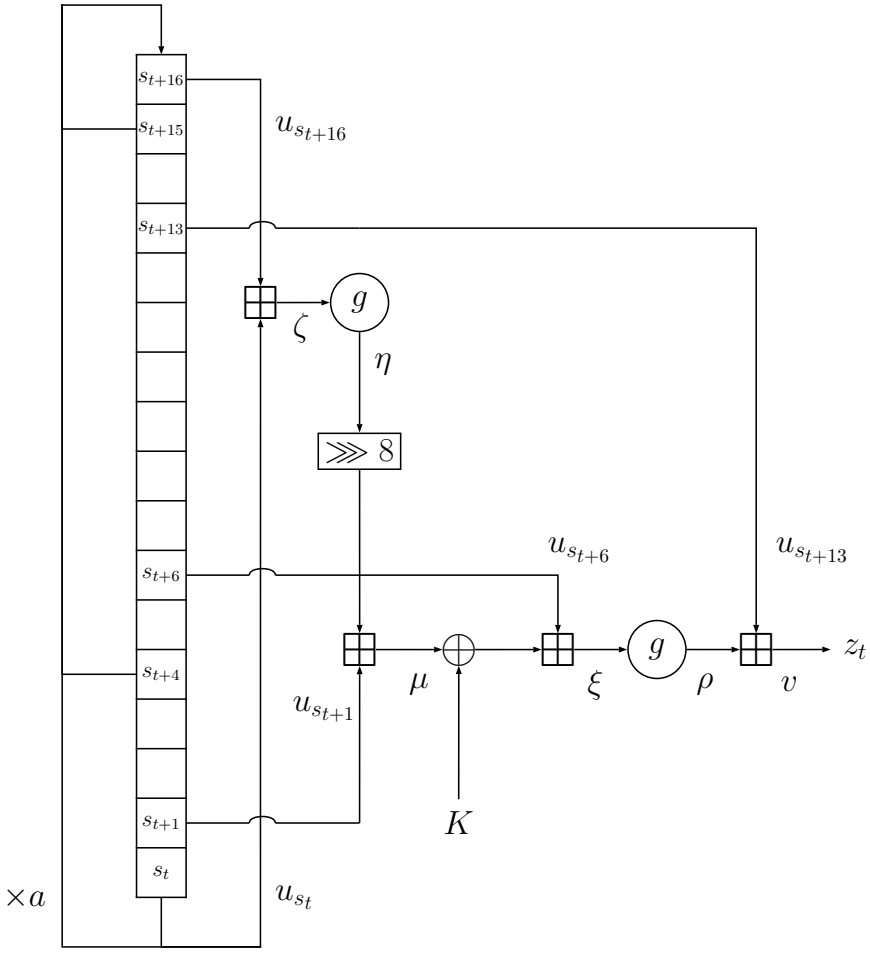


Figure 4.1: The SOBER-128 keystream generator.

$f_K\colon \mathbb{F}_2^{3\times n} \to \mathbb{F}_2^n$ as $f_K(x) = ((x^{(0)} \boxplus x^{(1)}) \oplus K) \boxplus x^{(2)}$. Thus, if we set $n = 32$, the output $z_t$ of the NLF can be written as

$$z_t = g(f_K((g(s_t \boxplus s_{t+16}) \ggg 8), s_{t+1}, s_{t+6})) \boxplus s_{t+13}, \quad t \geq 0.$$

## 4.2 Linear Masking of SOBER-128

We form a linear distinguisher for the SOBER-128 keystream generator using heuristic search methods. Our purpose is to search multiple approximations that partition the constant $K \in \mathbb{F}_2^{32}$ into different classes based on the correlation. This would allow us to gain information from $K$ by using linearly independent linear approximate relations as described in Section 3.4.4. We are especially interested in how the constants are partitioned based on the sign of the correlation, since then it is possible to get larger correlation differences between constant classes.

Let $v$ and $u$ denote the output and input masks of a linear approximation of $F_K$ respectively. The characteristic polynomial (4.1) for the LFSR yields the linear recurrence relation

$$s_{t+17} \oplus s_{t+15} \oplus s_{t+4} \oplus as_t = 0,$$

which can be used for forming the main distinguishing equation that consists of output words from the keystream generator. To form this distinguishing equation, we proceed as in Chapter 3. A linear approximation of $F_K$ is used four times: three times with the masks $v, u$ at times $t+4$, $t+15$, $t+17$ and one time with the masks $v_a, ua$ at time $t$. This procedure results in the following distinguishing equation:

$$vz_{t+17} \oplus vz_{t+15} \oplus vz_{t+4} \oplus v_a z_t = 0. \tag{4.3}$$

Let $\epsilon_{F_K}(v, u)$ denote the bias of a linear approximation $(v, u)$ of $F_K$ with a fixed $K \in \mathbb{F}_2^{32}$. Using the Piling-Up Lemma (3.3), the total bias $\epsilon_K(v, v_a)$ of the distinguishing equation (4.3) can be estimated to be

$$\epsilon_K(v, v_a) = 8\epsilon_{F_K}(v, u)^3 \epsilon_{F_K}(v_a, ua).$$

We determine the linear approximations $(v, u)$ and $(v_a, ua)$ by searching two linear chains of approximations over $F_K$, one to determine the mask pair $(v, u)$ and one to determine the mask pair $(v_a, ua)$. The linear chains are searched with the constant $K$ set to zero. A detailed description of the searching process is given in Section 4.2.2. The effect of $K$ to the correlation is examined after the linear chains have been formed. We do this by calculating the correlation of a linear approximation of $f_K$ with different $K$. The masks of this approximation are chosen to be the same as in the linear chain. Since $f_K$ is a T-function, one can determine the correlation efficiently by using the linear representation technique from Section 3.5.2. By enumerating

all $K \in \mathbb{F}_2^{32}$, it is easy to determine how the constants are partitioned into classes based on the correlation. We give the linear representation for linear approximations of $f_K$ in Section 4.2.4. Our results are given in the next section. In Section 4.2.3, we discuss how the constant $a$ in the characteristic polynomial affects linear distinguishing attacks on SOBER-128.

