Selected Cryptographic Methods for Securing Low-End Devices

Inauguraldissertation
zur Erlangung des akademischen Grades eines
Doktors der Naturwissenschaften

der Universität Mannheim

vorgelegt von

Dipl. Wirtsch.-Inf. Dirk Stegemann
aus Coesfeld (Westf.)

Mannheim, 2010

Dekan: Prof. Dr. Wolfgang Effelsberg, Universität Mannheim
Referent: Prof. Dr. Matthias Krause, Universität Mannheim
Korreferent: Prof. Dr. Stefan Lucks, Bauhaus-Universität Weimar

Tag der mündlichen Prüfung: 16. Dezember 2010

Abstract

We consider in this thesis the security goals confidentiality of messages and authenticity of entities in electronic communication with special focus on applications in environments with restricted computational power, e.g., RFID-tags or mobile phones. We introduce the concept of stream ciphers, describe and analyze their most important building blocks, analyze their security features, and indicate ways to improve their resistance against certain types of attacks. In the context of entity authentication, we describe special protocols based on randomly choosing elements from a secret set of linear vector spaces and relate the security of these protocols to the hardness of a certain learning problem.

Zusammenfassung

Wir betrachten in dieser Arbeit die Sicherheitsziele Vertraulichkeit von Nachrichten und Authentizität von Kommunikationspartnern im Umfeld elektronischer Kommunikation mit besonderem Schwerpunkt auf Anwendungen auf ressourcenbeschränkten Endgeräten wie RFID-Tags oder Mobiltelefonen. Wir betrachten insbesondere Stromchiffren, beschreiben und analysieren ihre wichtigsten Bestandteile, untersuchen ihre Sicherheitseigenschaften und zeigen Möglichkeiten auf, wie sich ihre Resistenz gegenüber bestimmten Angriffstechniken verbessern lässt. Im Zusammenhang mit der Authentifikation von Kommunikationspartnern beschreiben wir spezielle Authentifikationsprotokolle, die auf der zufälligen Auswahl von Elementen aus einer geheimen Menge von linearen Vektorräumen beruhen, und führen die Sicherheit dieser Protokolle auf die Komplexität eines bestimmten Lernproblems zurück.

Acknowledgment

After all the typing has been done, there is one page left to write – the page that ends up being one of the first in the thesis but might as well be the last, since it bears the opportunity to look back on the process of creating the document that is about to be finalized.

In this process, my supervisor Matthias Krause has been a most valuable source of support and inspiration. I have known few people with such a deep understanding of the theoretical foundations of cryptography and computer science and with such great reliability in telling apart promising ideas from dead end streets. Having been a teacher myself, it will always remain a secret to me how he manages to hold perfectly consistent 90-minute blackboard lectures right off the top of his head.

Together with the deputy head of the theoretical computer science and data security group, our secretary Karin Teynor, Matthias has additionally provided an efficient work environment with a productive combination of responsibilities and degrees of freedom that will be hard to duplicate.

My co-supervisor Stefan Lucks has always been my prime example of a natural-born cryptographer. He would talk about his work and share his ideas and experience with a sweeping passion that literally drags his students and colleagues into the world of cryptography. At the same time, he is one of the few living efficiently accessible cryptographic oracles (in the true sense) that have an answer to virtually any crypto-related question – either the solution right away or a "Oh yes, there was a paper at EUROCRYPT 2005 about that" type of pointer that reliably leads you in the right direction.

Matthias and Stefan have connected me to the international cryptographic community with many bright researchers and amazing talents. In the spirit of these people, who respect each other for the outcome of their work rather than for the titles they hold, and who obstinately organize the author lists of their papers in alphabetical order, I would like to express my gratitude for many fruitful and inspiring crypto-related as well as less-crypto-related discussions to a few colleagues, co-authors and community members that I had the pleasure to interact with over the years: Frederik Armknecht, Tobias Eibach, Simon Fischer, Andreas Kemper, Willi Meier, Emin Islam Tatlı, and Siavash Vahdati.

After all, there is more to the world than cryptography and IT-security, and I am grateful to have my family a handful of long-time friends taking great care in reminding me of the difference between *important* and *really important*. You all have contributed to the forthcoming pages more than you may know.

And finally, my life would be so much different without the love, understanding and unswerving support of my wife Olha, so this thesis is most naturally dedicated to her.

Contents

1 Introduction **1**
1.1 What this Thesis is about 1
1.2 Publications 2

I Confidential Communication with Stream Ciphers **5**

2 Algorithms for Confidential Communication **7**
2.1 Security Definitions and Attacker Models 7
2.2 Block Ciphers 9
2.3 Dedicated Stream Ciphers 10
2.4 Asymmetric Ciphers 13

3 Stream Cipher Building Blocks **15**
3.1 Boolean Functions 15
3.2 Feedback Shift Registers 16
3.2.1 Linear Feedback Shift Registers (LFSRs) 16
3.2.2 Feedback Shift Registers With Carry (FCSRs) 21

4 Stream Ciphers based on Feedback Shift Registers **33**
4.1 Generic Constructions 33
4.1.1 Combination Generators and Filter Generators 33
4.1.2 Additional Memory 34
4.1.3 Irregular Clocking 35
4.2 Example Ciphers 35
4.2.1 Self-Shrinking Generator 36
4.2.2 E_0 Generator 36
4.2.3 A5/1 Generator 38
4.2.4 Trivium 39
4.2.5 Grain-128 40
4.2.6 Filtered FCSRs 40
4.3 Abstraction: Internal Bitstream Generators 44

5 The BDD-Attack **47**
5.1 Introduction and Overview 47
5.2 Representing Boolean Functions with Binary Decision Diagrams 48
5.2.1 Free Binary Decision Diagrams (FBDDs) 48
5.2.2 Ordered Binary Decision Diagrams (OBDDs) 50

5.3 BDD-based Initial State Recovery . . . 52
5.4 Generic BDD Constructions . . . 54
5.4.1 Keystream Consistency Check Q_m . . . 54
5.4.2 FSR Consistency Check R_m . . . 56
5.5 Applications . . . 59
5.5.1 Self-Shrinking Generator . . . 59
5.5.2 Bluetooth Keystream Generator E_0 . . . 60
5.5.3 GSM Keystream Generator A5/1 . . . 61
5.5.4 TRIVIUM . . . 65
5.5.5 Grain-128 . . . 68
5.5.6 The F-FCSR Stream Cipher Family . . . 69
5.6 Divide-and-Conquer Strategies (DCS) . . . 70
5.6.1 DCS for regularly clocked (k, l)-Combiners . . . 71
5.6.2 DCS for the A5/1 Generator . . . 73
5.7 Simulations and Experimental Results . . . 76
5.8 Discussion of the BDD-Attack . . . 77

6 Other Generic Attacks on Stream Ciphers 79
6.1 Correlation Attacks . . . 79
6.1.1 The Basic Idea . . . 79
6.1.2 Analysis of the Special Case $C(x_t, q_t) = \alpha(x_t) \oplus \beta(q_t)$ . . . 82
6.2 Algebraic Attacks . . . 84
6.2.1 The Basic Idea . . . 84
6.2.2 Analysis of a restricted Scenario . . . 85
6.3 Countermeasures and Design Principles . . . 87
6.3.1 Increasing the Resistance against Correlation Attacks . . . 87
6.3.2 Increasing the Resistance against Algebraic Attacks . . . 89
6.4 Application to E_0 . . . 90

II Authenticity with Linear Protocols 95

7 Algorithms for Entity and Message Authentication 97
7.1 Security Definitions and Attacker Models . . . 97
7.1.1 Entity Authentication . . . 98
7.1.2 Entity Recognition . . . 98
7.1.3 Message Authentication . . . 99
7.1.4 Message Recognition . . . 99
7.1.5 Attacker Models . . . 100
7.2 Message Authentication Codes . . . 101
7.2.1 Message Authentication Codes based on Block Ciphers . . . 102
7.2.2 Message Authentication Codes based on Cryptographic Hash Functions . . . 102
7.3 Message Authentication with Digital Signatures . . . 103
7.4 Challenge-Response based Entity Authentication . . . 104
7.5 Authentication Schemes based on Hash Chains . . . 105
7.6 Authentication based on the Hardness of Learning Problems . . . 106

8 The HB Family of Authentication Protocols 107
8.1 The HB Protocol 107
8.2 The HB^{+} Protocol 108
8.3 Variants of the HB^{+} Protocol 109
8.3.1 The HB^{++} Protocol 110
8.3.2 The HB* Protocol 111
8.3.3 The HB-MP Protocols 112
8.3.4 The HB$^{\#}$ Protocol 113
8.3.5 The Trusted-HB Protocol 114

9 The (n, k, L) Family of Authentication Protocols 117
9.1 Introduction and Overview 117
9.2 The Linear (n, k, L) Protocol 117
9.3 The Linear $(n, k, L)^{+}$ Protocol 119
9.4 The Linear $(n, k, L)^{++}$ Protocol 119
9.5 Special Cases of Linear (n, k, L) Protocols 121
9.6 Security of Linear (n, k, L)-type Protocols and the LULS Problem 122
9.6.1 The Search-for-a-Basis Heuristic 122
9.6.2 The LULS Problem 123
9.6.3 On Solving the LULS Problem 124
9.7 Discussion 126

10 Conclusion 129

Bibliography 131

List of Tables 143

List of Figures 145

List of Algorithms 147

Index 147

Chapter 1

Introduction

1.1 What this Thesis is about

In this thesis, we consider two important security goals in electronic communication: confidentiality of messages and authenticity of entities.

While many algorithms and systems have been proposed for a variety of application scenarios, we focus our attention on methods that are particularly useful for devices with rather little computational power. This characterization is not entirely sharp – for example, it is arguable whether a modern smartphone is rather a particularly complex mobile phone or a small computer equipped with a telephone function – but will serve as a guideline to distinguish RFID tags and Bluetooth devices from personal computers and grids.

By confidentiality of messages, we mean that messages which are exchanged over a publicly observable channel should only be meaningful to legitimate communication partners. This is probably the most prominent service that cryptographic systems are expected to provide, commonly by encrypting (or enciphering) messages. In the first part of this thesis, we therefore introduce the concepts of block ciphers and stream ciphers and devote our main attention to hardware-oriented stream cipher constructions. We describe and analyze their most important building blocks, consider generic attack strategies – particularly the BDD-based attack correlation attacks and algebraic attacks – and indicate design principles that provide a certain resistance against these attacks. Furthermore, we review the design and security features of practically used algorithms such as the A5/1 algorithm used in the GSM standard and the E_0 cipher used in Bluetooth. In the case of E_0, we indicate possible design improvements in the light of the presented attacks.

In human face-to-face communication, authentication is implicitly and without further ado performed through face (and sometimes additionally voice) recognition. In electronic communication, e.g. on the internet, it is often not so easy to verify that a communication partner is in fact who she claims to be. Authentication of entities is therefore another important task of modern cryptographic systems. We address this security goal in the second part of this thesis by investigating lightweight authentication protocols that are based on randomly choosing elements from a set of L linear subspaces of $\mathrm{GF}(2)^{n+k}$, relate their security to the hardness of a certain learning problem and indicate

possible improvements and further research directions.

1.2 Publications

This thesis is based on the following publications.

Fully reviewed Publications

1. *Design Principles for Combiners with Memory*, Proceedings of the 6th International Conference on Cryptology in India (INDOCRYPT 2005), volume 3739 of LNCS, pages 104–117, Springer, 2005, with Frederik Armknecht and Matthias Krause

   Lower bounds on the complexities of algebraic attacks and correlation attacks, application to E_0 and proposal of a more secure E_0 variant

2. *Reducing The Space Complexity of BDD-based Attacks on Keystream Generators*, Proceedings of Fast Software Encrpytion, 13th International Workshop (FSE 2006), volume 4047 of LNCS, pages 163–178, Springer, 2006, with Matthias Krause

   Divide-and-conquer strategies for reducing the memory requirements of BDD-attacks, application to E_0, A5/1 and the Self Shrinking Generator

3. *Extended BDD-based Cryptanalysis of Keystream Generators*, Proceedings of the 14th International Workshop on Selected Areas in Cryptography (SAC 2007), volume 4876 of LNCS, pages 17–35, Springer, 2007

   Extension of the BDD-attack to NFSRs and arbitrary compression functions, application to Trivium, Grain and F-FCSR

4. *More on the Security of Linear RFID Authentication Protocols*, Proceedings of the 16th International Workshop on Selected Areas in Cryptography (SAC 2009), volume 5867 of LNCS, pages 182–196, Springer, 2009, with Matthias Krause

   Generalization of the CKK-protocol, security analysis in the active attack scenario, definition of linear (n, k, L) protocols and the LULS problem

5. *Some Remarks on FCSRs and Implications for FCSR-based Stream Ciphers*, Journal of Mathematical Cryptology, volume 3, pages 227–236, 2009, with Simon Fischer and Willi Meier

   Simplified description of the sequences produced by a single cell of a Galois FCSR given that the register's initial state is periodic, mappings between periodic states of the Fibonacci and the Galois representation of an FCSR, explicit determination of the autocorrelation of an l-sequence

Workshop Records and Technical Reports

1. *Equivalent Representations of the F-FCSR Keystream Generator*, Workshop Record of the State of the Art of Stream Ciphers (SASC 2008), with Simon Fischer and Willi Meier

2. *Building Stream Ciphers from FCSRs*, Workshop Record of the 2nd GI-Kryptowochenende, 2008

3. *Some Remarks on FCSRs and Implications for FCSR-based Stream Ciphers*, Workshop Record of the Second Workshop on Mathematical Cryptology (WMC '08), 2008, with Simon Fischer and Willi Meier

4. *Algebraic Attacks against Linear RFID Authentication Protocols*, Dagstuhl Seminar on Symmetric Cryptography, Workshop Record, 2009, with Matthias Krause

During my time as PhD student, I also contributed to a few other publications that are not mentioned in this thesis.

- Security Challenges of Location-aware Mobile Business, Proceedings of the 2nd IEEE International Workshop on Mobile Commerce and Services, pages 84–93, IEEE Computer Society, with Emin Islam Tatlı and Stefan Lucks

- Dynamic Anonymity, The 4th World Enformatika Conferences, International Conference on Information Security (ICIS '05), 2005, with Emin Islam Tatlı and Stefan Lucks

- Dynamic Mobile Anonymity with Mixing, Technical Report, University of Mannheim, 2006, with Emin Islam Tatlı and Stefan Lucks

- Workshop Record of the 2nd GI-Kryptowochenende (editor), 2006, with Frederik Armknecht

Part I

Confidential Communication with Stream Ciphers

Chapter 2

Algorithms for Confidential Communication

2.1 Security Definitions and Attacker Models

Often people want systems to be *secure* without having a precise idea of what they mean by security.

In a communication scenario in which two parties (usually called Alice and Bob), communicate over an insecure channel (e.g., a telephone land line or a TCP/IP connection), most people would agree that *security* means (possibly among other things) that an eavesdropper on the communication channel (wiretapping the telephone line or observing the messages when passing a router) should not be able to understand what Alice and Bob are talking about, or more formally, that the exchanged messages should remain confidential.

Many people have come across this problem already at some point in their childhood and most probably tried to solve it by enciphering the messages in one way or another, i.e., by transforming the plaintext messages into some meaningless-looking strings and having the receiver reverse the transformation in order to recover the plaintext.

This idea is probably almost as old as mankind, and the first documented ideas for enciphering methods date back to ancient times. Since then, cipher systems have been built and used for critical governmental and military applications with varying success, thereby influencing the course of history at quite a few points (Kahn, 1996).

Surprisingly enough, it was not until World War Two that a formal model and analysis of cipher systems was developed in a famous seminal paper by Shannon (1949), which is now considered one of the foundations of modern cryptography.

Since then, we typically assume a communication model consisting of a reliable but publicly observable communication channel, a sender who is equipped with an encryption algorithm E, a receiver running a decryption algorithm D, and a key source that provides encryption keys k_e and decryption keys k_d (see Fig. 2.1).

A cipher system is called *symmetric* if $k_e = k_d$ and *asymmetric* if $k_e \neq k_d$.

We usually relate the security of a communication system to an attacker

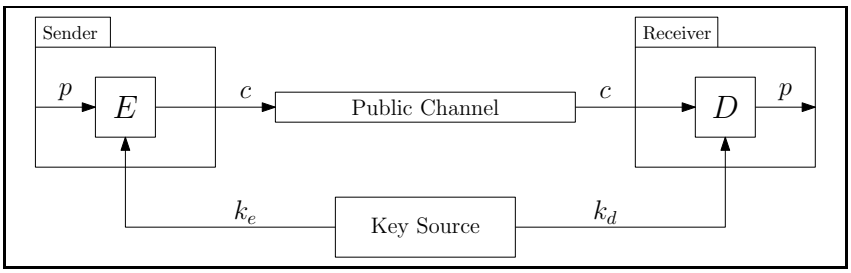

Figure 2.1: The Shannon communication model

that is defined by

- his goal
- his computational power
- the information available to him

and characterize the attacker by his success probability and his resource consumption in terms of time, memory and amount of utilized information. The most common goal is to obtain the secret key (leading to key recovery attacks), but also weaker goals such as deducing partial information about the exchanged information are often considered.

Depending on whether the attacker has unlimited or limited computational power (i.e., time and memory resources at his disposal), we talk about an *information theoretic* or a *complexity theoretic* security setting.

Concerning the amount of available information, we typically distinguish the following classes.

- *ciphertext-only*: The attacker has only access to the public channel.
- *known plaintext*: The attacker additionally knows a number of plaintexts and their encryptions under the unknown key.
- *chosen plaintext*: The attacker may have a number of plaintexts of his choice encrypted under the unknown key and obtain the corresponding ciphertexts.
- *chosen ciphertext*: The attacker may have a number of ciphertexts decrypted under the unknown key and obtain the corresponding plaintexts.

Note that we always assume the attacker to be able to eavesdrop on the public channel and, following Kerckhoffs' principle (Kerckhoffs, 1883), to know the complete specification of the cipher system. The only information about the system that he does not have is the secret key in use.

Performing a security analysis in this setting means to investigate how much effort it takes the attacker to reach his goal. Consequently, the more effort is required, the more secure we consider the system. Or put another way, the more powerful the attackers that a system is able to resist (i.e., whom the system is able to prevent from reaching their goal), the better.

One of Shannon's most important observations is the fact that the one-time pad (encrypting by XOR-synthesis of a binary message with an equally long random bit string) is information-theoretically secure, i.e., an attacker cannot recover the plaintext from the ciphertext even with unlimited computational power.

While this system is used to the present day by intelligence agencies for highly critical information, the requirement that the secret information (that has to be exchanged confidentially between sender and receiver in advance) has to be as long as the message makes it impractical for many important applications.

The way out is to trade off security and usability (in fact, a very common strategy in practical IT security), i.e., to relax the requirements of the system while hoping not to lose too many of its security properties.

In the case of cipher systems, the relaxation consists in limiting the secret information (most commonly called the *key*) to a size that is small enough to be efficiently exchanged between sender and receiver, and at the same time large enough for the system to resist attackers equipped with a realistic amount of resources.

Symmetric-key cipher implementations that are based on this idea can be classified into two categories, block ciphers and stream ciphers, which we describe in more detail in the following.

2.2 Block Ciphers

Suppose for the moment that we want to encrypt a plaintext block-wise (or word-wise) with a fixed block length l. In order to be able to decrypt, we use a bijective mapping (i.e., a permutation) $E : \{0,1\}^l \to \{0,1\}^l$ for encryption and its inverse $D = E^{-1}$ for decryption.

Ideally, we would like to choose E from all $2^l!$ possible permutations for a fixed block length l prior to the communication. However, only with very low probability, our choice will have a representation that is more efficient than a list of input-output pairs with 2^l entries, which is clearly too inefficient to be exchanged between sender and receiver for reasonable block sizes.

The compromise between security and usability in this case is to pick a set of 2^n permutations that can be efficiently implemented using a device that is parametrized with an n-bit string (the key) to determine which permutation it actually realizes. Such a device is commonly called a *block cipher*.