### 4.2.1 Results

We determined four independent linear distinguishers for the keystream generator. These distinguishers partition the constants $K \in \mathbb{F}_2^{32}$ into two classes of the same size based on the sign of the correlation. Hence, each distinguisher allows extracting one bit of information from $K$. The constant classes are defined by linear relations of form $w \cdot K = 0$, $w \in \mathbb{F}_2^{32}$, such that $\epsilon_K(v, v_a) > 0$ if and only if $w \cdot K = 0$. These constant classes are pairwise orthogonal, which means that we get 16 constant classes of the same size by combining these classes together. Our distinguishers are presented in Table 4.1 with their maximum, minimum, and average biases taken over $K$, and the constant classes defined by the vector $w$.

Table 4.1: Linear distinguishers for SOBER-128.

| $v$ | $v_a$ | $\lvert\epsilon_K(v, v_a)\rvert$ | | | $w$ |
|---|---|---|---|---|---|
| | | max | avg | min | |
| 0x01980000 | 0x00011000 | $2^{-53.288}$ | $2^{-56.735}$ | $2^{-62.001}$ | 0x01991000 |
| 0x00000181 | 0x24000001 | $2^{-55.385}$ | $2^{-58.290}$ | $2^{-62.385}$ | 0x0c40600c |
| 0x0040000c | 0x08006000 | $2^{-57.701}$ | $2^{-61.155}$ | $2^{-66.112}$ | 0x41000180 |
| 0x000000c0 | 0x21000000 | $2^{-58.959}$ | $2^{-62.279}$ | $2^{-66.638}$ | 0xa04000c0 |

Using the best linear approximation given in Table 4.1, one bit of information from $K$ can be obtained from $2^{113.5}$ keystream words on average. For obtaining two bits of information, also the second best mask is used. Therefore, we need $2^{116.6}$ keystream words on average for gaining two bits of information. We need not to pay attention only to the distinguisher with the best bias as with pure distinguishing attacks—multiple distinguishers with good biases give the possibility of gaining more information of the constant. The idea is comparable to Algorithm 1 in [Matsui, 1994], which applies for DES and other block ciphers. In this case, however, the constant is placed within a T-function and we do not always get a linear relation for different constant bits; indeed, some relations can be nonlinear as is shown in Section 4.2.4.

### 4.2.2 Searching the Masks

We search useful linear masks by using the techniques presented in Chapter 3 and also by using the algorithm by Wallén [2003]. We specifically take advantage of the possibility to generate all linear masks with a given correlation for one addition modulo $2^{32}$. In this section, the term correlation is used to refer to the absolute value of correlation unless otherwise specified. The linear distinguisher (4.3) makes use of the mask pairs $(v, u)$ and $(v_a, va)$. Two linear chains of approximations over the NLF are created to determine these mask pairs. The linear chains are created concurrently for one nonlinear component at a time. We progress to the next component, when we have found an approximation with correlation that is higher than the preset limits. During this process, we keep track of the total correlation using the Piling-Up Lemma. Since we assume here that different components of the cipher are statistically independent, the results are only rough estimates. More analysis is needed to assess their accuracy.

Used masks are depicted in Figure 4.1. The subscript $a$ is used to denote masks that work when the LFSR variable $s_i$ has been multiplied with $a$. We start by generating masks for the addition with $s_{t+1}$ as an input. All masks $u_{s_{t+1}}, \eta$, and $\mu$ are generated with a correlation $\geq 2^{-3}$. For each $u_{s_1}a$, we generate $\eta_a$ and $\mu_a$ with a correlation $\geq 2^{-4}$. The three least significant bytes of $\eta$ and $\eta_a$ are also the three least significant bytes of $\zeta$ and $\zeta_a$. Previous experiences show that large correlations are achieved with masks that have a low Hamming weight [Wallén, 2003, Watanabe et al., 2004]. Hence, we iterate all values with a Hamming weight $\leq 4$, for the most significant byte of $\zeta$, and generate $u_{s_t}$ and $u_{s_{t+16}}$ with a correlation $\geq 2^{-3}$. We continue with masks that have a nonzero correlation over $g$. For the input masks $u_{s_t}a$ and $u_{s_{t+16}}a$, we iterate all values with a Hamming weight $\leq 4$, for the most significant byte of $\zeta_a$, and compute the correlation for the addition and $g$. We continue with masks that have a correlation $\geq 2^{-6}$ over the addition and a nonzero correlation over $g$. We continue from the addition with $s_{t+6}$ as an input. Using $\mu$ we generate all masks $u_{s_{t+6}}$ and $\xi$ with a correlation $\geq 2^{-3}$. For each $u_{s_{t+6}}a$ and $\mu_a$, we generate $\xi_a$ with a correlation $\geq 2^{-4}$. These approximations fix the three least significant bytes of $\rho$ and $\rho_a$. We iterate all values for the most significant byte with a Hamming weight $\leq 4$ and generate $u_{s_{t+13}}$ and $v$ with a correlation $\geq 2^{-3}$. For the $u_{s_{t+13}}a$, we generate $v_a$ with a correlation $\geq 2^{-4}$ by iterating again all values with a Hamming weight $\leq 4$, for the most significant byte of $\rho_a$. A linear chain of approximations over the NLF has now been created.

### 4.2.3 Effect of $a$ in the Characteristic Polynomial

Without $a$ in the characteristic polynomial (4.1), the distinguishing equation (4.3) is formed using the same linear approximation $(v, u)$ four times.

Hence, we get the same equation as (4.3) but with $v_a$ replaced with $v$. The bias is determined as

$$\epsilon_K(v) = 8\epsilon_{F_K}(v,u)^4.$$

In this case, the sign of $\epsilon_{F_K}(v,u)$ would cancel out, which makes it harder to find constant classes that have large correlation differences. We considered this case for the purpose of showing how $a$ affects the security of the keystream generator. The distinguisher $v = \texttt{0x03000001}$ with (average) bias $\epsilon_{F_K}(v) = 2^{-36.771}$ was the best that we found. The results show that we get a distinguisher with much higher bias than with $a$ in the characteristic polynomial. It is harder, however, to gain information from $K$, since the distinguishing equation has only nonnegative correlations and thus, correlation differences between constant classes cannot be as large.

### 4.2.4 Linear Approximations of $f_K$

Recall that the function $f_K\colon \mathbb{F}_2^{3\times n} \to \mathbb{F}_2^n$ is defined as $f_K(x) = ((x^{(0)} \boxplus x^{(1)}) \oplus K) \boxplus x^{(2)}$, where $K = (k_0, \ldots, k_{n-1}) \in \mathbb{F}_2^n$ is a constant. This is obviously a T-function, for which the parameter $\alpha_j$ is the same function for $j = 1, \ldots, n-1$. In this section, we present the linear representation of the correlation $c_{f_K}(v,u)$. We also show how the correlation can vary depending on the value of $K$.

#### The Linear Representation for $f_K$

It is easy to see that the function $f_K$ is 2-narrow. It has three input words and two different components $f_{K,j}$, since $k_i \in \{0,1\}$, for $i = 0, \ldots, n-1$. We set $\alpha_0 = 0$ and use the parameter $\alpha_j = (\phi_j, \psi_j)$ with

$$\begin{cases} \phi_j = \lfloor (x_{0,j-1} + x_{1,j-1} + \phi_{j-1})/2 \rfloor, \\ \psi_j = \lfloor ((x_{0,j-1} \oplus x_{1,j-1} \oplus k_{j-1} \oplus \phi_{j-1}) + x_{2,j-1} + \psi_{j-1})/2 \rfloor, \end{cases}$$

for $j = 0, \ldots, n-1$. Hence, the linear representation of $c_{f_K}(v,u)$ has dimension $2^2 = 4$ and consists of vectors $L = (1,1,1,1)$ and $C = (1,0,0,0)^T$, and $2 \cdot 2^{3+1} = 32$ correlation matrices $A_r$. For clarity, we denote by $A^0_{\widetilde{r}}$ and $A^1_{\widetilde{r}}$ the matrices that correspond to the cases $k_j = 0$ and $k_j = 1$, respectively. The subscript $\widetilde{r}$ is only used to denote the linear approximation part of $r$, i.e., $\widetilde{r} = r \bmod 2^4$, for any $r \in \{0, \ldots, 31\}$. The matrices $A_r$ are given in Table 4.2.

#### Examples of Constant Classes

We give two small examples of how the constants $K \in \mathbb{F}_2^n$ are partitioned into classes with a fixed linear approximation $(v,u)$ of $f_K$. In our examples, $n = 5$, and the constants are of form $K = (k_0, k_1, k_2, k_3, k_4) \in \mathbb{F}_2^5$. The

Table 4.2: The matrices for the linear representation of $c_{f_K}(v,u)$.

$$A_0^0 = \frac{1}{8}\begin{pmatrix} 4 & 1 & 1 & 0 \\ 2 & 5 & 1 & 2 \\ 2 & 1 & 5 & 2 \\ 0 & 1 & 1 & 4 \end{pmatrix} \qquad A_{1,2}^0 = \frac{1}{8}\begin{pmatrix} 2 & 1 & 1 & 0 \\ -2 & -1 & -1 & 0 \\ 0 & 1 & 1 & 2 \\ 0 & -1 & -1 & -2 \end{pmatrix}$$

$$A_3^0 = \frac{1}{8}\begin{pmatrix} 0 & 1 & 1 & 0 \\ 2 & -3 & 1 & -2 \\ -2 & 1 & -3 & 2 \\ 0 & 1 & 1 & 0 \end{pmatrix} \qquad A_4^0 = -A_8^0 = \frac{1}{8}\begin{pmatrix} 2 & 1 & 1 & 0 \\ 0 & 1 & 1 & 2 \\ -2 & -1 & -1 & 0 \\ 0 & -1 & -1 & -2 \end{pmatrix}$$