Definition 2.1. *A* block cipher *consists of two mappings*

$$\begin{array}{rccc} E: & \{0,1\}^l \times \{0,1\}^n & \to & \{0,1\}^l \\ & (x,k) & \mapsto & y \end{array}$$

and

$$\begin{array}{rccc} D: & \{0,1\}^l \times \{0,1\}^n & \to & \{0,1\}^l \\ & (y,k) & \mapsto & x \end{array}$$

that satisfy $D(E(x,k),k) = x$ *for all* $x \in \{0,1\}^l$ *and all* $k \in \{0,1\}^n$. *We call* E *the encryption function,* D *the decryption function,* l *the block length, and* n *the key length of the block cipher.*

Note that for a fixed $k \in \{0,1\}^n$, $E(\cdot, k)$ and $D(\cdot, k)$ are permutations (i.e., bijective mappings) and inverse to each other, and we may view the key as an identifier of a particular permutation. In this sense, picking a key means to fix a particular permutation, and choosing a particular block cipher of block length l means choosing 2^n out of the $2^l!$ possible permutations of $\{0,1\}^l$.

On the security side, we demand that an attacker cannot distinguish the block cipher setting from the ideal case, more precisely that the permutations provided by the block cipher be indistinguishable from a permutation that was randomly chosen from all possible permutations.

We note that an attacker who can distinguish a block cipher based permutation from a randomly chosen permutation may not necessarily be able to deduce information about the encrypted messages nor the key. But conversely, we can be sure that an attacker who cannot tell whether a random permutation or a block cipher is used cannot deduce any nontrivial information.

Block ciphers are among the most widely used cryptographic primitives, with the Data Encryption Standard (DES) and the Advanced Encryption Standard (AES) being particularly prominent examples (see Menezes et al. (2001) for detailed descriptions and security considerations).

Care has to be taken when a sequence of blocks $b_1, b_2, \ldots, b_m \in \{0,1\}^l$ has to be encrypted. For common block lenghts of 128 bits or more, this is the case in virtually any practical application. The most straightforward procedure, the *Electronic Codebook* (ECB) mode, which computes the ciphertexts as $c_i := E(b_i, k)$, implies that coinciding plaintext blocks will have coinciding corresponding ciphertext blocks and is therefore not recommended.

A more suitable mode is the *Cipher Block Chaining* (CBC) mode (Ehrsam et al., 1976), which is defined by $E^{\text{CBC}}((b_1, \ldots, b_m), k, \text{IV}) := (c_1, \ldots, c_m)$ with

$$c_i := \begin{cases} E(\text{IV} \oplus b_1, k) & \text{for } i = 1 \\ E(c_{i-1} \oplus b_i, k) & \text{for } 1 < i \leq m \end{cases} . \tag{2.1}$$

and a (usually publicly known) initialization vector $\text{IV} \in \{0,1\}^l$.

Many more block cipher modes of operation for different purposes exist, see Menezes et al. (2001) for an introduction and overview.

2.3 Dedicated Stream Ciphers

Besides block ciphers, dedicated constructions exist for (immediately, i.e., not waiting for the next block to be filled) encrypting data streams. These constructions are typically called stream ciphers.

In this thesis, we want to focus on constructions that try to approximate the one-time pad by producing from a short, fixed-length secret information a long random-looking sequence that is XOR-combined with the plaintext in order to obtain the ciphertext.

Consequently, the heart of most such stream ciphers is a keystream generator, which is initialized at the beginning of the conversation with a secret key $\mathcal{K}$. Many modern constructions accept an additional initialization vector IV that can be seen as a pointer into the keystream produced for $\mathcal{K}$. Cipher designs normally assume the IV to be public, such that it can be easily exchanged between sender and receiver.

Hence, the keystream generator produces keystream bits $(z_t)_{t\geq 0}$ that are added to the plaintext stream $(p_t)_{t\geq 0}$ on the sender's side in order to obtain the ciphertext stream $(c_t)_{t\geq 0}$ as $c_t := p_t \oplus z_t$ for all $t \geq 0$. The receiver uses the same cipher and the same initialization data $\mathcal{K}$ and IV as the sender in order to compute the keystream $(z_t)_{t\geq 0}$ himself and to recover the plaintext as $p_t = c_t \oplus z_t$ for all $t \geq 0$ (see Fig. 2.2).

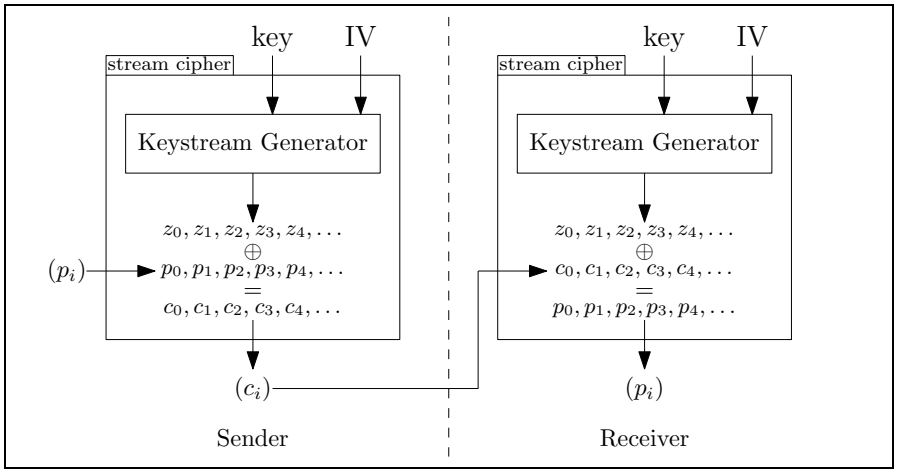

Figure 2.2: Stream cipher communication scenario

The keystream generator itself is often split into two components, a key/IV setup procedure and a finite state machine (FSM). The key/IV setup (or rather initial state setup) transforms the key and the IV into the initial state of the FSM. The FSM usually operates in a clocking-based manner, outputting a piece of keystream and updating its state in each clock cycle, hence producing the keystream sequence $(z_t)_{t\geq 0}$ (cf. Fig. 2.3).

More formally, the FSM is defined by a state update function $\delta : \{0,1\}^n \to \{0,1\}^n$ and a keystream function $g : \{0,1\}^n \to \{0,1\}^*$. In each clock t, keystream bits are produced according to $g(\omega(t))$ from the current state $\omega(t)$, and the internal state is updated to $\omega(t+1) = \delta(\omega(t))$. Hence, the output of the generator is completely determined by the starting state $\omega(0)$.

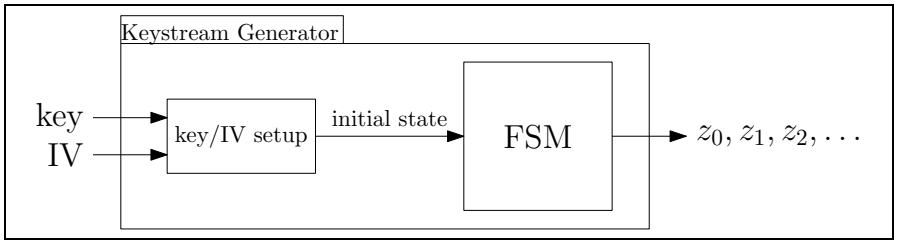

Figure 2.3: Common construction of the keystream generator

Definition 2.2. *We call an FSM-state* periodic *if, when running, the FSM will return to the same state after a finite number of steps.*

We will later need the notion of equivalent FSMs, which we define as follows.

Definition 2.3. *The FSMs M_1 and M_2 are called* equivalent *if for each possible starting state of M_1 there exists a corresponding starting state of M_2 and vice versa such that, when running, M_1 and M_2 produce the same output.*

Definition 2.4. *We call a sequence $\mathsf{u} = (u_i)_{i\geq 0}$* strictly periodic *(or simply* periodic*) with period T if $u_{i+T} = u_i$ for all $i \geq 0$. We call a sequence u* eventually periodic *if there exists a $t \geq 0$ such that $\mathsf{u}' = (u_i)_{i\geq t}$ is periodic.*

Definition 2.5. *For a (deterministic) finite state machine we can define a (directed) state transition graph as follows. The vertex set consists of the set of possible states, and there exists an edge from state u to state v if and only if v is the image of u under the state transition function.*

In order to approximate the one-time pad and its security features, the output of the keystream generator should look random, or more formally, the output should not be efficiently distinguishable from a truely random sequence. Therefore, the sequence should share as many properties with truely random sequences as possible.

The National Institute of Standards and Technology (NIST) maintains a collection of such properties and provides infrastructure for checking pseudorandom number generators against these properties (Rukhin et al., 2010).

We will exemplarily consider as properties the period length of the sequence, the number of occurrences of a particular block in one period of the sequence, and its autocorrelation.

Definition 2.6. *The* autocorrelation $\theta_\tau(\mathsf{u})$ of a binary sequence $\mathsf{u} = (u_i)_{i\geq 0}$ with shift τ *is the correlation of the sequences $(u_i)_{i\geq 0}$ and $(u_{i+\tau})_{i\geq 0}$, i.e.,*

$$\begin{aligned}\theta_\tau(\mathsf{u}) &:= \sum_{i\geq 0}(-1)^{u_i \oplus u_{i+\tau}} \\ &= |\{i : u_i \oplus u_{i+\tau} = 0\}| - |\{i : u_i \oplus u_{i+\tau} = 1\}| \\ &= |\{i : u_i = u_{i+\tau}\}| - |\{i : u_i \neq u_{i+\tau}\}| \ .\end{aligned} \tag{2.2}$$

Observation 2.7. *A truely random sequence is aperiodic, the probability of a τ-bit block's occurrence at position i in the sequence is $2^{-\tau}$, and its autocorrelation is zero-valued for all shifts τ.*

Consequently, we require that a keystream generator's output bitstream u satisfy the following postulates.

Pseudorandomness Postulate 1. *A keystream sequence should have a large period T (for many applications at least $T \geq 2^{50}$).*

Pseudorandomness Postulate 2. *A keystream seqeunce should contain a given τ-bit block around $T \cdot 2^{-\tau}$ times.*

Pseudorandomness Postulate 3. *For a keystream sequence u, $\frac{|\theta_\tau(\mathsf{u})|}{T}$ should be small for any shift $\tau < T$.*

2.4 Asymmetric Ciphers

It was assumed for a long time that a reasonable cipher system could only be symmetric, i.e., the encryption key k_e and the decreption key k_d had to be equal. Only in the 1970s, the first practical asymmetric cipher systems based on different keys for encryption and decryption were proposed, with the RSA cryptosystem being one of the most prominent examples (see, e.g., Vaudenay (2006) for an introduction).

In systems in which deducing the decryption key from the encryption key is infeasible (which is the case for practical asymmetric ciphers), there no need to keep the encryption key secret any more. Therefore, the encryption key k_e is often called the *public key* and the decryption key k_d is called the *private key*, and cipher systems that allow for publishing the encryption key are also called *public key (cipher) systems*. The receiver can publish his public key, and any potential sender can encrypt messages using this public key without the need for establishing a common key as in symmetric cipher systems.

However, asymmetric cipher systems usually require much more computational effort for encryption and decryption than symmetric cipher systems for comparable security levels, which limits their suitability for low-end devices.

Chapter 3

Stream Cipher Building Blocks

We now present the most important building blocks for stream ciphers, with a special focus on components that are particularly useful for hardware-oriented ciphers.

3.1 Boolean Functions

Definition 3.1. *We call a function*

$$\begin{array}{rccc} f: & \{0,1\}^n & \to & \{0,1\}^m \\ & (x_1,\ldots,x_n) & \mapsto & (y_1,\ldots,y_m) \end{array}$$

an m-output Boolean function *in n variables. We say that f* depends on the input x_i *if*

$$f(x_1,\ldots,x_{i-1},x_i,x_{i+1},\ldots,x_n) \neq f(x_1,\ldots,x_{i-1},x_i \oplus 1,x_{i+1},\ldots,x_n) \ .$$

Definition 3.2. *We call a Boolean function $f : \{0,1\}^n \to \{0,1\}$* balanced *if $|f^{-1}(0)| = |f^{-1}(1)|$.*

Observation 3.3. *Each Boolean function $f : \{0,1\}^n \to \{0,1\}$ can be equivalently represented in* algebraic normal form*, i.e., as a polynomial*

$$F(w_1,\ldots,w_n) = \bigoplus_{j\in M} m_j \text{ with monomials } m_j = \bigwedge_{l\in M^j} w_l \text{ and } M^j(f) \subseteq \{1,\ldots,n\} \ .$$

$|M^j(f)|$ is called the degree *of the monomial m_j. The degree of the polynomial F (abbreviated by $\deg(F)$) is defined to be the maximum over the degrees of the monomials occuring in F.*

We call a Boolean function F with $\deg(F) = 1$ a *linear function.*

Definition 3.4. *For a binary vector $x = (x_1,\ldots,x_n) \in \{0,1\}^n$, we denote by the* Hamming weight *of x (abbreviated by $\mathrm{wt}(x)$) the number of non-zero components in x, i.e.,*

$$\mathrm{wt}(x) := |\{i \in \{1,\ldots,n\} | x_i \neq 0\}| \ .$$

For ease of notation, we will often implicitly identify a vector $(u_0, \ldots, u_{k-1}) \in \{0,1\}^k$ with the integer $u = \sum_{i=0}^{k-1} u_i 2^i$.

3.2 Feedback Shift Registers

Feedback shift registers have turned out to be particularly useful devices for producing bitstreams with good pseudorandomness properties.

Definition 3.5. *A* Feedback Shift Register (FSR) *in Fibonacci architecture consists of an n-bit register $a = (a_0, \ldots, a_{n-1})$ and a state update function $f : \{0,1\}^n \to \{0,1\}$. Starting from an initial configuration a^0, in each clock a_0 is produced as output and the register is updated according to $a := (a_1, \ldots, a_{n-2}, f(a_0, \ldots, a_{n-1}))$. Depending on whether f is a linear function, we call the register a* Linear Feedback Shift Register (LFSR) *or a* Nonlinear Feedback Shift Register (NFSR).

The FSR-construction is illustrated in Fig. 3.1.

The definition implies that the output bitstream $(w_t)_{t \geq 0}$ produced from an initial configuration $a^0 = (a_0^0, \ldots, a_{n-1}^0)$ can be expressed as

$$w_t = \begin{cases} a_t^0 & \text{for} \quad t \in \{0, \ldots, n-1\} \\ f(w_{t-n}, \ldots, w_{t-1}) & \text{for} \quad t \geq n \end{cases},$$

while the state of the FSR after t clockings corresponds to $(w_t, \ldots, w_{t+n-1})$.

Surprisingly, even after many decades of research, the properties of general FSRs and the sequences they produce are hardly understood. We therefore focus on two special cases, linear feedback shift registers and feeback shift registers with carry, which are much less resistant to analysis and have found their way into practical applications.

3.2.1 Linear Feedback Shift Registers (LFSRs)

Fibonacci and Galois representations of LFSRs

Definition 3.6. *An n-stage Linear Feedback Shift Register (LFSR) in Fibonacci architecture (see Fig. 3.2) contains a main register with n binary cells $(y_0, \ldots, y_{n-1})$ and fixed binary feedback taps $(d_0, \ldots, d_{n-1})$. From an initial state y, the LFSR outputs in each clock t the value y_0, computes the sum $\sigma = \sum_{i=0}^{n-1} y_i d_{n-i-1}$ over the integers and updates the register according to $y = (y_1, y_2, \ldots, y_{n-1}, \sigma \bmod 2)$.*

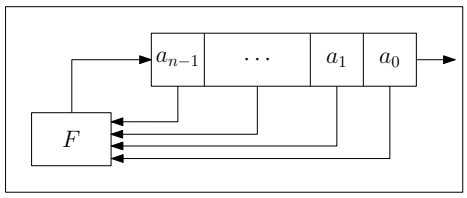

Figure 3.1: Feedback shift register (FSR) of length n

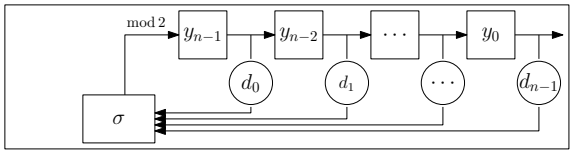

Figure 3.2: LFSR in Fibonacci architecture

Based on an initial configuration y^0, we can describe the output bitstream $(w_t)_{t\geq 0}$ of a Fibonacci LFSR by

$$w_t = \begin{cases} y_t^0 & \text{for} \quad t \in \{0, \ldots, n-1\} \\ \sigma_t \bmod 2 & \text{for} \quad t \geq n \end{cases} ,$$

where $\sigma_t = \sum_{i=1}^{n} w_{t-i} d_{i-1}$ for $t \geq n$.

Note that for performance reasons, the feedback bit ($\sigma \bmod 2$) is usually computed as

$$\sigma \bmod 2 = \bigoplus_{i=0}^{n-1} y_i d_{n-i-1} \ .$$

Additionally to the (most commonly used) Fibonacci architecture, there exists a Galois architecture for LFSRs.

Definition 3.7. *An n-stage LFSR in Galois architecture (see Fig. 3.3) contains n binary main register cells $(x_0, \ldots, x_{n-1})$ with fixed binary feedback taps $(d_0, \ldots, d_{n-1})$, $d_{n-1} \neq 0$. Starting from an initial state x, the Galois LFSR outputs in each clock the value x_0, computes the sums $\sigma_i = x_{i+1} + x_0 d_i$ for $0 \leq i < n$ (with $x_n = 0$) and updates x_i to σ_i mod 2 for all $0 \leq i < n-1$.*

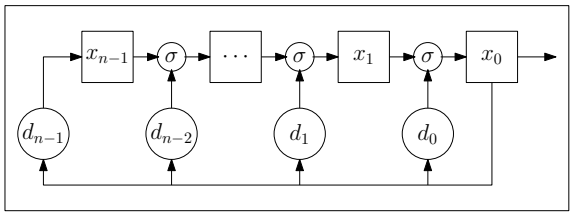

Figure 3.3: LFSR in Galois architecture

Again, we may equivalently compute the update value for x_i as $x_{i+1} \oplus x_0 d_i$.

Algebraic model: Formal Power Series and $\mathbb{F}_{2^n}$

We denote the ring of formal power series $\alpha(X) = \sum_{i=0}^{\infty} u_i X^i$ with $u_i \in \{0, 1\}$ (i.e., with coefficients in the integers modulo 2) by $\mathbb{F}_2[[X]]$, and the Galois field with 2^n elements by $\mathbb{F}_{2^n}$.

Theorem 3.8 (Golomb (1981)). *There is a one-to-one correspondence between quotients of polynomials $\alpha(X) = \frac{h(X)}{q(X)} \in \mathbb{F}_2[[X]]$ and eventually periodic binary sequences u which associates to each such quotient its coefficient sequence $\mathsf{u} = (u_0, u_1, \ldots)$. The sequence u is strictly periodic if and only if* $\deg(h(X)) < \deg(q(X))$.