$$A_{5,6}^0 = -A_{9,10}^0 = \frac{1}{8}\begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & -1 & -1 & 0 \\ 0 & -1 & -1 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix} \qquad A_7^0 = -A_{11}^0 = \frac{1}{8}\begin{pmatrix} -2 & 1 & 1 & 0 \\ 0 & 1 & 1 & -2 \\ 2 & -1 & -1 & 0 \\ 0 & -1 & -1 & 2 \end{pmatrix}$$

$$A_{12}^0 = \frac{1}{8}\begin{pmatrix} 0 & -1 & -1 & 0 \\ 2 & 3 & -1 & -2 \\ -2 & -1 & 3 & 2 \\ 0 & -1 & -1 & 0 \end{pmatrix} \qquad A_{13,14}^0 = \frac{1}{8}\begin{pmatrix} 2 & -1 & -1 & 0 \\ -2 & 1 & 1 & 0 \\ 0 & -1 & -1 & 2 \\ 0 & 1 & 1 & -2 \end{pmatrix}$$

$$A_{15}^0 = \frac{1}{8}\begin{pmatrix} 4 & -1 & -1 & 0 \\ 2 & -5 & -1 & 2 \\ 2 & -1 & -5 & 2 \\ 0 & -1 & -1 & 4 \end{pmatrix}$$

$$A_0^1 = \frac{1}{8}\begin{pmatrix} 5 & 2 & 2 & 1 \\ 1 & 4 & 0 & 1 \\ 1 & 0 & 4 & 1 \\ 1 & 2 & 2 & 5 \end{pmatrix} \qquad A_{1,2}^1 = \frac{1}{8}\begin{pmatrix} 1 & 2 & 0 & 1 \\ -1 & -2 & 0 & -1 \\ 1 & 0 & 2 & 1 \\ -1 & 0 & -2 & -1 \end{pmatrix}$$

$$A_3^1 = \frac{1}{8}\begin{pmatrix} -3 & 2 & -2 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & -2 & 2 & -3 \end{pmatrix} \qquad A_4^1 = -A_8^1 = \frac{1}{8}\begin{pmatrix} 1 & 0 & 2 & 1 \\ 1 & 2 & 0 & 1 \\ -1 & 0 & -2 & -1 \\ -1 & -2 & 0 & -1 \end{pmatrix}$$

$$A_{5,6}^1 = -A_{9,10}^1 = \frac{1}{8}\begin{pmatrix} 1 & 0 & 0 & 1 \\ -1 & 0 & 0 & -1 \\ -1 & 0 & 0 & -1 \\ 1 & 0 & 0 & 1 \end{pmatrix} \qquad A_7^1 = -A_{11}^1 = \frac{1}{8}\begin{pmatrix} 1 & 0 & -2 & 1 \\ 1 & -2 & 0 & 1 \\ -1 & 0 & 2 & -1 \\ -1 & 2 & 0 & -1 \end{pmatrix}$$

$$A_{12}^1 = \frac{1}{8}\begin{pmatrix} 3 & 2 & -2 & -1 \\ -1 & 0 & 0 & -1 \\ -1 & 0 & 0 & -1 \\ -1 & -2 & 2 & 3 \end{pmatrix} \qquad A_{13,14}^1 = \frac{1}{8}\begin{pmatrix} -1 & 2 & 0 & -1 \\ 1 & -2 & 0 & 1 \\ -1 & 0 & 2 & -1 \\ 1 & 0 & -2 & 1 \end{pmatrix}$$

$$A_{15}^1 = \frac{1}{8}\begin{pmatrix} -5 & 2 & 2 & -1 \\ -1 & 4 & 0 & -1 \\ -1 & 0 & 4 & -1 \\ -1 & 2 & 2 & -5 \end{pmatrix}$$

Table 4.3: The constant classes for the linear approximation with masks $v = 00011$, $u^{(0)} = 00011$, $u^{(1)} = 00011$, $u^{(2)} = 00011$ of $f_K$, when $n = 5$.

| $c_{f_K}$ | $-2^{-1}$ | 0 | $2^{-1}$ |
|---|---|---|---|
| $K$ | 00001 | 00000, 10000 | 00011 |
| | 00101 | 00010, 10010 | 00111 |
| | 01001 | 00100, 10100 | 01011 |
| | 01101 | 00110, 10110 | 01111 |
| | 10001 | 01000, 11000 | 10011 |
| | 10101 | 01010, 11010 | 10111 |
| | 11001 | 01100, 11100 | 11011 |
| | 11101 | 01110, 11110 | 11111 |

Table 4.4: The constant classes for the linear approximation with masks $v = 11101$, $u^{(0)} = 10111$, $u^{(1)} = 11110$, $u^{(2)} = 11101$ of $f_K$, when $n = 5$.

| $c_{f_K}$ | $-2^{-4}$ | 0 | $2^{-4}$ |
|---|---|---|---|
| $K$ | 00010 | 00000, 10000 | 00011 |
| | 00111 | 00001, 10001 | 00110 |
| | 01000 | 00100, 10100 | 01001 |
| | 01101 | 00101, 10101 | 01100 |
| | 10011 | 01010, 11010 | 10010 |
| | 10110 | 01011, 11011 | 10111 |
| | 11001 | 01110, 11110 | 11000 |
| | 11100 | 01111, 11111 | 11101 |

partitions are presented in Tables 4.3 and 4.4, where the constants under certain correlation belong to the same class. For clarity, we denote the vectors in $\mathbb{F}_2^5$ as binary numbers.

The constants are divided into classes according to the following relations. In Table 4.3, the constants belong to a class with a zero or nonzero correlation depending on whether $k_0 = 0$ or 1. Furthermore, depending on whether $k_1 = 0$ or 1, the constants belong to the class with a negative or positive correlation. This linear approximation also serves as an example of Theorem 1: the three most significant bits—$k_2$, $k_3$, and $k_4$—do not affect the correlation. In Table 4.4, the constants belong to the class with a zero correlation if the nonlinear relation $k_1 \oplus k_3 \oplus k_1k_2 \oplus k_0k_1k_2 \oplus k_1k_2k_3 \oplus k_0k_1k_2k_3 = 0$ holds. The rest of the constants belong to the class with a negative or positive correlation depending whether $k_0 \oplus k_1 \oplus k_2 \oplus k_3 \oplus k_4 = 1$ or 0. Hence, the classes are not always determined by linear relations, when the constant is within a T-function.

# Chapter 5

# Cryptanalysis of Shannon

Shannon is a synchronous stream cipher designed by Hawkes, McDonald, Paddon, Rose, and de Vries [2007] of Qualcomm Australia. It has been designed according to PROFILE 1 of the ECRYPT call for stream cipher primitives [2005]—but well after the call. Although Shannon is refered to as a software-oriented stream cipher in the specification [Hawkes et al., 2007], the authors add that Shannon should be at home in simple hardware implementations as well. In addition to keystream generation, Shannon also offers message authentication functionality that is directly incorporated into its structure. In our analysis of Shannon, we consider only the keystream generator part, however.

Our attack on Shannon is a linear distinguishing attack, where the distinguisher uses multiple linear approximations. Since Shannon is a NLFSR-based nonlinear filter generator, we consctruct the distinguisher as described in Chapter 3. A linear chain of approximations is created for both the NLF and the nonlinear recurrence relation of the NLFSR. The approximation of the recurrence relation is used to cancel out the NLFSR state variables in order to obtain an approximate linear relation involving keystream variables only. As with SOBER-128, there is a secret key-dependent constant in Shannon. Our attack is a pure distinguishing attack in the sense that it does not gain any information from the constant. It is able to distinguish the keystream of Shannon from about $2^{106.996}$ keystream words.

The structure of the chapter is as follows. A short description of Shannon is given in Section 5.1. In Section 5.2, we show how the linear distinguisher is constructed and estimate its efficiency.

## 5.1 Description of Shannon

The keystream generator of Shannon produces a keystream of 32-bit words based on a 256-bit secret key. It is based on a single NLFSR and an NLF. The NLFSR of Shannon consists of 17 memory cells, each containing an

element from $\mathbb{F}_2^{32}$. We use $(R_t, s_{t+1}, \ldots, s_{t+15})$ to denote the state at time $t \geq 0$. The state of Shannon is updated according to the following relations:

$$\begin{cases} s_{t+16} = f_1(s_{t+12} \oplus s_{t+13} \oplus K) \oplus (R_t \lll 1), \\ R_{t+1} = s_{t+1} \oplus f_2(s_{t+3} \oplus s_{t+16}), \end{cases} \tag{5.1}$$

where $f_1, f_2 \colon \mathbb{F}_2^{32} \to \mathbb{F}_2^{32}$ are nonlinear Boolean functions, and $K \in \mathbb{F}_2^{32}$ is a 32-bit secret constant that is derived in the initialization process. The output $z_t$ at time $t \geq 0$ is given as

$$z_t = s_{t+9} \oplus s_{t+13} \oplus f_2(s_{t+3} \oplus s_{t+16}), \tag{5.2}$$

for all $t \geq 0$. The functions $f_1$ and $f_2$ are defined as

$$\begin{cases} f_1(x) = g(g(x, 5, 7), 19, 22), \\ f_2(x) = g(g(x, 7, 22)), 5, 19), \end{cases} \tag{5.3}$$

with the function $g$ defined as

$$g(x, a, b) = x \oplus ((x \lll a) \vee (x \lll b)). \tag{5.4}$$

It follows straight from the definition of $f_1$ and $f_2$ that $f_i(x) \lll a = f_i(x \lll a)$, for $i = 1, 2$. The functions $f_1$ and $f_2$ are not surjective. According to the specification of Shannon [Hawkes et al., 2007] they cover about 84.74% and 84.34% of the codomain $\mathbb{F}_2^{32}$ respectively. For further details of Shannon, such as the initialization procedure and message authentication functionality, we refer to the specification [Hawkes et al., 2007].