For both the Fibonacci and the Galois architecture, we define the *connection polynomial* $q(X)$ by

$$q(X) := d_{n-1}X^n + d_{n-2}X^{n-1} + \ldots + d_0X - 1$$

and associate a Fibonacci state $(y_0, \ldots, y_{n-1})$ with the polynomial

$$h(X) = \sum_{k=0}^{n-1} \sum_{i=0}^{k} d_{i-1} y_{k-i} X^k, \text{ where } d_{-1} = 1 \ , \tag{3.1}$$

and a Galois state $(x_0, \ldots, x_{n-1})$ with the polynomial

$$h(X) = -\left(x_0 + x_1X + \ldots + x_{n-1}X^{n-1}\right) \ . \tag{3.2}$$

Theorem 3.9 (Golomb (1981)). *The output sequence of an LFSR with feedback tap vector corresponding to the connection polynomial $q(X)$ and an initial state corresponding to $h(X)$ is the coefficient sequence of $\alpha(X) = \frac{h(X)}{q(X)}$.*

Corollary 3.10. *The LFSR's output sequence is strictly periodic for any initial state.*

Proof. Since by the definition of $h(X)$, $\deg(h(X)) \leq n-1 < n = \deg(q(X))$, the claim follows from Theorem 3.8. □

The Fibonacci and Galois architectures can be related in the following way. Suppose that $\deg(q) = n$ and q is irreducible, let α denote a root of $q(X)$ in $\mathbb{F}_{2^n}$, express a $p \in \mathbb{F}_{2^n}$ as linear combination of the elements in $\{1, \alpha, \alpha^2, \ldots, \alpha^{n-1}\}$, and define

$$\begin{array}{rccc} T: & \mathbb{F}_{2^n} & \to & \{0,1\} \\ & p_0 + p_1\alpha + \ldots + p_{n-1}\alpha^{n-1} & \mapsto & p_0 \ . \end{array} \tag{3.3}$$

For periodic Galois states x, we define a mapping E by

$$\begin{array}{rccc} E: & \{\text{periodic Galois states}\} & \to & \mathbb{F}_{2^n} \\ & (x_0, \ldots, x_{n-1}) & \mapsto & x_0 + x_1\alpha + x_2\alpha^2 + \ldots + x_{n-1}\alpha^{n-1} \ , \end{array} \tag{3.4}$$

For an element $p \in \mathbb{F}_{2^n}$, we define a mapping S by

$$\begin{array}{rccc} S: & \mathbb{F}_{2^n} & \to & \{\text{periodic Fibonacci states}\} \\ & p & \mapsto & \left(T(\alpha^{1-n}p), T(\alpha^{2-n}p), \ldots, T(\alpha^{-1}p), T(p)\right) \ , \end{array} \tag{3.5}$$

i.e., $y_i = T(\alpha^{-i}p)$.

Theorem 3.11 (Goresky and Klapper (2002)). *The mappings E and S are one-to-one, i.e., there exist inverse functions E^{-1} and S^{-1} that map elements of $\mathbb{F}_{2^n}$ to the set of periodic Galois states and periodic Fibonacci states to $\mathbb{F}_{2^n}$, respectively.*

We note that since $\{1, \alpha^{-1}, \ldots, \alpha^{1-n}\}$ is a basis for $\mathbb{F}_{2^n}$ over $\mathbb{F}_2$, E^{-1} and S^{-1} may be efficiently computed by solving systems of linear equations in $\{x_0, \ldots, x_{n-1}\}$ and p, respectively.

We can now describe the evolution of the LFSR states (resp. their $\mathbb{F}_{2^n}$-representations) in the following way.

Theorem 3.12 (Golomb (1981), Goresky and Klapper (2002)). *For an initial LFSR state corresponding to $p \in \mathbb{F}_{2^n}$, the sequence $(p^t)_{t\geq 0}$ of $\mathbb{F}_{2^n}$-representations of the register state at time t is given by $p^t = \alpha^{-t}p \in \mathbb{F}_{2^n}$, and the t-th output bit of the register can be computed as $z^t = T(\alpha^{-t}p) \in \{0,1\}$. The period of the sequence $(p^t)_{t\geq 0}$ equals the order of α in $\mathbb{F}_{2^n}$.*

Corollary 3.13 (Golomb (1981), Goresky and Klapper (2002)). *If $q(X)$ is not only an irreducible but also a primitive polynomial, α has the maximum possible order $2^n - 1$ and hence the period, too, reaches its maximum $2^n - 1$.*

Consequently, we call LFSRs with primitive connection polynomials *maximum-lengh* LFSRs and the sequences they produce *m-sequences*.

Remark 3.14. *There are $\frac{\varphi(2^n-1)}{n}$ primitive polynomials of degree $n \geq 1$ over $\mathbb{F}_2$, where for $m \in \mathbb{N}$, $\varphi(m) = \{i \in \{1,\ldots,m\} | \gcd(i,m) = 1\}$.*

Sequences produced by individual Register Cells

We now want to describe the sequences of values taken by a particular LFSR register cell. In the case of Fibonacci LFSRs, the following observation is straightforward to make.

Theorem 3.15. *For an n-stage Fibonacci LFSR with connection polynomial $q(X)$ and initial state y^0, the sequence of values $(y_i^t)_{t\geq 0}$ taken by the i-th register cell y_i is the original sequence shifted by i positions, i.e., given by the Fibonacci LFSR-sequence with connection polynomial $q(X)$ produced from the initial state $S(\alpha^{-i}S^{-1}(x))$ with α a root of $q(X)$.*

A similar correspondence holds for Galois LFSRs.

Theorem 3.16. *For an n-stage Galois LFSR with connection polynomial $q(X)$ and initial state polynomial $h(X)$, the sequence of values taken by the i-th register cell x_i is the sequence produced by a Galois LFSR with connection polynomial $q(X)$ and initial state polynomial*

$$h_i(X) = x_i(0) \cdot q(X) + X \cdot (h_{i+1}(X) + d_i h_0(X)) \text{ with } h_n(X) \equiv 0.$$

Proof. Obviously, $h_0(X) = h(X)$. Since $\deg(q) = n$, we have $d_{n-1} = 1$, which implies $x_{n-1}(t+1) = x_0(t)$. We obtain for $i = n-1$

$$\begin{aligned}\frac{h_{n-1}(X)}{q(X)} &= \sum_{t=0}^{\infty} x_{n-1}(t) \cdot X^t = x_{n-1}(0) + X \cdot \sum_{t=0}^{\infty} x_{n-1}(t+1)X^t \\ &= x_{n-1}(0) + X \cdot \sum_{t=0}^{\infty} x_0(t) \cdot X^t \\ &= x_{n-1}(0) + X \cdot \frac{h_0(X)}{q(X)} \ ,\end{aligned}$$

and therefore

$$h_{n-1}(X) = x_{n-1}(0) \cdot q(X) + X \cdot h_0(X) \ .$$

For $0 \leq i < n-1$, we have

$$\begin{aligned}\frac{h_i(X)}{q(X)} &= \sum_{t=0}^{\infty} x_i(t) \cdot X^t = x_i(0) + X \cdot \sum_{t=0}^{\infty} x_i(t+1) \cdot X^t \\ &= x_i(0) + X \cdot \sum_{t=0}^{\infty} (x_{i+1}(t) + d_i x_0(t)) \cdot X^t \\ &= x_i(0) + X \cdot \left(\frac{h_{i+1}(X)}{q(X)} + d_i \frac{h_0(X)}{q(X)} \right) \ ,\end{aligned}$$

which implies

$$h_i(X) = x_i(0) \cdot q(X) + X \cdot (h_{i+1}(X) + d_i h_0(X)) \ . \qquad \square$$

We can write the relation for $h_i(X)$ in closed form as follows.

Lemma 3.17. *The recurrence relation*

$$h_i(X) = x_i(0) \cdot q(X) + X \cdot (h_{i+1}(X) + d_i h_0(X)) \ \textit{with} \ h_n(X) \equiv 0$$

can be expressed as $h_i(X) = F_i(x) \cdot q(X) + M_i \cdot h_0(X)$ *with*

$$M_i = X \cdot \sum_{j=i}^{n-1} d_j X^{j-i} \ \textit{and} \ F_i(x) = \sum_{j=i}^{n-1} x_j(0) X^{j-i} \ .$$

Proof. The claimed formula is straightforwardly obtained by induction. $\square$

Mappings between periodic Galois and Fibonacci states

Proposition 3.18. *There exists a bijective mapping between periodic initial Galois LFSR states and periodic initial Fibonacci LFSR states such that the registers produce the same output (see Fig. 3.4).*

Proof. According to Theorem 3.11, the mapping

$$\begin{array}{rccl} \Phi: & \{\text{periodic Galois states}\} & \rightarrow & \{\text{periodic Fibonacci states}\} \\ & x & \mapsto & S(E(x)) \end{array}$$

with E and S defined as in Eqs. (3.4) and (3.5) is one-to-one. $\square$

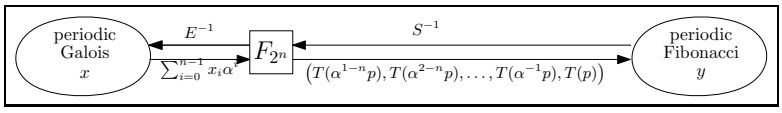

Figure 3.4: Mapping between periodic Galois and Fibonacci LFSR states

Lemma 3.19. *The value x_i of the i-th cell in the main register of a Galois LFSR can be computed in polynomial time from the state y of the corresponding Fibonacci LFSR as the i-th component of the vector $E^{-1}(S^{-1}(x))$.*

Proof. The claim follows immediately from Theorem 3.11. $\square$

Statistical Properties of m-Sequences

m-sequences are statistically very similar to truely random sequences. Concerning the three properties that we selected in Section 2.3, their behaviour can be characterized as follows.

Observation 3.20 (Golomb (1981)). *Consider an m-sequence u produced by an n-stage LFSR.*

- *The period of u is $T = 2^n - 1$.*
- *Any τ-bit block B occurs in one period of u exactly $2^{n-\tau}$ times if $B \neq 0$ and $2^{n-\tau} - 1$ times if $B = 0$.*
- *The autocorrelation $\theta_\tau(\mathsf{u})$ satisfies $\frac{|\theta_\tau(\mathsf{u})|}{T} = \frac{1}{2^n - 1}$.*

The definition of LFSRs suggests another pseudorandomness critereon, the linear complexity.

Definition 3.21. *The* linear complexity *of a binary sequence $\mathsf{u} = (u_i)_{i \geq 0}$ (abbreviated by $\mathrm{lc}(\mathsf{u})$) is the length of the shortest LFSR that generates the sequence.*

Lemma 3.22. *A sequence $\mathsf{u} = (u_i)_{i \geq 0}$ with period T satisfies $\mathrm{lc}(\mathsf{u}) \leq T$.*

Proof. The T-stage Fibonacci LFSR with feedback taps $(0, \ldots, 0, 1) \in \{0,1\}^T$ will obivously generate u from the initial state $y = (u_0, \ldots, u_{T-1})$. □

Remark 3.23. *There exists an algorithm that computes for a given sequence u with $l = \mathrm{lc}(\mathsf{u})$ in time $\mathcal{O}(l^3)$ and from the first $2l$ bits of u the value l and the feedback tap vector of an l-stage LFSR that generates u. This algorithm is known as the Berlekamp-Massey algorithm for register synthesis (see, e.g., Menezes et al. (2001) for a description).*

We conclude that the linear complexity of a keystream sequence should be large enough such that a generating LFSR cannot be determined with realistic resources.

Pseudorandomness Postulate 4. *The linear complexity $\mathrm{lc}(\mathsf{u})$ of a keystream sequence u should be reasonably large.*

We note that although the period of an m-sequence is $T = 2^n - 1$, its linear complexity is only n, i.e. logarithmic in T, and therefore much lower than the upper bound given by Lemma 3.22. Conversely, if an LFSR is to produce a sequence with linear complexity l^*, its required minimum size is exponential in l^*, which is impractical for most applications. In fact, this is the main reason why LFSRs – despite their many other desirable statistical properties – are not suitable for direct use as keystream generators.

3.2.2 Feedback Shift Registers With Carry (FCSRs)

Feedback with carry shift registers (FCSRs) have been discussed since the mid-1990s in the context of efficient pseudorandom number generation, particularly as an alternative to LFSRs (Couture and L'Ecuyer, 1994, Klapper and Goresky, 1997, Marsaglia and Zaman, 1992).

Analogously to Section 3.2.1, we describe the structure of FCSRs and make some observations on the properties of their output sequences. All our results have been experimentally confirmed with the computer algebra system MAGMA (Bosma et al., 1997).

Fibonacci and Galois representations of FCSRs

Definition 3.24. *An n-stage FCSR in Fibonacci architecture (see Fig. 3.5) contains a main register with n binary cells $(y_0, \ldots, y_{n-1})$ and fixed binary feedback taps $(d_0, \ldots, d_{n-1})$ as well as an additional memory b. From an initial state (y, b), the FCSR outputs in each clock t the value y_0, computes the sum $\sigma = b + \sum_{i=0}^{n-1} y_i d_{n-i-1}$ over the integers and updates the register and memory according to $b = \sigma \text{ div } 2$ and $y = (y_1, y_2, \ldots, y_{n-1}, \sigma \bmod 2)$.*

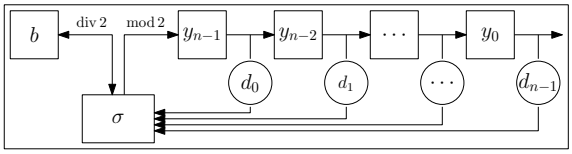

Figure 3.5: FCSR in Fibonacci architecture

Based on an initial configuration (y^0, b^0), we can describe the output bitstream $(w_t)_{t\geq 0}$ of a Fibonacci FCSR by

$$w_t = \begin{cases} y_t^0 & \text{for} \quad t \in \{0, \ldots, n-1\} \\ \sigma_t \bmod 2 & \text{for} \quad t \geq n \end{cases} ,$$

where $\sigma_t = b^{t-n} + \sum_{i=1}^{n} w_{t-i} d_{i-1}$ and $b^{t-n+1} = \sigma_t \text{ div } 2$ for $t \geq n$, which implies

$$\sigma_t = (\sigma_{t-1} \text{ div } 2) + \sum_{i=1}^{n} w_{t-i} d_{i-1} \text{ with } \sigma_{n-1} = 2b^0 \ . \tag{3.6}$$

We note that in general, b may be an arbitrarily large value. However, if the register's state is periodic, b may be bounded as follows.

Proposition 3.25 (Klapper and Goresky (1997)). *If the Fibonacci FCSR is in a periodic state, the value of the memory b satisfies $0 \leq b < \mathrm{wt}(d+1)$.*

Corollary 3.26. *A Fibonacci FCSR with a periodic initial state will not require more than $\lfloor \log_2(\mathrm{wt}(d+1) - 1) \rfloor + 1$ bits to store the value b at any time.*

Similarly to the Galois architecture for LFSRs, there exists a Galois architecture for FCSRs, which was first observed by Noras (1997) and further analyzed by Goresky and Klapper (2002).

Definition 3.27. *An n-stage FCSR in Galois architecture (see Fig. 3.6) contains n binary main register cells $(x_0, \ldots, x_{n-1})$ with fixed binary feedback taps $(d_0, \ldots, d_{n-1})$, $d_{n-1} \neq 0$, and $n-1$ memory cells $(a_0, \ldots, a_{n-2})$. Starting from an initial state (x, a), the Galois FCSR outputs in each clock the value x_0, computes the sums $\sigma_i = x_{i+1} + a_i d_i + x_0 d_i$ for $0 \leq i < n$ (with $x_n = 0$ and $a_{n-1} = 0$) and updates x_i to $\sigma_i \bmod 2$ and a_i to $\sigma_i \text{ div } 2$ for all $0 \leq i < n-1$.*

We will assume that memory cells are only present at those positions with feedback taps, i.e., $a_i = 0$ if $d_i = 0$ for all $0 \leq i < n-1$, since $a_i = 0$ for all i with $d_i = 0$ is a necessary condition for periodic states (x, a).

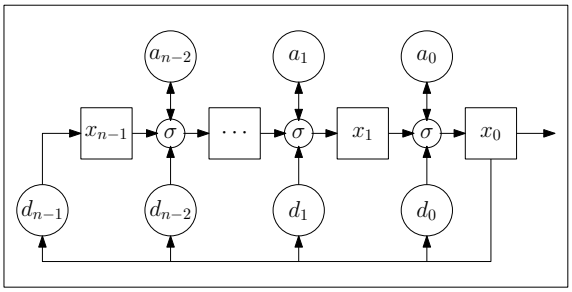

Figure 3.6: FCSR in Galois architecture

Implementors will often prefer the Galois architecture to the Fibonacci architecture since the size of the memory is intrinsically limited and the memory bits can be updated in parallel, with each addition involving at most three bits.

Algebraic model: 2-adic Numbers and $\mathbb{Z}_2$

The algebraic structure that is associated with FCSRs is the ring of 2-adic numbers. A 2-adic integer is a formal power series $\alpha = \sum_{i=0}^{\infty} u_i 2^i$ with $u_i \in \{0, 1\}$. The collection of all such formal power series forms the ring of 2-adic numbers. This ring especially contains rational numbers p/q, where p and q are integers and q is odd. 2-adic numbers and eventually periodic sequences are linked by the following Theorem.

Theorem 3.28 (Klapper and Goresky (1997)). *There is a one-to-one correspondence between rational numbers $\alpha = p/q$ (with odd q) and eventually periodic binary sequences u which associates to each such rational number α the bit sequence $\mathsf{u} = (u_0, u_1, \ldots)$ of its 2-adic expansion. The sequence u is strictly periodic if and only if $\alpha \leq 0$ and $|\alpha| \leq 1$.*

For both FCSR architectures, we define the *connection integer* q as $q = 1 - 2d$. We identify a Galois state (x, a) with the integer

$$p = x + 2a \tag{3.7}$$

and a Fibonacci state (y, b) with the integer

$$p = b2^n - \sum_{k=0}^{n-1} \sum_{i=0}^{k} q_i y_{k-i} 2^k \quad . \tag{3.8}$$

Theorem 3.29 (Klapper and Goresky (1997)). *The output sequence of an FCSR with feedback tap vector corresponding to the connection integer q and an initial state corresponding to p is the 2-adic expansion of $\alpha = \frac{p}{q}$.*

Corollary 3.30. *The output sequence of an FCSR with feedback tap vector corresponding to q will be strictly periodic if and only if the integer p that corresponds to the initial state satisfies $0 \leq p \leq |q|$.*

Proof. The claim is an immediate consequence of Theorem 3.28. □

Theorem 3.29 justifies the following definition.

Definition 3.31. *We call two Galois states* (x, a) *and* (x', a') equivalent *if*

$$x + 2a = x' + 2a' \ .$$

Similarly, we call two Fibonacci states (y, b), (y', b') equivalent *if*

$$b2^n - \sum_{k=0}^{n-1}\sum_{i=0}^{k} q_i y_{k-i} 2^k = b'2^n - \sum_{k=0}^{n-1}\sum_{i=0}^{k} q_i y'_{k-i} 2^k \ .$$

Note that although equivalent states produce the same output, the sequence of states $((x(t), a(t))_{t\geq 0}$ in the Fibonacci case and $(y(t), b(t))_{t\geq 0}$ in the Galois case) obtained by running the FCSR from equivalent starting states may be different.

Similarly to the LFSR-case, we can now describe the evolution of the FCSR states based on their representations in the set $\mathbb{Z}/(q\mathbb{Z})$ of integers modulo q, which we denote for simplicity by $\mathbb{Z}_q$.

Theorem 3.32 (Klapper and Goresky (1997)). *For an initial state corresponding to* $p \in \mathbb{Z}_{|q|}$, *the sequence of integer representations of the states* $(p^t)_{t\geq 0}$ *is given by* $p^t = 2^{-t}p \bmod q$ *and the* t*-th output bit can be computed as* $z^t = p^t \bmod 2 = (2^{-t}p \bmod q) \bmod 2$. *If* $0 < p < |q|$, q *odd, and* p *and* q *are coprime, then the period of the sequence* $(p^t)_{t\geq 0}$ *equals the order of* 2 *modulo* q . *For* $p = 0$ *and* $p = |q|$, *the FCSR produces the 2-adic expansions of* $0/q = 0$ *and* $|q|/q = -1$, *respectively, which both have period one. If* $0 < p < |q|$, q *odd, and* p *and* q *are coprime, then the period of the sequence* $(p^t)_{t\geq 0}$ *equals the order of* 2 *modulo* q.

Corollary 3.33. *If* q *is a (negative) prime for which* 2 *is a primitive root, the period reaches its maximum* $|q| - 1$.

Consequently, we call FCSRs with prime connection integers for which w is a primitive root *maximum-length* FCSRs and the sequences they produce *l-sequences*.

In contrast to the number of primitive polynomals in the LFSR-case, the number of connection integers q producing l-sequences is not known with certainty. However, there exists the following conjecture.

Conjecture 3.34 (Hooley (1967), Klapper (2004)). *The number of primes* q *of bitlength* n *for which the order of* 2 *modulo* q *is* $q-1$ *is asymptotically* $\frac{cn}{\log(n)}$, *where* $c \approx 0.37$ *is a constant.*

For a maximum-length Galois FCSR with connection integer q, the state transition graph (see Definition 2.5) has exactly three connected components, i.e., the two fixed points $(0, 0)$ and $(2^{n-1}, d - 2^{n-1})$ (corresponding to $p = 0$ and $p = |q|$) and a component containing all the remaining states. This component consists of a main cycle of length $|q| - 1$ and paths of lengths at most $n + 4$ leading to it (Arnault et al., 2008). In other words:

Observation 3.35. *An* n*-stage maximum-length Galois FCSR will be in a periodic state after at most* $n + 4$ *clockings.*

Sequences produced by individual Register Cells

As for LFSRs, the sequences produced by individual main register cells of an FCSR are again FCSR-sequences.

Theorem 3.36. *For a Fibonacci FCSR with connection integer q and an initial state corresponding to the integer p, the sequence $(y_i^t)_{t\geq 0}$ of values taken by the main register cell y_i is the FCSR sequence given by the 2-adic expansion of p_i/q with $p_i = 2^{-i}p$.*

Theorem 3.37 (Arnault and Berger (2005a), Theorem 4). *For a Galois FCSR with initial state (x, a) and $p = x+2a$, the sequence $(x_i^t)_{t\geq 0}$ of values taken by the main register cell i is again an FCSR-sequence, more precisely the 2-adic expansion of p_i/q with $p_i = F_i(x, a) \cdot q + M_i \cdot p$, $F_i(x, a) = \sum_{j=i}^{n-1}(x_j + 2a_j)2^{j-i}$, and with constants $M_i = 2\sum_{j=i}^{n-1} d_j 2^{j-i}$.*

It is interesting to note (and will prove useful in Section 3.2.2) that if the initial state (x, a) is periodic, this expression can be further simplified as follows.

Proposition 3.38. *For a maximum-length Galois FCSR with connection integer q, a periodic initial state (x^0, a^0), and $p^t = x^t + 2a^t$, the sequence $(x_i^t)_{t\geq 0}$ of values taken by a fixed main register cell i corresponds to $(p^{t+s_i} \bmod 2)_{t\geq 0}$ with $s_i = -\log_2(M_i) \bmod q$ and $M_i = 2\sum_{j=i}^{n-1} d_j 2^{j-i}$.*

Proof. If (x^0, a^0) is periodic, the 2-adic expansions of p_i/q have to be strictly periodic for all i. Theorem 3.28 implies that $0 \leq p_i < |q|$, hence $p_i = p_i \bmod q = M_i \cdot p^0 \bmod q$. In a maximum-length Galois FCSR, each possible value of $p_i \bmod q$ is passed after s_i iterations, hence $p_i = 2^{-s_i}p^0 \bmod q$, and we have $M_i = 2^{-s_i} \bmod q$. □

Proposition 3.38 implies that the sequence $(x_i^t)_{t\geq 0}$ corresponds to the sequence produced by the whole FCSR (i.e., $(x_0^t)_{t\geq 0}$) shifted by s_i positions. Note that the phase shifts s_i are independent of the initial state p and depend on i (and q) only.

Example 3.39. *Consider the toy example of Arnault and Berger (2005a) with $q = -347$, hence $n = 8$ and $d = 174$. The output of the FCSR is strictly periodic with period $-q - 1 = 346$. We find $M_0 = 1$, $M_1 = 174$, $M_2 = 86$, $M_3 = 42$, $M_4 = 20$, $M_5 = 10$, $M_6 = 4$, $M_7 = 2$. The phase shifts are $s_0 = 0$, $s_1 = 1$, $s_2 = 23$, $s_3 = 250$, $s_4 = 67$, $s_5 = 68$, $s_6 = 344$, $s_7 = 345$.*

Mappings between periodic Galois and Fibonacci States

There is an onto function

$$\begin{array}{rccl} E: & \{\text{periodic Galois states}\}\backslash\{(1,\ldots,1;a_0,\ldots,a_{n-2})\} & \to & \mathbb{Z}_{|q|} \\ & (x,y) & \mapsto & x + 2a \bmod q \end{array} \tag{3.9}$$

that assigns to a Galois state an element of $\mathbb{Z}_{|q|}$.

Moreover, there exists a one to one mapping S from $\mathbb{Z}_{|q|}$ onto the set of strictly periodic states of the Fibonacci FCSR with connection integer q except

for the state $(1,\ldots,1;\mathrm{wt}(q+1)-1)$, namely

$$\begin{array}{rccl} S: & \mathbb{Z}_{|q|} & \to & \{\text{periodic Fibonacci states}\}\backslash\{(1,\ldots,1;\mathrm{wt}(q+1)-1)\} \\ & p & \mapsto & (y,b) \end{array} \tag{3.10}$$

with

$$y_i = ((2^{-i}p \bmod q) \bmod 2) \text{ for } 0 \le i \le n-1$$

and

$$b = \frac{1}{2^n}\left(p + \sum_{k=0}^{n-1}\sum_{j=0}^{k} d_{j-1}y_{k-j}2^k\right) \quad .$$

Conversely, for a given periodic Fibonacci state (y,b) the corresponding integer p will satisfy $0 \le p < |q|$.

Hence, for an arbitrary initial state of a Galois FCSR with connection integer q, we can compute a periodic initial state of a Fibonacci FCSR with connection integer q and vice versa such that the two registers will produce the same output (Goresky and Klapper, 2002).

Obviously, the mapping E from the Galois states to $\mathbb{Z}_{|q|}$ is not one to one, i.e., generally more than one state is mapped to the same $p \in \mathbb{Z}_{|q|}$. However, the following Proposition shows how to compute for given $p \in \mathbb{Z}_{|q|}$ the uniquely determined corresponding periodic state (x,a).

Proposition 3.40. *For all $p \in \mathbb{Z}_{|q|}$, the only strictly periodic state (x,a) with $x+2a=p$ of a maximum-length Galois FCSR of size n with connection integer q is given by $x_i = M_i \cdot p \bmod q \bmod 2$ and $a=(p-x)/2$ with M_i defined as in Proposition 3.38.*

Proof. We first observe that $x+2a = x+2\frac{p-x}{2} = p$, hence (x,a) corresponds to p. If $p=0$, we have $(x,a)=(0,0)$ at all times, so (x,a) is periodic. Similarly for $p=|q|$, the only possible state (x,a) is $(2^n-1, d-2^{n-1})$, and this state is periodic (see Section 3.2.2). If $p \neq 0$, the state transition graph representing the evolution of the states consists of a main cycle of length $|q|-1$ and paths converging to it. Hence, for each initial state (x',a') with $x'+2a'=p$, there exists exactly one state $(\tilde{x},\tilde{a})$ with $\tilde{x}+2\tilde{a}=p$ that lies on the main cycle. For this state $(\tilde{x},\tilde{a})$, the sequences $(\tilde{x}_i^t)_{t\ge 0}$ have to be strictly periodic. Due to Proposition 3.38, the first bit of the 2-adic expansion of p_i/q and hence $\tilde{x}_i$ is equal to $p_i \bmod 2$ with $p_i = M_i \cdot p \bmod q$. Moreover, $\tilde{a}$ is uniquely determined by $\tilde{x}$ and p, which implies $(\tilde{x},\tilde{a}) = (x,a)$. □

Proposition 3.40 provides a possible answer to the open question raised by Goresky and Klapper (2002) how to intrinsically characterize the periodic states corresponding to a particular $p \in \mathbb{Z}_{|q|}$ and allows us to link periodic Fibonacci and periodic Galois states similarly to the LFSR-case.

Corollary 3.41. *There exists a bijective mapping between periodic initial Galois FCSR states and periodic initial Fibonacci FCSR states such that registers produce the same output (see Fig. 3.7).*

Proof. Proposition 3.40 implies a mapping $\tilde{E}$ of periodic Galois states onto $\mathbb{Z}_{|q|}$. With E and S defined by Eqs. (3.9) and (3.10), the claim follows. □

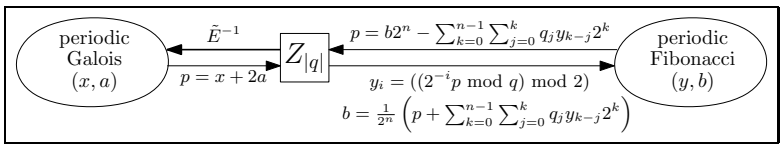

Figure 3.7: Mapping between periodic Galois and Fibonacci FCSR states

Example 3.42. *Continuing Example 3.39, let* $q = -347$*. For* $p = 100$*, we compute* $x_i = M_i \cdot p \mod q \mod 2$*, which yields* $x = (01010000)_2 = 64 + 16 = 80$*, and obtain* $a = (p - x)/2 = 10$*. Hence, the strictly periodic initial state corresponding to* $p = 100$ *is* $(x, a) = (80, 10)$*. Plugging the values of* p *and* q *into Eq.* (3.10) *yields the corresponding periodic Fibonacci state* $(y, b) = (148, 2)$*.*

Finally, we may obtain the sequence produced by a Galois main register cell from a Fibonacci FCSR as follows.

Lemma 3.43. *The value* x_i *of the* i*-th cell in the main register of a Galois FCSR can be computed from the strictly periodic state* (y, b) *of the corresponding Fibonacci FCSR by*

$$x_i = M_i \left(b2^n - \sum_{k=0}^{n-1} \sum_{j=0}^{k} d_{j-1} y_{k-j} 2^k \right) \bmod q \bmod 2 \ .$$

Proof. The claimed formula is an immediate consequence of Eq. (3.8) and Propositions 3.38 and 3.40. □

Statistical properties of l-sequences

The period T of an l-sequence produced by an n-stage FCSR with connection integer q is $|q| - 1$ (Klapper and Goresky, 1997), i.e., $2^{n-1} - 1 < T < 2^n - 1$. The linear complexity of l-sequences is close to $\frac{|q|-1}{2}$ (Tian and Qi, 2009).

Theorem 3.44 (Blum et al. (1986), Goresky and Klapper (2006)). *Let* u *be an* l*-sequence with connection integer* q*. The number of occurrences of any block* $e = (e_0, e_1, \ldots, e_{\tau-1})$ *of size* τ *in* u *varies at most by* 1 *as the block varies over all* 2^τ *possibilities. That is, there is an integer* w' *so that every block of length* τ *occurs either* w' *times or* $w' + 1$ *times in* u*. The number of blocks of length* τ *that occur* $w' + 1$ *times is* $(q - 1) mod\ 2^\tau$*, and the number of blocks of length* τ *that occur* w' *times is* $2^\tau - ((q - 1) mod\ 2^\tau)$*.*

We are especially interested in the occurrences of a particular block $B = (b_0, \ldots, b_{\tau-1})$ in one period of a sequence u with period length m, i.e., in the indices i, $0 \le i < m$, such that

$$u_{i \bmod m} = b_0, u_{(i+1) \bmod m} = b_1, \ldots, u_{(i+\tau-1) \bmod m} = b_{\tau-1} \ .$$

Theorem 3.45. *Let* u *be an* l*-sequence with connection integer* q*. Then for all* $0 < \tau < q$*, the number of occurences of any block* $B = (b_0, \ldots, b_{\tau-1})$ *in a period of* u *is* $\lfloor (q - 1 - v)/2^\tau \rfloor + 1$ *if* $v \neq 0$ *and* $\lfloor (q - 1 - v)/2^\tau \rfloor$ *if* $v = 0$*, where* $v = (-q \cdot \sum_{i=0}^{\tau-1} b_i 2^i) mod\ 2^\tau$*.*

This result was essentially observed by Klapper (2004). We provide an alternative proof illustrating some methods that will be useful in the remainder of this section.

Proof (Theorem 3.45). Identify the block B with the integer $\beta := \sum_{i=0}^{\tau-1} b_i 2^i$, $0 \le \beta < 2^\tau$. The number of occurrences of B in a period of u equals the number of shifts of u starting with B, which in turn equals the number of integers u with $-u/q \equiv \beta \bmod 2^\tau$ and $0 < u < q$. Since q is invertible mod 2^τ, we have $u \equiv -q\beta \bmod 2^\tau$. Set $v = -q\beta \bmod 2^\tau \in \{0, \ldots, 2^\tau - 1\}$. If $v \le q-1$, the set of integers u fulfilling this condition can be written as

$$\begin{array}{ll} \{v, v+2^\tau, \ldots, v + \lfloor \frac{q-1-v}{2^\tau} \rfloor \cdot 2^\tau\} & \text{for } v \neq 0 \\ \{v+2^\tau, \ldots, v + \lfloor \frac{q-1-v}{2^\tau} \rfloor \cdot 2^\tau\} & \text{for } v = 0 \end{array} .$$

The size of this set is $\lfloor (q-1-v)/2^\tau \rfloor + 1$ if $v \neq 0$ and $\lfloor (q-1-v)/2^\tau \rfloor$ if $v = 0$.

If $v > q-1$, the block B will not appear in u. In this case, we have $-2^\tau \le q-1-v < 0$, which means $\lfloor (q-1-v)/2^\tau \rfloor = -1$ and therefore $\lfloor (q-1-v)/2^\tau \rfloor + 1 = 0$. □

The expected autocorrelation of l-sequences can be shown to be zero (Xu and Qi, 2006). For any given shift τ, the autocorrelation is in $\mathcal{O}(\ln^2 q)$ (Xu et al., 2009), but how to compute its exact value is believed to be difficult (Goresky and Klapper, 1997) and is only known for $q = p^e$, where p is prime and $e \ge 2$, and τ of a special form (Xu and Qi, 2006).

We now describe a method for computing the exact value of the autocorrelation which is based on counting the number of occurrences of particular $(\tau+1)$-bit blocks $B = (b_\tau, b_{\tau-1}, \ldots, b_0)$ in the sequence. The main idea is to fix the first and the last bit in B and to compute the correlation based on how often these restricted blocks occur in the sequence. Hence, we define

$$B_{ij}^\gamma := \text{ number of blocks } B = (i, b_{\tau-1}, b_{\tau-2}, \ldots, b_1, j) \text{ that occur } \gamma \text{ times}$$

for $(i,j) \in \{0,1\}^2$ and compute the correlation as

$$\theta_\tau = \left(\sum_\gamma \gamma \cdot B_{00}^\gamma + \sum_\gamma \gamma \cdot B_{11}^\gamma \right) - \left(\sum_\gamma \gamma \cdot B_{10}^\gamma + \sum_\gamma \gamma \cdot B_{01}^\gamma \right) . \tag{3.11}$$

Theorem 3.45 implies that a given $(\tau+1)$-bit block B_{ij}^γ occurs either w times or $w+1$ times in a period of u, which means that $\gamma \in \{w, w+1\}$ in Eq. (3.11).

We now want to characterize more precisely the sets of τ-bit blocks that occur equally often.

Lemma 3.46. *The block $B = (b_0, b_1, \ldots, b_{\tau-1})$ occurs $w+1$ times in a period of u if $\beta = \sum_{i=0}^{\tau-1} b_i 2^i$ fulfills $0 < -q\beta \bmod 2^\tau \le q-1 \bmod 2^\tau$, and w times otherwise, where $w = \lfloor 2^{k-\tau} \rfloor + ((q \bmod 2^k) \operatorname{div} 2^\tau)$ and $k = \lfloor \log_2(q) \rfloor$*

Proof. Let $v = -q\beta \bmod 2^\tau$, hence $0 \le v < 2^\tau$. We first consider the case $\tau \le k$ and define $e = q \bmod 2^k$, $x = e \operatorname{div} 2^\tau$ and $y = e \bmod 2^\tau$. Since q is odd, we have $y > 0$ and $(y-1) \bmod 2^\tau = (y \bmod 2^\tau) - 1$ and therefore

$$q - 1 - v = 2^k + x \cdot 2^\tau + \underbrace{y - 1 - v}_{S} .$$

Since $0 \leq y - 1 < 2^\tau$, we have $-2^\tau < S < 2^\tau$, and $S \geq 0$ if and only if $y - 1 \geq v$. Hence,

$$\left\lfloor \frac{q-1-v}{2^\tau} \right\rfloor = \begin{cases} 2^{k-\tau} + x & \text{for } S \geq 0 \\ 2^{k-\tau} + x - 1 & \text{for } S < 0 \end{cases} .$$

We have $v = 0$ if and only if $\beta = 0$, and $v = 0$ implies $S \geq 0$. We obtain the result by applying Theorem 3.45.

In case $\tau > k$, we have $0 \leq q-1 < 2^{k+1} \leq 2^\tau$ and therefore $-2^\tau < q-v-1 < 2^\tau$, which implies $\lfloor (q-1-v)/2^\tau \rfloor \in \{-1, 0\}$, i.e., the block corresponding to v occurs either $0 = w$ or $1 = w + 1$ times in u. It is $q - 1 = q - 1 \bmod 2^\tau$ since $\tau > k$, and we have $\lfloor (q-1-v)/2^\tau \rfloor = 0$ if and only if $v \leq q - 1$. Hence, by Theorem 3.45, the block corresponding to v occurs $w + 1 = 1$ times in u if and only if $0 < v \leq q - 1 \bmod 2^\tau$. □

Note that since there are $q-1 \bmod 2^\tau$ elements $\beta \in \mathbb{Z}_{2^\tau}$ fulfilling $0 < -q\beta \leq q - 1 \bmod 2^\tau$, Lemma 3.46 implies similarly to Theorem 3.44 that the number of τ-bit blocks occuring $w + 1$ times in a period of u is $q - 1 \bmod 2^\tau$ and the number of blocks occuring w times is $2^\tau - ((q-1) \bmod 2^\tau)$.

Based on our observations, we can reformulate Eq. (3.11) as

$$\theta_\tau = (B_{00} + B_{11}) - (B_{01} + B_{10}) \text{ with } B_{ij} = w \cdot B_{ij}^w + (w+1) \cdot B_{ij}^{w+1} .$$

Before we derive an explicit formula for θ_τ, we state a preliminary Lemma that is useful for speeding up the computation.

Lemma 3.47. *For a block B of length τ corresponding to $\beta = \sum_{i=0}^{\tau-1} b_i 2^i \in \mathbb{Z}_{2^\tau}$ and $v = -q\beta \bmod 2^\tau$, we have $\beta \equiv v \bmod 2$.*

Proof. We have

$$\begin{aligned} \beta \bmod 2 &= \left(v \cdot (-q)^{-1} \bmod 2^\tau\right) \bmod 2 = v \cdot (-q)^{-1} \bmod 2 \\ &= \begin{cases} 0 & v \equiv 0 \bmod 2 \\ 1 & v \equiv 1 \bmod 2 \end{cases} , \end{aligned}$$

since $q^{-1} \bmod 2^\tau$ is odd if and only if q is odd. □

Hence, in order to compute B_{00}^{w+1}, it suffices to compute the number of blocks corresponding to even $\beta \in \mathbb{Z}_{2^\tau}$ that occur $w + 1$ times, which, by Lemmas 3.46 and 3.47, is equal to the number of $\beta \in \mathbb{Z}_{2^{\tau-1}}$ such that $0 < 2\beta(-q) \bmod 2^{\tau+1} \leq q - 1 \bmod 2^{\tau+1}$, which is equivalent to

$$2\beta q \bmod 2^{\tau+1} \geq -q \bmod 2^{\tau+1} .$$

The blocks corresponding to the remaining β will occur w times. By similar arguments, we obtain $B_{ij}^w + B_{ij}^{w+1} = 2^{\tau-1}$ for all pairs $(i, j) \in \{0, 1\}^2$.

From the complement property of l-sequences we know that $B_{00}^{w+1} = B_{11}^{w+1}$ and $B_{01}^{w+1} = B_{10}^{w+1}$. Moreover, we have $B_{00}^{w+1} + B_{01}^{w+1} + B_{10}^{w+1} + B_{11}^{w+1} = q - 1 \bmod 2^{\tau+1}$ due to Theorem 3.44. Hence,

$$\begin{aligned} B_{01} &= w(2^{\tau-1} - B_{01}^{w+1}) + (w+1)B_{01}^{w+1} \\ &= w2^{\tau-1} + B_{01}^{w+1} \\ &= w2^{\tau-1} + \frac{q - 1 \bmod 2^{\tau+1}}{2} - B_{00}^{w+1} , \end{aligned}$$

and therefore

$$\begin{aligned}\theta_\tau &= (B_{00} + B_{11}) - (B_{01} + B_{10}) \\ &= 2(B_{00} - B_{01}) \\ &= 2\left(wB_{00}^{w} + wB_{00}^{w+1} - w2^{\tau-1} - \frac{q-1 \bmod 2^{\tau+1}}{2} + 2B_{00}^{w+1}\right) \\ &= 2w(B_{00}^{w} + B_{00}^{w+1} - 2^{\tau-1}) - (q-1 \bmod 2^{\tau}) + 4B_{00}^{w+1} \\ &= 4B_{00}^{w+1} - (q-1 \bmod 2^{\tau+1}) \ .\end{aligned}$$

Altogether, we obtain the following result.