## 5.2 Linear Masking of Shannon

To build a linear distinguisher for Shannon we need to linearize both the nonlinear update procedure (5.1) and the NLF (5.2). We rewrite the update procedure (5.1) as follows:

$$s_{t+16} = f_1(s_{t+12} \oplus s_{t+13} \oplus K) \oplus ((s_t \oplus f_2(s_{t+2} \oplus s_{t+15})) \lll 1). \tag{5.5}$$

We combine the relations (5.5) and (5.2) to get a linear approximate relation, whose distribution reflects the distribution of the keystream $z_0, z_1, \ldots$. The objective is to find a relation, whose distribution differs from the uniform distribution as much as possible. To this end, we try to include as few nonlinear terms as possible into the relation. The best relation that we found was formed by first treating the arguments to $f_1$ and $f_2$ as uniformly distributed independent random variables $x_i$ and then adding $z_t \lll 1$ and $z_{t+16}$ together. Using the relation (5.5), it follows that

$$\begin{aligned} (z_t \lll 1) \oplus z_{t+16} &= (s_{t+9} \oplus s_{t+13} \oplus f_2(x_1)) \lll 1 \\ &\quad \oplus s_{t+25} \oplus s_{t+29} \oplus f_2(x_2) \\ &= (f_2(x_1) \lll 1) \oplus f_2(x_2) \oplus f_1(x_3) \oplus (f_2(x_4) \lll 1) \\ &\quad \oplus f_1(x_5) \oplus (f_2(x_6) \lll 1). \end{aligned}$$

Since $x_i$'s are uniformly distributed and independent, and $f_i(x) \lll a = f_i(x \lll a)$ for $i = 1, 2$, we get

$$(z_t \lll 1) \oplus z_{t+16} = f_2(x_1) \oplus f_2(x_2) \oplus f_1(x_3) \oplus f_2(x_4) \oplus f_1(x_5) \oplus f_2(x_6). \quad (5.6)$$

Hence, we get a linear distinguisher for Shannon by using $(z_t \lll 1) \oplus z_{t+16}$ as the transformation for the input sequence in the transformation phase. Note that this is a multidimensional linear transformation, which could be written using multiple one-dimensional transformations as in Section 3.4.3. The distribution of $(z_t \lll 1) \oplus z_{t+16}$ is time-invariant, and independent of the constant $K$ and the initialization procedure. Thus, information from $K$ is not gained by using this linear distinguisher. As was discussed in Section 3.4.3, the keystream requirement of the distinguisher, which uses multiple one-dimensional linear approximations, can be determined from the correlations of all nonzero linear combinations of the one-dimensional approximations. Hence, we need to determine the correlation of the approximation

$$v \cdot ((z_t \lll 1) \oplus z_{t+16}) = 0, \quad (5.7)$$

for all $v \in \mathbb{F}_2^{32}$, in order to determine the keystream requirement. The terms $v \cdot f_i(x_j)$ are independent binary random variables so the correlation of (5.7) can be determined from the correlations of the approximations $v \cdot f_1(x) = 0$ and $v \cdot f_2(x) = 0$ using the Piling-Up Lemma (3.3). Hence, we determine the correlations of the approximations $v \cdot f_1(x) = 0$ and $v \cdot f_2(x) = 0$ for all $v \in \mathbb{F}_2^{32}$. To facilitate the computations, we use techniques from Chapter 3. By Section 3.5.1, these correlations can be computed within reasonable time using the FWHT. Roughly $2^{32}$ steps are needed to determine the distributions of each $f_1(x)$ and $f_2(x)$. The FWHT requires $n2^n$ computations, and therefore about $2^{37}$ steps are needed to compute the correlations of $v \cdot f_1(x) = 0$ and $v \cdot f_2(x) = 0$, for all $v \in \mathbb{F}_2^{32}$. Let $c_{f_1}(v, 0)$ and $c_{f_2}(v, 0)$ denote the correlations of the approximations $v \cdot f_1(x) = 0$ and $v \cdot f_2(x) = 0$, respectively. The correlation $c(v)$ of the linear approximation (5.7) is given by

$$c(v) = c_{f_1}(v, 0)^2 c_{f_2}(v, 0)^4.$$

In order to estimate the advantage of using a multidimensional transformation over a one-dimensional transformation in the distinguisher, we search for the mask $v \in \mathbb{F}_2^{32}$, which gives the highest correlation $c(v)$.

### 5.2.1 Results

To distinguish the correlation $c(v)$ in the keystrem, one needs $\mathcal{O}(1/c(v)^2)$ keystream words $z_t$. To distinguish the distribution of $(z_t \lll 1) \oplus z_{t+16}$ from uniform distribution, $\mathcal{O}(1/\sum_{v \neq 0} c(v)^2)$ keystream words $z_t$ are needed. The largest value of $|c(v)|$ for the approximation $v \cdot ((z_t \lll 1) \oplus z_{t+16}) = 0$ is achieved with the mask $v =$ `0x0410a4a1` or with any of its rotated versions

$v \lll i$, $i = 1, \ldots, 31$. Linear distinguishing equations (5.7) with these masks have the correlation $c(v) = 2^{-56}$. To distinguish such correlation in $v \cdot ((z_t \lll 1) \oplus z_{t+16}) = 0$, we need approximately $2^{112}$ keystream words $z_t$. Approximately $2^{106.996}$ keystream words $z_t$ are needed to distinguish the full distribution of $((z_t \lll 1) \oplus z_{t+16})$. Hence, the distinguisher with only one linear approximation needs about $2^5$ times the keystream than the distinguisher with multiple linear approximations. If all nozero masks would induce the same correlation as the masks `0x0410a4a1` $\lll i$, $i = 0, \ldots, 31$, the keystream requirement would reduce by a factor of $2^{32}$. Since the reduction factor is $2^{5.004}$, all other masks have a negligible effect on the requirement: they reduce it by a factor of $2^{0.004}$.