Proposition 3.48. *Let* u *denote an l-sequence with connection integer* q*. Then for a given shift* $\tau > 0$ *the autocorrelation* $\theta_\tau(\mathsf{u})$ *is equal to* $4B(\tau, q) - (q - 1 \bmod 2^{\tau+1})$*, where* $B(\tau, q)$ *denotes the number of* $\beta \in \mathbb{Z}_{2^{\tau-1}}$ *such that*

$$2\beta q \bmod 2^{\tau+1} > -q \bmod 2^{\tau+1} \ .$$

The effort required for computing $\theta_\tau(\mathsf{u})$ is dominated by the computation of $B(\tau, q)$. The straightforward approach to test all $\beta \in \{0, \ldots, 2^{r-1} - 1\}$ can be performed by evaluating the function

$$f(\beta) = 2q\beta \bmod 2^{\tau+1} = \begin{cases} f(\beta-1) + 2q \bmod 2^{\tau+1} & \text{for } \beta > 0 \\ 0 & \text{for } \beta = 0 \end{cases}$$

for $\beta \in \{0, \ldots, 2^{r-1}-1\}$, which needs little memory, but $\mathcal{O}\left(\left(2^{\tau-1} - 1\right)\log_2 2^{\tau}\right) = \mathcal{O}\left(\tau \cdot 2^{\tau}\right)$ operations. Hence, our method is only practical for small shifts τ.

Example 3.49. *For* $q = -347$ *and the corresponding l-sequence* u*, we compute for the shift* $\tau = 6$ *the values* $B(\tau, q) = 8$ *and* $q - 1 \bmod 2^{\tau+1} = 26$*, which implies* $\theta_\tau(\mathsf{u}) = 4 \cdot 8 - 26 = 6$*. Similarly for* $\tau = 8$*, we obtain* $B(\tau, q) = 24$ *and* $q - 1 \bmod 2^{\tau+1} = 90$*, hence* $\theta_\tau(u) = 6$*.*

Analogously to the definition of linear complexity (Definition 3.21), Klapper and Goresky (1997) have established the notion of 2-adic span.

Definition 3.50. *The 2-adic span* $\lambda_2(\mathsf{u})$ *of a binary sequence* u *is the size (in terms of number of cells) of the smallest FCSR that generates* u*.*

Lemma 3.51 (Klapper and Goresky (1997)). *For a sequence* u *let* $\alpha = \sum_{i=0}^{\infty} a_i 2^i = \frac{p}{q}$ *be the fraction in lowest terms whose 2-adic expansion agrees with* u*. Then*

$$|(\lambda_2(\mathsf{u}) - 2) - \varphi_2(\mathsf{u})| \leq \log_2(\varphi_2(\mathsf{u})) \ ,$$

where $\varphi_2(\mathsf{u}) = \log_2(max(|p|, |q|))$*.*

Remark 3.52. *Let* u *be an eventually periodic sequence with 2-adic span* λ_2*. Then it is possible to compute integers* p, q *such that the 2-adic expansion of* p/q *is* u *in time* $\mathcal{O}((\lambda_2)^2)$ *while using only the first* $2\lambda_2 + 1$ *bits of* u*. This algorihm resembles the Berlekamp-Massey algorithm for LFSR synthesis and is described by Arnault et al. (2004), Klapper and Xu (2004).*

This suggests to add the 2-adic span to our list of pseudorandomness postulates.

Pseudorandomness Postulate 5. *The 2-adic span* $\lambda_2(\mathsf{u})$ *of a keystream sequence* u *should be reasonably large.*

We conclude that similarly to the LFSR-case, the output of maximum-length FCSRs is not directly suitable as keystream due to its low 2-adic span. However, their otherwise desirable statistical properties recommend both LFSRs and FCSRs, when combined with other devices, as building blocks for stream ciphers.

Chapter 4

Stream Ciphers based on Feedback Shift Registers

4.1 Generic Constructions

We have seen in the previous chapter that both LFSRs and FCSRs may provide sequences with good pseudorandomness properties, but the LFSRs' low linear complexity and the FCSRs' small 2-adic span prevent both devices from being directly used as keystream generators. Nevertheless, many stream ciphers try to benefit from the desirable properties of FSR-sequences and combine one or more FSRs with other components in order to compensate their weaknesses.

We consider in this chapter several generic strategies, namely running FSR sequences through additional Boolean functions before outputting keystream (combination generators and filter generators), adding a small number of memory bits that are updated in a nonlinear way, and state-dependent clocking of the FSRs.

4.1.1 Combination Generators and Filter Generators

A *combination generator* (more precisely, the FSM of a combination generator) consists of a small number of feedback shift registers $R^0, \ldots, R^{k-1}$ and a Boolean function $C : \{0,1\}^k \to \{0,1\}$ that combines the output sequences of the internal registers in order to produce the output keystream (Rueppel (1992), see Fig. 4.1). More precisely, in each clock cycle t, each FSR R^j provides a bit x_t^j and the generator produces a keystream bit $z_t = C(x_t)$, where $x_t = (x_t^0, \ldots, x_t^{k-1})$.

A *filter generator* (again, the FSM of a filter generator to be precise) contains only one feedback shift register R of length n and a Boolean filter function $C : \{0,1\}^n \to \{0,1\}$ that produces the output keystream from the current contents of certain register cells (Rueppel (1992), see Fig. 4.2).

Some combination or filter generators (e.g., the F-FCSR stream cipher family to be discussed in Section 4.2.6) produce more than one output bit per clock cycle, i.e., the keystream function C maps into $\{0,1\}^*$ instead of $\{0,1\}$.

Theorem 3.11 and Corollary 3.41 have shown that for both LFSRs and FCSRs, there exist one-to-one mappings between periodic Fibonacci states and periodic Galois states such that the output sequences produced from these states

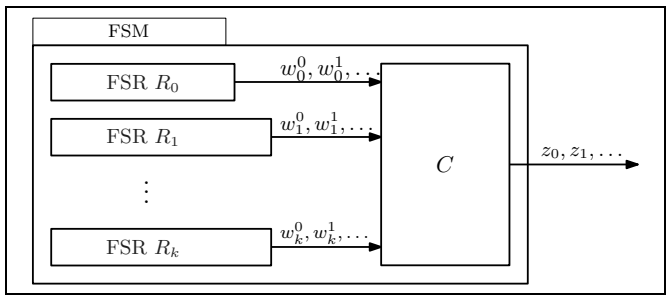

Figure 4.1: FSR-based combination generator

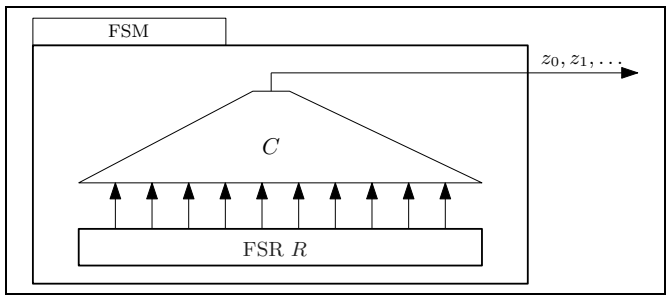

Figure 4.2: FSR-based filter generator

coincide. Moreover, the sequences produced by individual main register cells are again LFSR/FCSR-sequences (Theorem 3.16 and Proposition 3.38).

These observations imply that we can transform a Galois LFSR/FCSR-based filter generator into a Galois LFSR/FCSR-based combination generator that contains as many registers (with appropriate starting states) as the filter function has inputs. Furthermore, Galois registers in the combination generator may be arbitrarily replaced by Fibonacci registers with equivalent starting states. Finally, we may even build an equivalent filter generator based on a Fibonacci LFSR/FCSR (with modified filter) based on Lemmas 3.19 and 3.43. Figure 4.3 summarizes these equivalences.

Note that if all operations of the generator's FSM are linear, its initial state can be efficiently determined from a number of keystream bits by solving a system of linear equations. Therefore, especially in the case that all FSRs are LFSRs, a non-linear function should be chosen as keystream function C.

4.1.2 Additional Memory

In order to improve resistance against correlation attacks and algebraic attacks (to be discussed in Sections 6.1 and 6.2), the keystream generation component of a combination generator is sometimes equipped with a few bits of additional memory, thereby becoming a keystream-FSM. The keystream-FSM takes k bits as input from the FSRs and consists of l memory bits, a keystream function

$$C : \{0,1\}^k \times \{0,1\}^l \to \{0,1\}^* \ ,$$

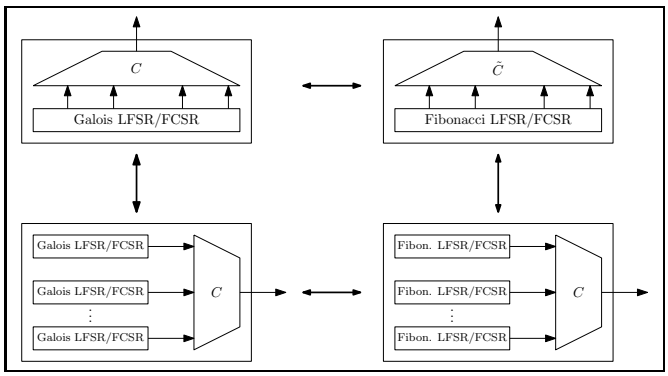

Figure 4.3: Equivalent representations of combination and filter generators

and a memory update function

$$\delta : \{0,1\}^k \times \{0,1\}^l \to \{0,1\}^l \ .$$

In each clock t, it produces from the current input $x_t = (x_t^0, \ldots, x_t^{k-1})$ and the current memory state $q_t = (q_t^0, \ldots, q_t^{l-1})$ the keystream output $C(x_t, q_t)$ and updates the memory to $q_{t+1} := \delta(x_t, q_t)$.

A regularly clocked keystream generator with a keystream-FSM of the described form is commonly called *regularly clocked* (k, l)*-combiner (with memory)*. Observe that in this notation, the memoryless combination generator described in Section 4.1.1 corresponds to a $(k, 0)$-combiner.

LFSR-based combiners with memory were originally introduced by Rueppel (1986). Since then, they have been widely examined in cryptography and have found their way into practical applications. The perhaps best known example used in practice is the E_0 keystream generator, which is in the set of example ciphers that we are going to examine more closely in the remainder of this thesis.

4.1.3 Irregular Clocking

Another way to introduce nonlinearity into a keystream generator is to clock the FSRs in an irregular manner. This is often accomplished by a clock control mechanism which determines based on the current FSM state how often each register's update function is applied before the next keystream bits are produced.

Examples for this desgin include the A5/1 generator (to be described in Section 4.2.3) and the shrinking generator (Coppersmith et al., 1994).

4.2 Example Ciphers

In the ECRYPT stream cipher project eStream (eStream), a number of new ciphers have been proposed and analyzed in the past few years. Many of these recent designs partly replace LFSRs by other feedback shift registers such as nonlinear feedback shift registers (NFSRs) and feedback shift registers with carry (FCSRs) in order to prevent standard cryptanalysis techniques like algebraic attacks and correlation attacks. Moreover, combinations of different types of feedback shift registers permit alternative compression functions.

As examples for these recent proposals, we consider the ciphers TRIVIUM, Grain and the F-FCSR family along with the more aged self-shrinking generator, the E_0 generator, and the A5/1 generator.

4.2.1 Self-Shrinking Generator

The self-shrinking generator was proposed by Meier and Staffelbach (1994) and consists of only one LFSR and no memory. Every two clock cycles of the LFSR, the generator produces a keystream bit according to the function

$$\begin{aligned} \text{shrink} : \{0,1\}^2 &\to \{0,1,\epsilon\} \\ (a,b) &\mapsto \begin{cases} b & \text{if } a = 1 \\ \epsilon & \text{otherwise} \end{cases}, \end{aligned}$$

where ϵ denotes the empty word. For an internal bitstream $w = (w_0, \ldots, w_{2m-1})$, the self-shrinking generator produces the keystream $z = (z_0, \ldots, z_{m-1})$ according to

$$\begin{aligned} \text{shrinkstream} : \{0,1\}^{2m} &\to \{0,1,\epsilon\}^m \\ (w_0, \ldots, w_{2m-1}) &\mapsto (\text{shrink}(w_0, w_1), \ldots, \text{shrink}(w_{2m-2}, w_{2m-1})), \end{aligned}$$

i.e., $z_t = \text{shrink}(w_{2t}, w_{2t+1})$ for $t \in \{0, \ldots, m-1\}$.

The designers proposed a short-keystream attack requiring about $2^{0.75n}$ operations, which was improved to $2^{0.694n}$ by Zenner et al. (2001). The currently best short-keystream attack is a guess-and-determine attack due to Hell and Johansson (2006) requiring around $2^{0.65n}$ operations and an amount of memory that is polynomial in n. The BDD-Attack on the self-shrinking generator, which we will describe in Section 5.5.1, needs roughly as many operations, but exponentially more memory.

The long-keystream attack by Mihaljević (1996) needs at least $2^{0.3n}$ keystream bits in order to compute the initial state in less than $2^{0.6563n}$ polynomial-time operations. Its asymptotic runtime was improved by Hell and Johansson (2006), Zhang and Feng (2006) for the case that up to $2^{0.5n}$ keystream bits are available, while even an improved tradeoff is possible if the weight of the LFSR feedback polynomial is at most 5 (Debraize and Goubin, 2008).

4.2.2 E_0 Generator

The Bluetooth stream cipher has key length 128 bits and IV-length 128 bits. It consists of a key/IV setup procedure and the keystream generator E_0 (The Bluetooth SIG, 2001).

E_0 is a regularly clocked $(k, l) = (4, 4)$ combiner. It consists of four LFSRs $R^0, \ldots, R^3$ of lenghts $(n_0, \ldots, n_3) = (25, 31, 33, 39)$ and a four-bit memory unit. We denote by $x_t = (x_t^0, \ldots, x_t^3) \in \{0,1\}^4$ the bits read from the LFSRs and by $q_t = (q_t^0, \ldots, q_t^3) \in \{0,1\}^4$ the memory state at time t.

The keystream function $g : \{0,1\}^4 \times \{0,1\}^4 \to \{0,1\}$ is defined as

$$g(x_t, q_t) := \bigoplus_{i=0}^{3} x_t^i \bigoplus_{i=0}^{3} c^i q_t^i \ ,$$

where $(c^3, \ldots, c^0) = (0, 1, 0, 0)$.

The memory update function $\delta : \{0,1\}^4 \times \{0,1\}^4 \to \{0,1\}^4$ is given by

$$\delta(x_t, q_t) := (q_{t+1}^3, q_{t+1}^2, q_{t+1}^1, q_{t+1}^0)$$

where

$$\begin{aligned}
(q_{t+1}^1, q_{t+1}^0) &:= (q_t^3, q_t^2) \\
(q_{t+1}^3, q_{t+1}^2) &:= s_t \oplus T_1(q_t^3, q_t^2) \oplus T_2(q_t^1, q_t^0) \\
s_t &:= \lfloor \frac{x_t^3 + x_t^2 + x_t^1 + x_t^0 + 2 \cdot q_t^3 + q_t^2}{2} \rfloor \in \{0,1\}^2 \\
T_1(q_t^3, q_t^2) &:= (q_t^3, q_t^2) \\
T_2(q_t^3, q_t^2) &:= (q_t^2, q_t^3 \oplus q_t^2) \ .
\end{aligned}$$

Figure 4.4 illustrates the design of E_0.

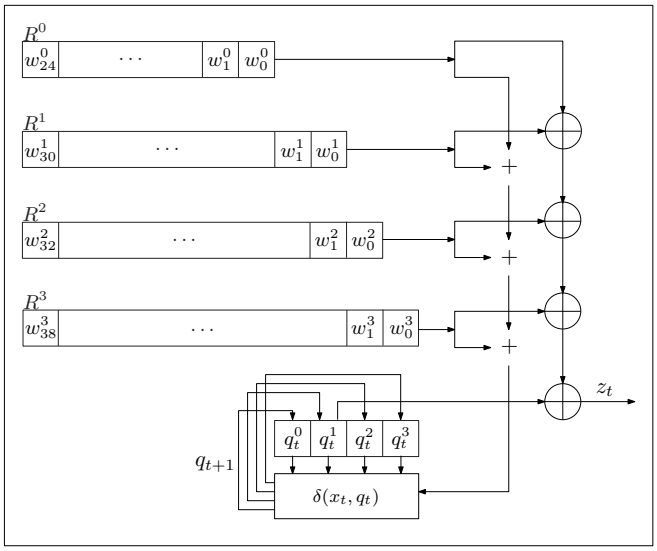

Figure 4.4: The E_0 keystream generator

Note that we may write s_t as

$$s_t = \lfloor \frac{s'_t + 2 \cdot q_t^3 + q_t^2}{2} \rfloor \text{ with } s'_t := \sum_{i=0}^{3} x_t^i \ . \tag{4.1}$$

Hence, the memory update function depends only on the sum s'_t. Similarly, the keystream function g depends only on the value $\bigoplus_{i=0}^{3} x_t^i = s'_t \bmod 2$, which implies

$$g(x_t, q_t) = (s'_t \bmod 2) \oplus \bigoplus_{i=0}^{3} c^i q_t^i \ . \tag{4.2}$$

Since the Bluetooth technology so far is most often applied in wireless voice transmission and data exchange between personal information managers and other mobile devices, confidentiality of the communication is one of the most important security requirements.

Consequently, the security of the Bluetooth encryption has been analyzed in several papers (Armknecht and Krause, 2003, Courtois, 2003, Ekdahl, 2003, Fluhrer and Lucks, 2001, Golić et al., 2002, Hermelin and Nyberg, 1999, Jakobsson and Wetzel, 2001, Krause, 2002, Lu and Vaudenay, 2005, 2004, Saarinen, 2000). Armknecht et al. (2004) showed that an efficient attack on E_0 implies an efficient attack on the whole cipher. Therefore, improving the security of E_0 is a natural demand.

The best currently known long-keystream attacks against E_0 are algebraic attacks (Armknecht and Krause, 2003) and correlation attacks (Lu and Vaudenay, 2004, Lu et al., 2005). However, all these attacks need a large amount of keystream (2^{28} to 2^{39} in the case of correlation attacks), and even in terms of time and memory requirements, the attack by Lu et al. (2005) is the only feasible one among them.

We note that, when applied to the Bluetooth setting, the correlation attacks by Lu and Vaudenay (2004), Lu et al. (2005) depend on the linearity of the key-schedule and other specific properties of the Bluetooth encryption system.

4.2.3 A5/1 Generator

The A5/1 keystream generator is used in the GSM standard for mobile telephones. The initialization procedure transforms a 64-bit secret key and a 22-bit public frame number into the 64-bit initial state of the generator. According to Briceno et al. (1999), who obtained the A5/1 design by reverse engineering, the generator consists of 3 LFSRs R^0, R^1, R^2 of lengths n_0, n_1, n_2, respectively, and a clock control ensuring that the keybits do not linearly depend on the initial states of the LFSRs. For each $r \in \{0,1,2\}$, a register cell q_{N^r}, $N^r \in \{\lceil \frac{n_r}{2} \rceil - 1, \lceil \frac{n_r}{2} \rceil\}$, is selected in LFSR R^r as input for the clock control. The GSM standard uses the parameters $(n_0, n_1, n_2) = (19, 22, 23)$ and $(N^0, N^1, N^2) = (11, 12, 13)$.