Keystream generation in Shannon is limited to $2^{64}$ words for one key and to $2^{40}$ words for one initialization vector. Since the distribution of $((z_t \lll 1) \oplus z_{t+16})$ does not depend on either—the key or the $IV$—these limitations do not matter even though the keystream requirement for the distinguisher is much larger: in theory, one could generate enough keystream for the distinguisher by initializing the cipher with a new key or $IV$ when it is necessary.

# Chapter 6

# Conclusions

In this thesis, we studied linear cryptanalysis of stream ciphers and presented linear distinguishing attacks on SOBER-128 and Shannon. Our focus was on techniques for finding useful linear transformations, which are applied on the input sequence in linear distinguishers. We studied linear approximations of a class of vector-valued Boolean functions, called $\omega$-narrow T-functions, and presented a technique for determining the correlation of a linear approximation efficiently for a small $\omega$. A second topic discussed in this thesis was the analysis of secret constants in T-functions using linear approximations. We observed that the correlation of a linear approximation varies with the constant, and used this fact in a linear distinguishing attack on SOBER-128, which has a T-function with a secret, key-dependent constant in its output filter. It was shown that a number of bits from the secret constant can be recovered with a slight overhead in the complexity compared to a pure distinguishing attack. In the attack on Shannon, we took advantage of multidimensional approximation to reduce the attack complexity. Using 32 linearly independent approximations the data complexity could be reduced by a factor of $2^5$ compared to the attack with one linear approximation.

In his paper on narrow T-functions, Daum [2005] concluded that a subclass of T-functions with small narrowness appears to be weak for cryptographic purposes, since the efficiency of the proposed algorithm for solving equation systems given by T-functions depends significantly on the narrowness of the involved T-functions. The results in this thesis are similar: linear approximation of T-functions with small narrowness can be studied more efficiently. T-functions, whose narrowness grows with the length of the input, seem to be the most difficult T-functions to analyze with our current techniques. It would be desirable to find a technique for studying linear approximations of T-functions from this class also.

One possible direction for future research is linear approximation of functions with secret constants. It would be useful to develop more techniques

to analyze linear approximations of such functions. This has applications in linear cryptanalysis of stream ciphers and block ciphers, since secret constants are used in ciphers of both classes. A second possible topic for future research is further development of the attack on SOBER-128 from using multiple single linear approximations to full-fledged multidimensional attack.

# Bibliography


T. Baignères, P. Junod, and S. Vaudenay. How far can we go beyond linear cryptanalysis? In *Advances in Cryptology—Asiacrypt 2004*, volume 3329 of *Lecture Notes in Computer Science*, pages 432–450. Springer-Verlag, 2004.

E. R. Berlekamp. *Algebraic Coding Theory*. McGraw-Hill, 1968.

E. Biham and A. Shamir. Differential cryptanalysis of DES-like cryptosystems. In *Advances in Cryptology—Crypto 1990*, pages 2–21. Springer-Verlag, 1990.

A. Biryukov, C. D. Cannière, and M. Quisquater. On multiple linear approximations. In *Advances in Cryptology—Crypto 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 1–22. Springer-Verlag, 2004.

J. Y. Cho and J. Pieprzyk. Distinguishing attack on SOBER-128 with linear masking. In *Information Security and Privacy 2006*, volume 4058 of *Lecture Notes in Computer Science*, pages 29–39. Springer-Verlag, 2006a.

J. Y. Cho and J. Pieprzyk. Crossword puzzle attack on NLS. In *Selected Areas in Cryptography 2006*, volume 4356 of *Lecture Notes in Computer Science*, pages 249–265. Springer-Verlag, 2006b.

D. Coppersmith, S. Halevi, and C. Jutla. Cryptanalysis of stream ciphers with linear masking. In *Advances in Cryptology—Crypto 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 515–532. Springer-Verlag, 2002.

J. Daemen, R. Govaerts, and J. Vandewalle. Correlation matrices. In *Fast Software Encryption 1994*, volume 1008 of *Lecture Notes in Computer Science*, pages 275–285. Springer-Verlag, 1995.

M. Daum. Narrow T-functions. In *Fast Software Encryption 2005*, volume 3557 of *Lecture Notes in Computer Science*, pages 50–67. Springer-Verlag, 2005.

E. Dawson, W. Millan, L. Burnett, and G. Carter. On the design of 8*32 S-boxes. Unpublished report, Information Systems Research Centre (ISRC), Queensland University of Technology (QUT), 1999.

ECRYPT. Call for stream cipher primitives. `http://www.ecrypt.eu.org/stream/call/`, 2005.

P. Ekdahl and T. Johansson. Distinguishing attacks on SOBER-t16 and t32. In *Fast Software Encryption 2002*, volume 2365 of *Lecture Notes in Computer Science*, pages 210–224. Springer-Verlag, 2002.

S. R. Fluhrer, I. Mantin, and A. Shamir. Weaknesses in the key scheduling algorithm of RC4. In *Selected Areas in Cryptography 2001*, volume 2259 of *Lecture Notes in Computer Science*, pages 1–24. Springer-Verlag, 2001.

J. D. Golić. Correlation via linear sequential circuit approximation of combiners with memory. In *Advances in Cryptology—Eurocrypt 1992*, volume 658 of *Lecture Notes in Computer Science*, pages 113–123. Springer-Verlag, 1993.

J. D. Golić. Intrinsic statistical weakness of keystream generators. In *Advances in Cryptology—Asiacrypt 1994*, volume 917 of *Lecture Notes in Computer Science*, pages 91–103. Springer-Verlag, 1995.