Let v_i and v'_i denote the bits at the control and at the output positions in register R^i for $i \in \{0,1,2\}$. In each master clock of the generator, the keystream bit $z_i = f(v'_0, v'_1, v'_2) := v'_0 \oplus v'_1 \oplus v'_2$ is produced, and register R^i is clocked if and only if $v_i = \mathrm{maj}_3(v_0, v_1, v_2)$, where

$$\begin{aligned} \mathrm{maj}_3 : \{0,1\}^3 &\to \{0,1\} \\ (a,b,c) &\mapsto \begin{cases} 1 & \text{if } a+b+c \geq 2 \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

The A5/1 construction is illustrated in Fig. 4.5.

The first short-keystream attack on A5/1 was given by Golić (1997) and needs 2^{42} polynomial time operations. Afterwards, several long-keystream attacks on A5/1 were proposed. Biryukov et al. (2000) present an attack that breaks A5/1 from 2^{15} known keystream bits within minutes after a preprocessing step of 2^{48} operations. Ekdahl and Johansson (2001), Maximov et al. (2005) exploit the linearity of the initialization procedure and manage to break the cipher within minutes, requiring only a few seconds of conversation and little computational resources. A recent effort by a research group around Nohl and Krißler (2010) has received much attention for implementing a distributed, time-memory tradeoff-based brute-force attack that produced a 2-terabyte rainbow

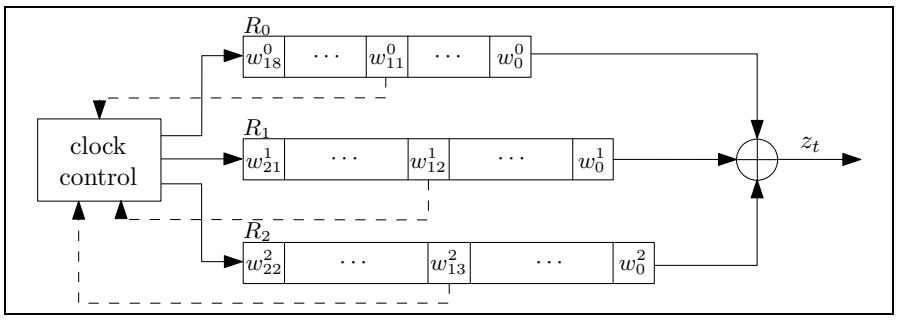

Figure 4.5: The A5/1 keystream generator

table for A5/1, such that the session key of any conversation can be easily derived. A5/1 is supposed to be replaced by A5/3, but only recently, Dunkelman et al. (2010) have published a practical attack on its underlying block cipher.

4.2.4 Trivium

Trivium (de Cannière and Preneel, 2005) is a regularly clocked combination generator that consists of three interconnected NFSRs R^0, R^1, R^2 of lenghts 93,84,111, respectively. The 288-bit initial state of the generator is derived from an 80-bit key and an 80-bit IV by 1152 initialization rounds. The keystream function computes a keystream bit z_t by linearly combining six bits taken from the registers, with each NFSR contributing two bits.

More precisely, from an initial state $(s_1, \ldots, s_{288})$ the algorithm produces keystream bits z_t as follows.

for $t = 0$ to $N - 1$ **do**
$t_1 \leftarrow s_1 \oplus s_{28}$
$t_2 \leftarrow s_{94} \oplus s_{109}$
$t_3 \leftarrow s_{178} \oplus s_{223}$
$z_t \leftarrow t_1 \oplus t_2 \oplus t_3$
$u_1 \leftarrow t_1 \oplus s_2 s_3 \oplus s_{100}$
$u_2 \leftarrow t_2 \oplus s_{95} s_{96} \oplus s_{202}$
$u_3 \leftarrow t_3 \oplus s_{179} s_{180} \oplus s_{25}$
$(s_1, \ldots, s_{93}) \leftarrow (s_2, \ldots, s_{93}, u_3)$
$(s_{94}, \ldots, s_{177}) \leftarrow (s_{95}, \ldots, s_{177}, u_1)$
$(s_{178}, \ldots, s_{288}) \leftarrow (s_{179}, \ldots, s_{288}, u_2)$
end for

Due to its simplicity, especially its low non-linearity, Trivium has received much cryptanalytic attention (see, e.g., Aumasson et al. (2009), Eibach (2008), eSTREAM Discussion Forum (2005), Maximov and Biryukov (2007)). While the best key recovery attack, which is due to Dinur and Shamir (2009), is able to tackle 767 out of 1152 initialization rounds with 2^{45} to 2^{36} operations, the full cipher still remains unbroken.

4.2.5 Grain-128

The regularly clocked combination generator Grain-128 (Hell et al., 2005) supports keys of size 128 bits and IVs of size 96 bits. The design is based on two interconnected FSRs, an LFSR R^0 and an NFSR R^1, both of lenghts 128 bits, and a non-linear keystream function. We denote the content of the LFSR by $(s_t, s_{t+1}, \ldots, s_{t+127})$ and the content of the NFSR by $(b_t, b_{t+1}, \ldots, b_{t+127})$.

In each clock cycle, the registers are updated according to

$$\begin{aligned} s_{t+128} &= s_t \oplus s_{t+7} \oplus s_{t+38} \oplus s_{t+70} \oplus s_{t+81} \oplus s_{t+96} \\ b_{t+128} &= s_t \oplus b_t \oplus b_{t+26} \oplus b_{t+56} \oplus b_{t+91} \oplus b_{t+96} \oplus b_{t+3}b_{t+67} \oplus b_{t+11}b_{t+13} \\ &\quad \oplus b_{t+17}b_{t+18} \oplus b_{t+27}b_{t+59} \oplus b_{t+40}b_{t+48} \oplus b_{t+61}b_{t+65} \oplus b_{t+68}b_{t+84} \ , \end{aligned}$$

and a keystream bit z_t is derived as

$$\begin{aligned} z_t = &\left(\bigoplus_{j \in A} b_{t+j} \right) \oplus b_{t+12}s_{t+8} \oplus s_{t+13}s_{t+20} \oplus b_{t+95}s_{t+42} \\ &\oplus s_{t+60}s_{t+79} \oplus b_{t+12}b_{t+95}s_{t+95} \end{aligned}$$

with $A = \{2, 15, 36, 45, 64, 73, 89\}$.

Besides a generic time-memory-data-tradeoff attack (Biryukov and Shamir, 2000) that recovers the key with time and keystream around 2^{128}, the related-key chosen-IV attack due to Lee et al. (2008) is able to recover the key with $2^{26.59}$ chosen IVs, $2^{31.39}$ keystream bits and 2^{27} operations.

4.2.6 Filtered FCSRs

In order for the initial state not to be recoverable from a number of observed keystream bits by solving a system of linear equations, we have to demand that keystream generators contain nonlinear operations to a certain extent. Since all LFSR-operations are linear by definition, nonlinearity must be introduced into an LFSR-based combination or filter generator by a carefully chosen keystream function C.

FCSRs, on the other hand, have a nonlinear update function, which suggests choosing a simple XOR operation (which is $\mathbb{F}_2$-linear) as keystream function. This has been done in the case of the F-FCSR stream cipher family.

The F-FCSR Stream Cipher Family

F-FCSR-H is an FCSR-based filter generator that consists of a single Galois FCSR of length $n = 160$ with carry cells present at $l = 82$ positions. The connection integer is chosen as

$$q = -199352459131827501532804161134421503646014008796 3 \ ,$$

which implies

$$d = \left(\frac{1-q}{2} \right) = (\text{ae985dff 26619fc5 8623dc8a af46d590 3dd4254e})_{16} \ .$$

At each clock, the generator uses the static filter $d = F$ to extract a pseudorandom byte. The filter splits into 8 subfilters (subfilter j is obtained by selecting the bit j in each byte of F)

$$\begin{aligned} F_0 &= (0011\ 0111\ 0100\ 1010\ 1010)_2, F_4 = (0111\ 0010\ 0010\ 0011\ 1100)_2 \\ F_1 &= (1001\ 1010\ 1101\ 1100\ 0001)_2, F_5 = (1001\ 1100\ 0100\ 1000\ 1010)_2 \\ F_2 &= (1011\ 1011\ 1010\ 1110\ 1111)_2, F_6 = (0011\ 0101\ 0010\ 0110\ 0101)_2 \\ F_3 &= (1111\ 0010\ 0011\ 1000\ 1001)_2, F_7 = (1101\ 0011\ 1011\ 1011\ 0100)_2 \ . \end{aligned}$$

The bit b_i (with $0 \leq i \leq 7$) of each extracted byte is expressed by

$$b_i = \bigoplus_{j=0}^{19} f_i^{(j)} x_{8j+i} \quad \text{where } F_i = \sum_{j=0}^{19} f_i^{(j)} 2^j \ ,$$

and where the x_k are the bits contained in the main register.

The cipher is initialized with an 80-bit key K and an IV of length $0 \leq v \leq 80$ according to Algorithm 1. After the setup phase, the output stream is produced by Algorithm 2.

Algorithm 1 F-FCSR-H-KeyIVSetup(K, IV)

$x := K + 2^{80} \cdot \text{IV} = (0^{80-v}||\text{IV}||K)$
$a := 0 = (0^{82})$
for $i = 0$ to 19 **do**
  Clock the FCSR automaton
  Extract a pseudorandom byte S_i using the filter F
end for
$x := \sum_{i=0}^{19} S_i \cdot 8^i = (S_{15}|| \ldots ||S_0)$
Clock the FCSR automaton 162 times (discard output in this step)

Algorithm 2 F-FCSR-H-KeystreamGeneration

while true do
  Clock the FCSR
  Extract a pseudorandom byte S using the filter F
  Output the value S as keystream byte
end while

F-FCSR-16 works analogously to F-FCSR-H, only with larger parameters. It consists of a Galois FCSR of length $n = 256$ with carry cells present at $l = 130$ positions. The connection integer is chosen as

$$\begin{aligned} q = \ & (183971440845619471129869161809344131658 2 \\ & 9831765592313575301712846215561871501 9)_{10} \ , \end{aligned}$$

which implies

$$\begin{aligned} d = \left(\tfrac{1-q}{2}\right) = \ & (\text{cb5e129f ad4f7e66 780caa2e c8c9cedb} \\ & \text{2102f996 baf08f39 efb55a6e 390002c6})_{16} \ . \end{aligned}$$

To extract two pseudorandom bytes, the static filter $F = d$ is used. The filter F is split into 16 subfilters (subfilter j is obtained by selecting the bit j in each 16-bit word of F)

$$\begin{aligned}
F_0 &= (0110\ 0011\ 0001\ 1000)_2, F_8 = (1010\ 0000\ 1101\ 1010)_2 \\
F_1 &= (1111\ 0101\ 1100\ 0101)_2, F_9 = (1101\ 0101\ 0011\ 1101)_2 \\
F_2 &= (1111\ 1100\ 0100\ 1101)_2, F_{10} = (0011\ 0001\ 0001\ 1000)_2 \\
F_3 &= (1110\ 1111\ 0001\ 0100)_2, F_{11} = (1011\ 1111\ 0111\ 1110)_2 \\
F_4 &= (1100\ 0001\ 0111\ 1000)_2, F_{12} = (0101\ 1000\ 0110\ 0110)_2 \\
F_5 &= (0001\ 0100\ 0011\ 1100)_2, F_{13} = (0011\ 1100\ 1110\ 1010)_2 \\
F_6 &= (1011\ 0011\ 0010\ 0101)_2, F_{14} = (1001\ 1011\ 0100\ 1100)_2 \\
F_7 &= (0100\ 0011\ 0110\ 1001)_2, F_{15} = (1010\ 0111\ 0111\ 1000)_2
\end{aligned}$$

The bit b_i (with $0 \leq i \leq 15$) of each extracted word is expressed by

$$b_i = \bigoplus_{j=0}^{15} f_i^{(j)} x_{16j+i} \quad \text{where } F_i = \sum_{j=0}^{15} f_i^{(j)} 2^j \ ,$$

and where the x_k are the bits contained in the main register.

The cipher is initialized with a 128-bit key K and an IV of length $0 \leq v \leq 128$ according to Algorithm 3. After the setup phase, the output stream is produced by Algorithm 4.

Algorithm 3 F-FCSR-16-KeyIVSetup(K, IV)

$x := K + 2^{128} \cdot \text{IV} = (0^{128-v}||\text{IV}||K)$
$a := 0 = (0^{130})$
for $i = 0$ to 15 **do**
  Clock the FCSR automaton
  Extract a pseudorandom word S_i using the filter F.
end for
$x := \sum_{i=0}^{15} S_i \cdot 2^{16i} = (S_{15}|| \ldots ||S_0)$
$a := 0 = (0^{130})$
Clock the FCSR automaton 258 times (discard output in this step)

Algorithm 4 F-FCSR-16-KeystreamGeneration

while true do
  Clock the FCSR
  $S = x \wedge F$
  Split S into 16 words of length 16 bits each, such that $S = \sum_{i=0}^{15} S_i 2^{16i}$
  Output the value $\bigoplus_{i=0}^{15} S_i$ as keystream word
end while

Security Considerations

Using FCSRs as building blocks for stream ciphers had initially been suggested by Klapper and Goresky (1997, 1994). A few years later Arnault and Berger

(2005a) revisited the idea by proposing and analyzing a generic filter generator based on a Galois FCSR and the XOR operation as keystream function. Several concrete instantiations of this idea were proposed (Arnault and Berger, 2005b) and improved in the light of cryptanalysis results (Jaulmes and Muller, 2006, 2005), before the the two ciphers F-FCSR-H and F-FCSR-16 in the form described above were specified by Arnault et al. (2006).

In the absence of any apparent weaknesses in these versions, the ECRYPT stream cipher project eStream suggested F-FCSR-H and F-FCSR-16 for practical applications (Babbage et al., 2008).

Various analyses suggest that the produced keystream has good pseudorandomness properties (Arnault and Berger, 2005a, Arnault et al., 2008). The filter function computes the binary XOR of its inputs, and its initialization procedure ensures that the initial state of the generator is periodic. Hence, by Proposition 3.38, the keystream generation procedure is equivalent to taking the bitwise XOR-sum of different parts of the same l-sequence, while the starting position is given by the initial state and the distances between the parts are constant. This design was motivated by the conjecture that linear and 2-adic operations are unrelated and that the correlation between two distant parts of the same l-sequence is low (Arnault and Berger, 2005a). Our explicit computation of the distances for F-FCSR-H based on Proposition 3.38 shows that the parts of the sequence are indeed almost evenly distributed over the period (see Fischer et al. (2008), Appendix A).

However, while Arnault et al. (2008) had shown that the cells of the carry and the main register will not be zero for several consecutive clock cycles, Hell and Johansson (2008, 2009) observed that the sequence of main register states $(x_t)_{t\geq 0}$ that are passed during the operation of the cipher are likely to contain sufficiently long runs of the form $(0, \ldots, 0, 1, 0)$, which turns the state update function into a linear function and allows for setting up a system of linear equations in order to recover the register state. With a few optimizations, Hell and Johansson show how to recover the state of the register in F-FCSR-H from around $2^{23.7}$ bytes of keystream in 10 seconds on average with standard PC hardware. The same idea yields efficient attacks on F-FCSR-16 and X-FCSR (Stankovski et al., 2009), a software-oriented stream cipher based on FCSRs (Arnault et al., 2007). Hence, it turns out that the update function of FCSRs does not introduce as much nonlinearity as originally expected. In the light of these attacks, F-FCSR-H and F-FCSR-16 were removed from the eStream list of recommended ciphers.

Finally, we want to note that replacing the Galois FCSR in an F-FCSR-H-like construction by a Fibonacci FCSR while keeping the XOR filter function yields an insecure keystream generator. This can be seen as follows.

Consider the F-FCSR-H parameters, i.e., $n = 160$, $l = 82$ and with $k = 8$ linear filters, but applied to a Fibonacci FCSR. Initially, there are 160 binary variables (ignoring the memory), and each updated bit is represented by a new variable (ignoring the details of the construction and assuming independence). Each iteration gives another 8 linear equations in these (initial and newly introduced) state variables. The main register can be recovered by solving the system of linear equations if the number of equations is at least as large as the number of variables. This requires r iterations, where $8r \geq 160 + r$. Consequently, $r = 23$ iterations are sufficient, or 184 bits of keystream. Gaussian elimination of this system requires a computational effort of about 184^3, which is about 2^{23}.

After recovering the main register, one can recover the contents of the memory cells. If the FCSR is in a periodic state (which can be expected already after the initialization phase), then the effective size of the memory reduces to 7 bits. Consequently, the memory can be guessed or recovered by FCSR-synthesis, and the whole state can be recovered in about 2^{30} steps and with less than 200 bits of keystream. A similar attack is possible for any other construction of this type with $k > 1$.

4.3 Abstraction: Internal Bitstream Generators

The keystream generators that we analyze in this thesis are FSR-based in the sense that their internal state is distributed over a small number of feedback shift registers $R^0, \ldots, R^{k-1}$ that provide input for the keystream function C.

For our subsequent analysis, it is convenient to think of these FSRs as a single entity, the internal bitstream generator, that produces an internal bitstream $(w_t)_{t\geq 0}$ defined by

$$w_t := w_{s(t)}^{r(t)} \text{ with } r(t) = t \bmod k \text{ and } s(t) = s \operatorname{div} k \ ,$$

i.e., the t-th internal bit corresponds to the $s(t)$-th bit in the bitstream produced by $R^{r(t)}$ (see Fig. 4.6). Again, the internal bitstream (and hence the

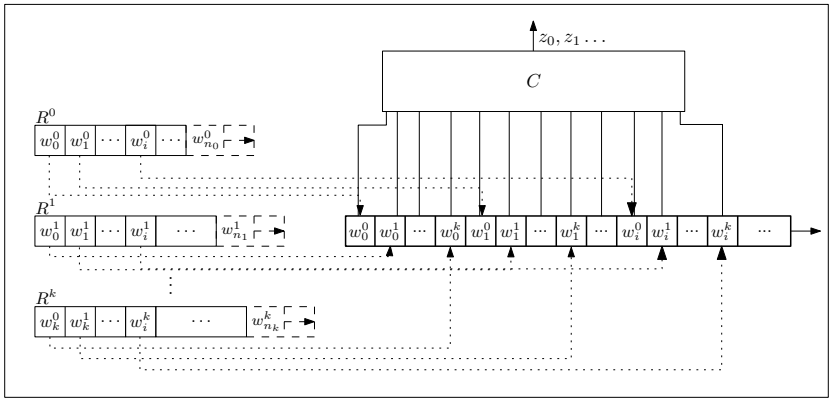

Figure 4.6: Derivation of the keystream from the internal bitstream

output of the keystream generator) are entirely determined by the generator's starting state $\omega(0)$, and the first m bits can be computed as $(w_0, \ldots, w_{m-1}) = H_{\leq m}(\omega(0))$, where

$$H_{\leq m} : \{0,1\}^n \to \{0,1\}^m \ .$$

Definition 4.1. *We call an integer i an* initial position *in w, if w_i corresponds to a bit from the initial state of some FSR, and a* combined position *otherwise. Correspondingly, we denote by* $\mathrm{IP}(i)$ *the set of initial positions and by* $\mathrm{CP}(i)$ *the set of combined positions in $\{0, \ldots, i-1\}$. We let* $\mathrm{IB}(w)$ *denote the bits at the initial positions in w, $n_{\min}$ the maximum i for which all $i' \leq i$ are initial positions, and $n_{\max}$ the minimum i for which all $i' > i$ are combined positions.*

In an FSR-based bitstream generator, the FSRs may be interconnected in the sense that the update function F^i of R^i may also depend on the current content of the other registers such that $F^i : \{0,1\}^{n_i} \to \{0,1\}$, $n_i \leq n$, for all $i \in \{0,\ldots,k-1\}$.

The keystream function $C : \{0,1\}^n \to \{0,1\}^*$, which derives keystream bits from the current state, usually depends on one or more state bits from each FSR. For $|w| = m$ we denote the keystream prefix that is produced from w by $C_m(w)$, where $C_m : \{0,1\}^m \to \{0,1\}^*$.

Generally, we call a keystream generator regularly clocked, if for all $j \in \{0,\ldots,k-1\}$, the register R^j is clocked equally often in each clocking of the whole generator. This definition translates into our notion of FSR-based internal bitstream generators as follows.

Definition 4.2. *Let $D(w,t) := \{w_i | z_t$ depends on $w_i\}$. We call an FSR-based keystream generator* regularly clocked *if $|D(w,t) \backslash D(w,t')|$ is constant for all internal bitstreams w and all $0 \leq t' < t$.*

Note that this definition corresponds to the notion of an *oblivious* keystream generator that was established by Krause (2007).

Two important parameters of FSR-based keystream generators are the *best-case compression ratio* and the *information rate*, which we define as follows.

Definition 4.3. *If γm is the maximum number of keybits that the generator produces from internal bitstreams of length m, we call $\gamma \in (0,1]$ the* best-case compression ratio *of the generator. Moreover, for a randomly chosen and uniformly distributed internal bitstream $W^{(m)} \in \{0,1\}^m$ and a random keystream Z, we define as* information rate *α the average information that Z reveals about $W^{(m)}$, i.e., $\alpha := \frac{1}{m} I\left(W^{(m)}, Z\right) \in (0,1]$.*[1]

For a randomly chosen and uniformly distributed internal bitstream $w \in \{0,1\}^m$, the probability of the keybits' $C_m(w)$ being a prefix of a given keystream $z \in \{0,1\}^*$ can be expressed as

$$\begin{aligned} &\Pr_{w \in \{0,1\}^m}[C_m(w) \text{ is prefix of } z] = \\ &\sum_{i=0}^{\lceil \gamma m \rceil} \Pr_{w \in \{0,1\}^m}[|C_m(w)| = i] \cdot \Pr_{\substack{w \in \{0,1\}^m \\ |C_m(w)| = i}}[C_m(w) = (z_0,\ldots,z_{i-1})] \ . \end{aligned} \tag{4.3}$$

Concerning this probability, we make the following assumption.

Assumption 4.4 (Independence Assumption). *For all $m \geq 1$, a randomly chosen, uniformly distributed internal bitstream $w \in \{0,1\}^m$, and all keystreams $z \in \{0,1\}^*$, we have $\Pr_w[C_m(w)$ is prefix of $z] = p_C(m)$, i.e., the probability of $C_m(w)$ being a prefix of z is independent of z.*

As shown by Krause (2002), the computation of α can be simplified as follows if the generator fulfills the Independence Assumption.

Lemma 4.5. *If a keystream generator satisfies the Independence Assumption, we have $\alpha = -\frac{1}{m}\log_2(p_C(m))$.*

[1] Recall that for two random variables A and B, the value $I(A,B) = H(A) - H(A|B)$ defines the information that B reveals about A.

Proof. The definitions of information and entropy imply

$$\alpha = \frac{1}{m} I\left(W^{(m)}, Z\right) = \frac{1}{m}\left(H\left(W^{(m)}\right) - H\left(W^{(m)}|Z\right)\right) = \frac{1}{m}\left(m - H\left(W^{(m)}|Z\right)\right)$$

and

$$H\left(W^{(m)}|Z\right) =$$

$$\sum_{z\in\{0,1\}^*} \Pr\left[Z=z\right]\left(-\sum_{w\in\{0,1\}^m} \Pr\left[W^{(m)}=w|Z=z\right]\cdot\log_2\Pr\left[W^{(m)}=w|Z=z\right]\right) \quad .$$

Under the Independence Assumption (Assumption 4.4), all $w \in \{0,1\}^m$ and $z \in \{0,1\}^*$ satisfy

$$\Pr[W^{(m)}=w|Z=z] = \begin{cases} \frac{1}{p_C(m)\cdot 2^m} & \text{if } C(w) \text{ is prefix of } z \\ 0 & \text{otherwise} \end{cases} \quad .$$

With $\tilde{W} := \{w \in \{0,1\}^m | C_m(w) \text{ is prefix of } z\}$, we obatin

$$H\left(W^{(m)}|Z\right) = \sum_{z\in\{0,1\}^*} \Pr\left[Z=z\right] \underbrace{\left(-\sum_{w\in\tilde{W}} (p_C(m)2^m)^{-1}\cdot\log_2((p_C(m)2^m)^{-1})\right)}_{\log_2(p_C(m)2^m)}$$

$$= \log_2(p_C(m)2^m) \ ,$$

and finally

$$\alpha = -\frac{1}{m}(m - \log_2(p_C(m)2^m)) = \frac{1}{m}(m - \log_2 p_C(m) - m) = -\frac{1}{m}\log_2 p_C(m) \ \square$$

Corollary 4.6. *The information rate α of a regularly clocked FSR-based keystream generator fulfilling the Independence Assumption is given by $\alpha = \frac{\beta(m)}{m}$.*

Proof. The Independence Assumption and Definition 4.2 imply that the $2^{\beta(m)}$ possible keystream blocks of length $\beta(m)$ that can be produced from the m-bit internal bitstream all have probability $p_C(m)$. Hence $p_C(m) = 2^{-\beta(m)}$ and therefore $\alpha = -\frac{1}{m}\log_2(2^{-\beta(m)}) = \frac{\beta(m)}{m}$. $\square$

Observation 4.7. *For a regularly clocked FSR-based keystream generator with k FSRs that uses exactly one bit from each register for computing a keystream bit z_t, we have $\alpha = \frac{1}{k}$.*

Finally, we assume the internal bitstream to behave pseudorandomly, which we formalize as follows.

Assumption 4.8 (Pseudorandomness Assumption). *For $m \leq \lceil \alpha^{-1} n \rceil$, let w and $\omega(0)$ denote randomly chosen, uniformly distributed elements of $\{0,1\}^m$ and $\{0,1\}^{|\mathrm{IP}(m)|}$, respectively. Then, all keystreams z satisfy*
$\Pr_w[C_m(w) \text{ is prefix of } z] \approx \Pr_{\omega(0)}[C_m(H_{\leq m}(\omega(0))) \text{ is prefix of } z]$.

We expect the Pseudorandomness Assumption to hold since a significant violation would imply the vulnerability of the generator to a correlation attack.

Chapter 5

The BDD-Attack

5.1 Introduction and Overview

Krause (2002, 2007) proposed a Binary Decision Diagram (BDD) attack on LFSR-based combination generators. The BDD-attack is a generic attack in the sense that it does not depend on specific design properties of the respective cipher. It only relies on the assumptions that the generator's internal bitstream behaves pseudorandomly and that the test whether a given internal bitstream w produces a sample keystream can be represented in a Free Binary Decision Diagram (FBDD) of size polynomial in the length of w.

The attack reconstructs the secret initial state from the shortest information-theoretically possible prefix of the keystream (usually a small multiple of the state size), whereas other generic attack techniques in many cases require amounts of known keystream that are unlikely to be available in practice. Particularly in the case of E_0 and A5/1, the first keystream frame already suffices to obtain all the information that is needed to compute the initial state.

As an extension of the original attack by Krause (2002), we show that the BDD-based approach remains applicable in the presence of (possibly interdependent) NFSRs and FCSRs combined with arbitrary keystream functions, as long as not too many new internal bits are produced in each clock cycle of the cipher. Consequently, we apply the attack to the NFSR-based proposals Trivium, Grain, and the F-FCSR family, which were described in Section 4.2. In order to avoid redundancies, we directly outline this more general technique and treat the original attack by Krause as a special case.

One drawback of the BDD-attack is its high memory consumption. We approach this problem by presenting various efficiently parallelizable divide-and-conquer strategies (DCS) for E_0 and A5/1 that substantially reduce the memory requirements and allow us to tackle much larger key lengths with fixed computational resources. In the case of E_0, our DCS lowers the attack's memory requirements by a factor of 2^{25} and additionally yields a slight improvement of the theoretical runtime.

Finally, we present comprehensive experimental results for the BDD-attack on reduced versions of the E_0, A5/1 and the self-shrinking generator, which show that the attack performance in practice does not seem to substantially deviate from the theoretical figures.

5.2 Representing Boolean Functions with Binary Decision Diagrams

Boolean functions can be represented in many ways, e.g., in truth tables or symbolically as a formula in algebraic normal form (ANF). For our attack, yet another representation will turn out to be particularly useful, namely the graph-based representation in a Binary Decision Diagram (BDD).

BDDs and their variants have received much attention since the publication of the fundamental paper by Bryant (1986). We briefly review the definition of BDDs and their most important algorithmic properties and kindly refer the reader to Wegener (2000) for a more comprehensive overview.

Definition 5.1. *A* Binary Decision Diagram *(BDD) G over a set of variables $X_n = \{x_1, \ldots, x_n\}$ is a directed, acyclic graph $G = (V, E)$ with $E \subseteq V \times V \times \{0,1\}$. Each inner node v has exactly two outgoing edges, a* 0*-edge $(v, v_0, 0)$ and a* 1*-edge $(v, v_1, 1)$ leading to the* 0*-successor v_0 and the* 1*-successor v_1, respectively. G contains exactly two nodes with outdegree* 0*, the sinks s_0 and s_1. Each inner node v is assigned a label $v.label \in x_n$, whereas the two sinks are labeled $s_0.label = 0$ and $s_1.label = 1$. There is exacly one node with indegree* 0*, the root of G. We define the size of G (denoted by $|G|$) to be the number of nodes it contains, i.e., $|G| := |V|$. Each node $v \in V$ represents a Boolean Function $f_v \in B_n = \{f | f : \{0,1\}^n \to \{0,1\}\}$ in the following manner. For an input $a = (a_1, \ldots, a_n) \in \{0,1\}^n$, the computation of $f_v(a)$ starts in v. In a node with label x_i, the outgoing edge with label a_i is chosen, until one of the sinks is reached. The value $f_v(a)$ is then given by the label of this sink.*

Definition 5.2. *For a BDD G over x_n, let $G^{-1}(1) \subseteq \{0,1\}^n$ denote the set of inputs accepted by G, i.e., all inputs $a \in \{0,1\}^n$ such that $f_{\text{root}}(a) = 1$.*

Note that we may delete all $v \in V$ in G that are not reachable from the root without changing the function f_{root} that G computes.

We can straightforwardly use BDDs as a data structure for subsets of $\{0,1\}^n$. In order to represent $S \subseteq \{0,1\}^n$, we construct a BDD G_S that computes the characteristic function f_S of S given by $f_S(x) = 1$ if $x \in S$ and $f_S(x) = 0$ otherwise. Hence, G_S will accept exactly the elements of S. Moreover, we can compute a BDD representing the intersection $S \cap T$ of two sets S and T from their BDD-representations G_S and G_T by an AND-synthesis of G_S and G_T.

Remark 5.3. *Since general BDDs have many degrees of freedom for representing a particular Boolean function, many important operations and especially those that are needed in our context are* NP*-hard (cf. Wegener (2000) for details).*

We therefore concentrate on the more restricted models of Free Binary Decision Diagrams (FBDDs) and Ordered Binary Decision Diagrams (OBDDs).

5.2.1 Free Binary Decision Diagrams (FBDDs)

Definition 5.4. *An* oracle graph *$G^0 = (V, E)$ over a set of variables $X_n = \{x_1, \ldots, x_n\}$ is a modified BDD that contains only one sink s, labeled $*$, and for all $x_i \in X_n$ and all paths P from the root in G to the sink, there exists at most one node in P that is labeled x_i.*

Definition 5.5. *A* Free Binary Decision Diagram G *with respect to an oracle graph* G^0 *(abbreviated by* G^0*-FBDD) over a set of variables* $X_n = \{x_1, \ldots, x_n\}$ *is a BDD in which all inputs* $a \in \{0,1\}^n$ *satisfy the following condition. Let the list* $G^0(a)$ *contain the variables from* X_n *in the order in which they occur on the path defined by* a *in* G^0*. Similarly, let the list* $G(a)$ *contain the variables from* X_n *in the order in which the components of* a *are read in* G*. If* x_i *and* x_j *are both contained in* $G(a)$*, then they occur in* $G(a)$ *in the same order as in* $G^0(a)$*.*

We call a BDD G *an FBDD, if there exists an oracle graph* G^0 *such that* G *is a* G^0*-FBDD.*

Figure 5.1 shows examples for an oracle graph G^0 and a G^0-FBDD.

The definition of FBDDs implies their important *read-once property*, i.e., on each path in an FBDD G, each variable in X_n is tested at most once.

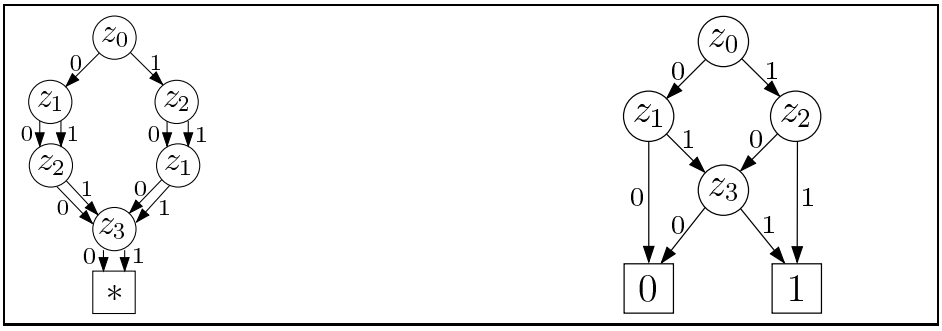

Figure 5.1: An oracle graph G^0 over $\{z_0, \ldots, z_3\}$ and a G^0-FBDD

FBDDs possess several algorithmic properties that will prove useful in our context. Let G^0 denote an oracle graph over $X_n = \{x_1, \ldots, x_n\}$ and let the G^0-FBDDs G_f, G_g and G_h represent Boolean functions $f, g, h : \{0,1\}^n \to \{0,1\}$.

FBDD Property 1. *There exists an algorithm MIN that computes for* G_f *in time* $\mathcal{O}(|G_f|)$ *the (uniquely determined) minimal* G^0*-FBDD* G *that represents* f*. Every minimal* G^0*-FBDD* G *over* X_n *satisfies* $|G| \leq n \cdot |G_f^{-1}(1)|$*.*

FBDD Property 2. *There exists an algorithm SYNTH that computes for* G_f*,* G_g *and* G_h *in time* $\mathcal{O}(|G^0| \cdot |G_f| \cdot |G_g| \cdot |G_h|)$ *a* G^0*-FBDD* G *of size* $|G| \leq |G^0| \cdot |G_f| \cdot |G_g| \cdot |G_h|$ *which represents the function* $f \wedge g \wedge h$*.*

FBDD Property 3. *There exists an algorithm SAT-ENUM that enumerates for a* G^0*-FBDD* G_f *all elements in* $G_f^{-1}(1)$ *in time* $\mathcal{O}(n \cdot |G_f^{-1}(1)|)$*.*

Definition 5.6. *We call an algorithm* A *over the input space* $\{0,1\}^n$ read-once algorithm*, if it reads each input bit at most once.*

Definition 5.7. *Fix a read-once algorithm* A *over* $\{0,1\}^n$*, an input* $x \in \{0,1\}^n$ *and an oracle graph* G^0 *over* X_n*. Let the list* $\pi(A,x)$ *contain the variables from* X_n *in the order in which they are read by* A *when procesing* x*, and let the list* $\pi(G^0,x)$ *contain the variables from* X_n *in the order in which they occur on the path defined by* x *in* G^0*. We say that the read-once algorithm* A respects the oracle graph G^0 *(*A *is* G^0*-respecting) if for all inputs* $x \in \{0,1\}^n$*, any two variables* x_i, x_j *from* $\pi(A,x)$ *occur in the same order in* $\pi(G^0,x)$ *as in* $\pi(A,x)$*.*

Read-once algorithms correspond to *Eraser Turing Machines* (Eraser-TMs), which only differ from general Turing machines in the property that each input bit is deleted immediately after being read (Ajtai et al., 1986, Krause et al., 1988). We consider Eraser-TMs that have an associated oracle graph which determines the reading order of the input bits. During the computation, the machine follows the path that the input defines in the oracle graph in order to determine the next bit to read.

The following observation links read-once algorithms to FBDDs and is an immediate consequence of the observations by Meinel (1989).

Observation 5.8. *Fix a subset $F \subseteq \{0,1\}^n$ and an oracle graph G^0 over $X_n = \{x_1, \ldots, x_n\}$. Each G^0-respecting read-once algorithm A that decides for an input $x = (x_1, \ldots, x_n) \in \{0,1\}^n$ whether $x \in F$ while using at most p bits of additional memory can be efficiently transformed into a G^0-FBDD of size at most $|G^0| \cdot 2^p$.*

Proof. Consider the Eraser-TM with p memory cells that corresponds to A. A configuration of A is given by the tuple $(v_i, y_1, \ldots, y_p)$, where v_i denotes the current vertex in G^0 and $(y_1, \ldots, y_p)$ represents the current content of the additional memory cells.

We transform A into a G^0-FBDD G_A as follows. The vertices in G_A have the form $[v_i, y] \in V(G^0) \times \{0,1\}^p$, and a vertex $[v_i, y]$ is labled with v_i.label. If (v_{i_0}, y_0) denotes the initial configuration of A, we define as root of G_A the vertex $[v_{i_0}, y_0]$. For each transition $\delta(v_i, y, x_{v_i.\text{label}}) = (v_i', y')$ of A, we add to G_A a directed $(x_{v_i.\text{label}})$-edge from vertex $[v_i, y]$ to vertex $[v_i', y']$. For a stop configuration $\delta(v_i, y, x_{v_i.\text{label}}) = (v_i, y)$ of A with output $b \in \{0,1\}$, we add a directed $(x_{v_i.\text{label}})$-edge from vertex $[v_i, y]$ to the b-sink.

We observe that since A is G^0-respecting, the reading order on each path from the root to a sink in G_A is consistent with G^0, which makes G_A a G^0-FBDD.

For a fixed $v_i \in V(G^0)$, A has at most 2^p configurations (v_i, y). Therefore, the maximum size of G_A is $|G^0| \cdot 2^p$. □

Many important Boolean functions can even be efficiently represented in a more restricted BDD variant, so-called Ordered Binary Decision Diagrams (OBDDs), which we are going to describe next.

5.2.2 Ordered Binary Decision Diagrams (OBDDs)

Ordered Binary Decision Diagrams were first described by Bryant (1986) and have become an important tool for circuit verification, VLSI-design and many other applications.

Definition 5.9. *A* variable ordering π *for a set of variables $X_n = \{x_1, \ldots, x_n\}$ is a permutation of the index set $I = \{1, \ldots, n\}$, where $\pi(i)$ denotes the position of x_i in the π-ordered variable list $x_{\pi^{-1}(1)}, x_{\pi^{-1}(2)}, \ldots, x_{\pi^{-1}(n)}$.*

Definition 5.10. *A* π-Ordered Binary Decision Diagram (π-OBDD) *with respect to a variable ordering π is a BDD in which the sequence of tests on a path from the root to a sink is restricted by π, i.e., whenever an edge leads from an x_i-node to an x_j-node, then $\pi(i) < \pi(j)$. A BDD G is called OBDD, if there exists a variable ordering π such that G is a π-OBDD.*

For an OBDD G we define its width as

$$w(G) := \max_i \{|\{v \in G | v.\text{label} = x_i\}|\} \ .$$

Note that we may view any π-OBDD as a degenerated G^0-FBDD in which the reading order on each path from the root to one of the sinks is consistent with π, i.e., G^0 is degenerated into a linear list that corresponds to π.

Conversely, a π-OBDD may at the same time be a G^0-FBDD in the following sense.

Definition 5.11. *A variable ordering π for X_n is said to be* consistent with an oracle graph G^0 *over X_n if for any $(i,j) \in \{1,\ldots,n\}^2$ with $\pi(i) < \pi(j)$, x_i occurs before x_j on all paths in G^0.*

Observation 5.12. *If a variable ordering π is consistent with an oracle graph G^0, then any π-OBDD is a G^0-FBDD.*

Figure 5.2 shows a π-OBDD that computes the function

$$f(z_0,\ldots,z_3) = z_0 z_2 \vee z_0 \bar{z_2} z_3 \vee \bar{z_0} z_1 z_3 \ .$$

Similarly to FBDDs, OBDDs allow for efficient implementations of the operations that we will be interested in. Let π denote a variable ordering for $X_n = \{x_1,\ldots,x_n\}$ and let the π-OBDDs G_f, G_g and G_h represent Boolean functions $f,g,h : \{0,1\}^n \to \{0,1\}$.