P. Hawkes, M. Paddon, and G. G. Rose. Primitive specification for SOBER-128. Technical report, Qualcomm Australia, 2003.

P. Hawkes, C. McDonald, M. Paddon, G. G. Rose, and M. W. de Vries. Primitive specification for shannon. Technical report, Qualcomm Australia, 2007.

B. Kaliski and M. Robshaw. Linear cryptanalysis using multiple approximations. In *Advances in Cryptology—Crypto 1994*, volume 839 of *Lecture Notes in Computer Science*, pages 26–39. Springer-Verlag, 1994.

A. Klimov and A. Shamir. A new class of invertible mappings. In *Cryptographic Hardware and Embedded Systems 2002*, volume 2523 of *Lecture Notes in Computer Science*, pages 471–484. Springer-Verlag, 2003.

A. Klimov and A. Shamir. New cryptographic primitives based on multiword t-functions. In *Fast Software Encryption 2004*, volume 3017 of *Lecture Notes in Computer Science*, pages 1–15. Springer-Verlag, 2004.

A. Klimov and A. Shamir. New applications of T-functions in block ciphers and hash functions. In *Fast Software Encryption 2005*, volume 3557 of *Lecture Notes in Computer Science*, pages 18–31. Springer-Verlag, 2005.

L. R. Knudsen. Contemporary block ciphers. In *Lectures on Data Security*, volume 1561 of *Lecture Notes in Computer Science*, pages 105–126. Springer-Verlag, 1999.

H. Lipmaa. On differential properties of pseudo-hadamard transform and related mappings. In *Indocrypt 2002*, volume 2551 of *Lecture Notes in Computer Science*, pages 48–61. Springer-Verlag, 2002.

H. Lipmaa. On the additive differential probability of exclusive-or. In *Fast Software Encryption 2004*, volume 3017 of *Lecture Notes in Computer Science*, pages 317–331. Springer-Verlag, 2004.

H. Lipmaa and S. Moriai. Efficient algorithms for computing differential properties of addition. In *Fast Software Encryption 2001*, volume 2355 of *Lecture Notes in Computer Science*, pages 336–350. Springer-Verlag, 2001.

J. L. Massey. Shift-register synthesis and BCH decoding. *IEEE Transactions on Information Theory*, 15(1):122–127, 1969.

M. Matsui. Linear cryptanalysis method for DES cipher. In *Advances in Cryptology—Eurocrypt 1993*, volume 765 of *Lecture Notes in Computer Science*, pages 386–397. Springer-Verlag, 1994.

M. Matsui and A. Yamagishi. A new method for known plaintext attack of FEAL cipher. In *Advances in Cryptology—Eurocrypt 1992*, volume 658 of *Lecture Notes in Computer Science*, pages 81–91. Springer-Verlag, 1993.

A. Maximov and T. Johansson. Fast computation of large distributions and its cryptographic applications. In *Advances in Cryptology—Asiacrypt 2005*, volume 3788 of *Lecture Notes in Computer Science*, pages 313–332. Springer-Verlag, 2005.

A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of applied cryptography*. CRC Press, 1997.

National Bureau of Standards (NBS). Data encryption standard. Federal Information Processing Standards Publication (FIPS PUB) 46, 1977.

National Institute of Standards and Technology (NIST). Escrowed encryption standard. Federal Information Processing Standards Publication (FIPS PUB) 185, 1994.

National Institute of Standards and Technology (NIST). Advanced encryption standard (AES). Federal Information Processing Standards Publication (FIPS PUB) 197, 2001.

K. Nyberg. Correlation theorems in cryptanalysis. *Discrete Applied Mathematics*, 111:177–188, 2001.

K. Nyberg and J. Wallén. Improved linear distinguishers for SNOW 2.0. In *Fast Software Encryption 2006*, volume 4047 of *Lecture Notes in Computer Science*, pages 144–162. Springer-Verlag, 2006.

E. Pasalic. *On Boolean Functions in Symmetric-Key Ciphers*. PhD thesis, Lund University, 2003.

Qualcomm Australia. The homepage for SOBER-128. `http://www.qualcomm.com.au/Sober128.html`, 2006.

G. G. Rose and P. Hawkes. The t-class of SOBER stream ciphers. Technical report, Qualcomm Australia, 1999.

R. A. Rueppel. *Analysis and Design of Stream Ciphers*. Springer-Verlag, 1986.

C. Shannon. Communication theory of secrecy systems. *Bell System Technical Journal*, 28(4):656–715, 1949.

S. Vaudenay. An experiment on DES statistical cryptanalysis. In *3rd ACM Conference on Computer Security*, pages 139–147. ACM Press, 1996.

J. Wallén. Linear approximations of addition modulo $2^n$. In *Fast Software Encryption 2003*, volume 2887 of *Lecture Notes in Computer Science*, pages 261–273. Springer-Verlag, 2003.

D. Watanabe and S. Furuya. A MAC forgery attack on SOBER-128. In *Fast Software Encryption 2004*, volume 3017 of *Lecture Notes in Computer Science*, pages 472–482. Springer-Verlag, 2004.

D. Watanabe, A. Biryukov, and C. D. Cannière. A distinguishing attack of SNOW 2.0 with linear masking method. In *Selected Areas in Cryptography 2003*, volume 3006 of *Lecture Notes in Computer Science*, pages 222–233. Springer-Verlag, 2004.