OBDD Property 1. *The size of G_f is bounded by $|G_f| \leq m \cdot w(G_f)$.*

OBDD Property 2. *There exists an algorithm MIN that computes in time $\mathcal{O}(|G_f|)$ the uniquely determined minimal π-OBDD G with $w(G) \leq |G_f^{-1}(1)|$ that represents f.*

OBDD Property 3. *There exists an algorithm SYNTH that computes in time $\mathcal{O}(|G_f| \cdot |G_g| \cdot |G_h|)$ a minimal π-OBDD G with $w(G) \leq w(G_f) \cdot w(G_g) \cdot w(G_h)$.*

OBDD Property 4. *There exists an algorithm SAT-ENUM that enumerates all elements of $G_f^{-1}(1)$ in time $O\left(n \cdot |G_f^{-1}(1)|\right)$.*

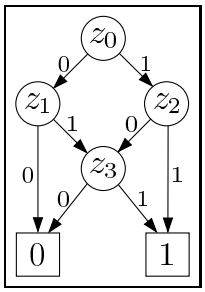

Figure 5.2: A π-OBDD over $\{z_0,\ldots,z_3\}$ with $\pi(0) = 0$, $\pi(1) = 2$, $\pi(2) = 1$ and $\pi(3) = 3$.

Definition 5.13. *We say that a read-once algorithm A* respects a variable ordering π *(A is* π-respecting*) if A does not read x_i after x_j in case $\pi(i) < \pi(j)$ for all $i, j \in \{1, \ldots .n\}$.*

Similarly to the correspondence of read-once algorithms and FBDDs described by Observation 5.8, we can transform a π-respecting read-once algorithm into a π-OBDD.

Observation 5.14. *Fix a subset $F \subseteq \{0,1\}^n$ and a variable ordering π for $X_n = \{x_1, \ldots, x_n\}$. Each π-respecting read-once algorithm A that decides for an input $x = (x_1, \ldots, x_n) \in \{0,1\}^n$ whether $x \in F$ while using at most p bits of additional memory can be efficiently transformed into a π-OBDD of width at most 2^p.*

Proof. The proof is largely analogous to the proof of Observation 5.8, but we include it for completeness.

Consider the eraser Turing machine with p memory cells that corresponds to A. We can assume w.l.o.g. that the input is π-ordered, i.e., it is given as $x_{\pi(1)}, x_{\pi(2)}, \ldots, x_{\pi(n)}$. A configuration of A is given by the tuple $(i, y_1, \ldots, y_p)$, where i denotes the current read position in the input and $(y_1, \ldots, y_p)$ represents the content of the p additional memory cells.

We transform A into a π-OBDD G_A as follows. The vertices in G_A have the form $[i, y] \in \{1, \ldots, n\} \times \{0,1\}^p$, and a vertex $[i, y]$ is labled with x_i. If (i_0, y_0) denotes the initial configuration of A, we define as root of G_A the vertex $[i_0, y_0]$. For each transition $\delta(i, y, x_i) = (i', y')$ of A, we add to G_A a directed x_i-edge from vertex $[i, y]$ to vertex $[i', y']$. For a stop configuration $\delta(i, y, x_i) = (i, y)$ of A with output $b \in \{0,1\}$, we add a directed x_i-edge from vertex $[i, y]$ to the b-sink.

We observe that since A is π-respecting, the reading order on each path from the root to a sink in G_A is consistent with π, which makes G_A a π-OBDD.

For a fixed $i \in \{1, \ldots, n\}$, A has at most 2^p configurations (i, y). Therefore, the maximum width of G_A is 2^p. □

5.3 BDD-based Initial State Recovery

The BDD-based attack on keystream generators is a known-plaintext initial state recovery attack, i.e., the attacker tries to reconstruct the unknown initial state $\omega(0)$ of the keystream generator from a few known plaintext bits $p_0, p_1, \ldots$ and their encryptions $c_1, c_2, \ldots$. In our scenario in which a ciphertext bit c_i is computed from a plaintext bit p_i and a keystream bit z_i via $c_i = p_i \oplus z_i$, the keystream bit z_i can be reconstructed from (p_i, c_i) by computing $p_i \oplus c_i = z_i$.

We first observe that for any internal bitstream $w \in \{0,1\}^m$ that yields a prefix of the observed keystream, the following two conditions must hold.

Condition 1. *w is an m-extension of the initial state bits in w, i.e., we have $H_{\leq m}(\mathrm{IB}(w)) = w$.*

Condition 2. *$C_m(w)$ is a prefix of the observed keystream z.*

We call any $w \in \{0,1\}^m$ that satisfies these conditions an *m-candidate*. Our strategy is now to start with $m = n_{\min}$ and to dynamically compute the m-candidates for $m > n_{\min}$ until only one m-candidate is left. The first bits of

this m-candidate will contain the initial state $\omega(0)$ that we are looking for. We can expect to be left with only one m-candidate for $m \geq \lceil \alpha^{-1} n \rceil$, which follows directly from the following Lemma.

Lemma 5.15 (Krause (2002)). *Under Assumption 4.8, all keystreams z and all $m \leq \lceil \alpha^{-1} n \rceil$ satisfy $|\{\omega(0) \in \{0,1\}^n : C_m(H_{\leq m}(\omega(0)))$ is prefix of $y\}| \approx 2^{|\mathrm{IP}(m)|-\alpha m} \leq 2^{n-\alpha m}$. Hence, there exist approximately $2^{n-\alpha m}$ m-candidates.*

The key problem that we have to solve is to compute and represent the m-candidates efficiently. Our solution is based on the following BDD-based approach. Let G_m^0 denote the oracle graph over $\{w_0, \ldots, w_{m-1}\}$ that represents the order in which the keystream function C_m reads the bits from the internal bitstream. We represent the bitstreams w fulfilling conditions 1 and 2 in the minimal G_m^0-FBDDs R_m and Q_m, respectively. Starting from $P_{n_{\min}} := Q_{n_{\min}}$, we compute for $n_{\min} < m \leq \lceil \alpha^{-1} n \rceil$ the G_m^0-FBDDs $P_m := \mathrm{MIN}(P_{m-1} \wedge Q_m \wedge S_m)$, where the minimum G_m^0-FBDD S_m tests whether w_{m-1} is in the m-extension of $\mathrm{IB}(w)$. Note that we have $P_m = \mathrm{MIN}(Q_m \wedge R_m)$ with $R_m = \bigwedge_{i=1}^{m} S_i$ for all m, and P_m accepts exactly the m-candidates. This strategy is summarized in Algorithm 5.

Algorithm 5 RecoverInitialState

$P = Q_{n_{min}}$
for $m = n_{min} + 1$ to $\lceil \alpha^{-1} n \rceil$ **do**
    $P = \mathrm{MIN}(P \wedge Q_m \wedge S_m)$
end for
return the initial state bits contained in one of the $w \in P^{-1}(1)$

The efficiency of Algorithm 5 essentially depends on the sizes of the intermediate results P_m, which we are going to estimate in the following.

Assumption 5.16 (BDD Assumption). *For all $m \geq n_{min}$ we assume that $|G_m^0|, |S_m|, |Q_m| \in m^{\mathcal{O}(1)}$ and that there exists an integer $p \geq 1$ such that $|R_m| \leq |G_m^0| \cdot 2^{p \cdot |\mathrm{CP}(m)|}$.*

Lemma 5.17. *Let $\mathcal{K}$ denote an FSR-based keystream generator with k FSRs $R^0, \ldots, R^{k-1}$ of lengths $n^{(0)}, \ldots, n^{(k-1)}$, and let $n = \sum_{i=0}^{k-1} n^{(i)}$. If $\mathcal{K}$ fulfills the BDD Assumption and the Pseudorandomness Assumption, we have for all $n_{\min} < m \leq \lceil \alpha^{-1} n \rceil$*

$$|P_m| \leq \max_{1 \leq m \leq \lceil \alpha^{-1} n \rceil} \left\{ \min \left\{ \epsilon(m) |Q_m| \cdot 2^{m - |\mathrm{IP}(m)|}, m \cdot 2^{|\mathrm{IP}(m)| - \alpha m} \right\} \right\}$$
$$\leq \epsilon(m) |Q_m| \cdot 2^{\frac{p(1-\alpha)}{p+\alpha} n} \leq n^{\mathcal{O}(1)} 2^{\frac{p(1-\alpha)}{p+\alpha} n}$$

with $\epsilon(m) = |G_m^0|^2$. If there exists a variable ordering π_m such that all G_m^0-FBDDs are π_m-OBDDs, $\epsilon(m)$ reduces to the constant 1.

The proof borrows from the ideas presented by Krause (2002, 2007) and works as follows.

Proof. The definitions of Q_m and R_m imply that $P_m = Q_m \wedge R_m$ for $n_{\min} < m \leq \lceil \alpha^{-1} n \rceil$, and therefore $|P_m| \leq |G_m^0| \cdot |Q_m| \cdot |R_m|$ (FBDD Property 2) in

the FBDD-case and $|P_m| \leq |Q_m| \cdot |R_m|$ (OBDD Property 3) in the OBDD-case. Under Assumption 5.16 we obtain

$$|P_m| \leq \epsilon(m)|Q_m| \cdot 2^{p \cdot |\mathrm{CP}(m)|} \ . \tag{5.1}$$

On the other hand, Lemma 5.15 implies that $|P_m| \leq m \cdot |P_m^{-1}(1)| \approx m \cdot 2^{m^* - \alpha m}$ for $m^* = |\mathrm{IP(m)}|$ and $n_{\min} < m \leq \lceil \alpha^{-1} n \rceil$, which means

$$|P_m| \leq m \cdot 2^{m^* - \alpha m} = m \cdot 2^{(1-\alpha)m^* - \alpha \cdot |\mathrm{CP}(m)|} \ . \tag{5.2}$$

Combining Eqs. (5.1) and (5.2), we obtain for $n_{\min} < m \leq \lceil \alpha^{-1} \rceil n$

$$\begin{aligned}
|P_m| &\leq \min\{\epsilon(m)|Q_m| \cdot 2^{p \cdot |\mathrm{CP}(m)|}, m \cdot 2^{(1-\alpha)m^* - \alpha \cdot |\mathrm{CP}(m)|}\} \\
&\leq \epsilon(m)|Q_m| \cdot \min\{2^{p \cdot |\mathrm{CP}(m)|}, 2^{(1-\alpha)m^* - \alpha \cdot |\mathrm{CP}(m)|}\} \\
&= \epsilon(m)|Q_m| \cdot \min\{2^{p \cdot r(m^*)}, 2^{(1-\alpha)m^* - \alpha r(m^*)}\} \text{ with } r(m^*) = |\mathrm{CP}(m)| \\
&\leq \epsilon(m)|Q_m| \cdot 2^{p \cdot r^*(m^*)} \ ,
\end{aligned}$$

where $r^*(m^*)$ denotes the solution of $p \cdot r(m^*) = (1-\alpha)m^* - \alpha r(m^*)$. We obtain $r^*(m^*) = \frac{1-\alpha}{p+\alpha} m^*$ and hence $|P_m| \leq \epsilon(m)|Q_m| \cdot 2^{\frac{p(1-\alpha)}{p+\alpha} m^*}$. With $n_{\min} < m \leq \lceil \alpha^{-1} n \rceil$ and therefore $\epsilon(m)|Q_m| \in m^{\mathcal{O}(1)} \subseteq n^{\mathcal{O}(1)}$ and $m^* = |\mathrm{IP}(m)| \leq n$, we obtain

$$|P_m| \leq \epsilon(m)|Q_m| \cdot 2^{\frac{p(1-\alpha)}{p+\alpha} n} \leq n^{\mathcal{O}(1)} 2^{\frac{p(1-\alpha)}{p+\alpha} n} \text{ for all } n_{\min} < m \leq \lceil \alpha^{-1} n \rceil \ . \quad \square$$

From this bound on $|P_m|$, we can straightforwardly derive the time, memory and data requirements of the BDD-based attack.

Theorem 5.18. *Let $\mathcal{K}$ denote a regularly clocked FSR-based keystream generator with an unknown initial state $\omega(0) \in \{0,1\}^n$, information rate α and best-case compression ratio γ. If $\mathcal{K}$ fulfills the Independence Assumption, the Pseudorandomness Assumption and the BDD Assumption, an initial state $\tilde{s}^0$ that yields the same keystream as $\omega(0)$ can be computed with time and memory requirements in $\mathcal{O}\left(\epsilon(n)|Q_n| 2^{\frac{p(1-\alpha)}{p+\alpha} n}\right)$ from the first $\lceil \gamma \alpha^{-1} n \rceil$ consecutive keystream bits of $\mathcal{K}$ under $\omega(0)$.*

Note that by setting $p = 1$ in Theorem 5.18, we obtain the main Theorem of Krause (2002).

5.4 Generic BDD Constructions

5.4.1 Keystream Consistency Check Q_m

In most cases, a BDD Q_m that checks Condition 2 can be straightforwardly derived from the definition of the keystream function C. If the computation of a keystream bit z_t depends on $u(j) > 1$ bits from an FSR R^j, a fixed bit in the bitstream produced by R^j will generally appear and have to be read in the computation of up to $u(j)$ keystream bits. In this case, we compute a keystream bit z_t from a number of *new bits* which are being considered for the first time,

and several *old bits* that were already involved in the computation of previous keystream bits. This would imply (at least in a straightforward implementation) reading a fixed variable more than once on the same path in Q_m, which is prohibited by the FBDD-definition. The less restrictive general BDDs would permit this construction, but could no longer guarantee the efficiency of the operations that our attack depends on (cf. Remark 5.3).

A similar problem has been considered by Krause (2002) in the context of the irregularly clocked A5/1 generator (cf. Section 4.2.3), which uses the bits of the internal bitstream both for computing keystream bits and as input for the clock control mechanism. His solution was to increase the number of unknowns by working with $u(j)$ synchronized duplicates of the R^j-bitstream at the expense of a reduced information rate α.

We now consider the more general situation that the keystream function depends on new bits and some function(s) $g_1, \ldots, g_r$ in the old bits. In order to preserve the read-once property, we introduce auxiliary variables for the values of these functions such that z_t is computed only from new bits. This construction is illustrated in the following example.

Example 5.19. *Consider the keystream function $z_t = C_m(w_{t+5}, w_{t+7}, w_{t+9})$, where C_m is defined by $C_m(x_1, x_2, x_3) = x_1 \oplus x_2 \oplus x_3$. Assuming canonical reading order, w_{t+9} would be the new bit and w_{t+5} and w_{t+7} the old bits. With the auxiliary variable $\tilde{w}_t := g_1(w_{t+5}, w_{t+7})$ and $g_1(x_1, x_2) := x_1 \oplus x_2$, we can express z_t as $z_t = \tilde{w}_t \oplus w_{t+9}$.*

In general, if we add for each of the r auxiliary variables an FSR to the generator that outputs at clock t the corresponding value of g_j, we can equivalently compute z_t without considering the bits from the internal bitstream more than once. Obviously, we obtain a generator with a lowered information rate, since more bits of the internal bitstream have to be read in order to compute the same number of keystream bits.

In the case of regularly clocked keystream generators, we define the set of variable indices that the keystream function depends on as

$$I := \{i | z_t \text{ depends on } w_{t+i}\} \subseteq \{0, \ldots, n-1\} \ .$$

The bits contributed by register R^j, $j \in \{1, \ldots, k\}$, can be expressed as

$$I_j := \{i \in I | i \equiv j \bmod k\} \ .$$

Then, the set of new bits is given as

$$I^* := \{i_1^*, \ldots, i_k^*\} \text{ with } i_j^* = \operatorname{argmax}_{i \in I_j}\{\pi_m(i)\} \text{ for } j \in \{1, \ldots, k\} \ ,$$

and the old bits are those in the set $I' := I \backslash I^*$.

Concerning the information rate of the modified generator, Observation 4.7 implies:

Observation 5.20. *Fix a regularly clocked keystream generator and denote by I the set of positions in the internal bitstream that its keystream function depends on. If the keystream function can be expressed as a function depending on the k variables in $\{w_{t+i} | i \in I^*\}$ and the values of r subfunctions depending on the variables in $\{w_{t+i} | i \in I'\}$, the keystream function can be transformed into an equivalent read-once keystream generation algorithm such that the resulting generator's information rate is $\alpha = \frac{1}{k+r}$.*

5.4.2 FSR Consistency Check R_m

Recall that each bit w_t of an internal bitstream w is either an initial state bit of some FSR or a combination of other internal bits. In order to decide for a given internal bitstream whether it satisfies Condition 1, we need to check whether the update relations imposed on the bits at the combined positions are fulfilled. Hence, if a combined bit w_t is produced by an update relation $f(s_0, \ldots, s_{n-1})$, we need to check whether $f(w_{i_1}, \ldots, w_{i_p}) = w_t$, which is equivalent to testing whether

$$\tilde{f}(w_{i_1}, \ldots, w_{i_p}, w_t) := f(w_{i_1}, \ldots, w_{i_p}) \oplus w_t = 0 \ .$$

The OBDD S_m implements this test for the single combined bit w_{m-1} and represents the constant-one function if w_{m-1} is an initial bit. The OBDD $R_m = \bigwedge_{i=1}^{m} S_i$ performs the consistency tests for the whole internal bitstream.

We first consider the case of FSRs (without additional memory), for which we need the following definition.

Definition 5.21. *For a polynomial* $f : \{0,1\}^n \to \{0,1\}$ *with*

$$f(w_1, \ldots, w_n) = \bigoplus_{j \in M} m_j \text{ with monomials } m_j = \bigwedge_{l \in M^j} w_l \text{ and } M^j(f) \subseteq \{1, \ldots, n\}$$

and a reading order $\pi \in \sigma_n$*, we define the set of* active monomials *at time* t *as*

$$\mathrm{AM}_\pi(f,t) := \{m_j : 0 < |\{\pi^{-1}(1), \ldots, \pi^{-1}(t)\} \cap M^j(f)| < |M^j(f)|\} \ .$$

Hence, $\mathrm{AM}(f,t)$ contains all monomials in f for which at least one, but not all factors are known after the first t inputs have been read.

Lemma 5.22. *For a polynomial* $f : \{0,1\}^n \to \{0,1\}$ *with* $n > 1$ *and a reading order* π *for the inputs, the set of inputs satisfying* $f(w_1, \ldots, w_n) = 0$ *can be represented in a* π*-OBDD of width* $2^{\max_{1 \le t \le n}\{|\mathrm{AM}_\pi(f,t)|\}+1}$.

Proof. Let $p := \max_{1 \le t \le n}\{|\mathrm{AM}_\pi(f,t)|\}$. In order to compute $f(w_1, \ldots, w_n)$, we may proceed in the following way. We define p auxiliary variables $b_1, \ldots, b_p$, which will store the intermediate values of partly evaluated monomials, and an additional variable b_0 for the sum of evaluated monomials. We initialize $b_0 := 0$, $b_t := 1$ for $t > 0$, and read the variables $w_1, \ldots, w_n$ in the order given by π. For each variable w_t, we update all auxiliary variables that are associated with monomials containing w_t. If a monomial becomes active by reading w_t, we allocate an auxiliary variable b_j and define $b_j := w_t$. If a monomial is entirely evaluated after reading w_t, we add its value to b_0 and free the associated auxiliary variable. Since there are at most p active monomials at any time, no more than $p+1$ auxiliary variables will be needed simultaneously.

Observation 5.14 implies that this strategy can be transformed into a π-OBDD of width 2^{p+1}, which imples the claim. □

From Lemma 5.22, we can directly derive an upper bound for the width of the π_m-OBDD S_m for an FSR.

Corollary 5.23. *For a given reading order* π_m*, an integer* $m > 0$*, an FSR* R *with update relation* f*, and* $p := \max_{0 \le t < m}\{|\mathrm{AM}_{\pi_m}(\tilde{f}, t)|\} + 1$*, we can construct a* π_m*-OBDD* S_m *of width at most* 2^p *that tests for an internal bitstream* $w \in \{0,1\}^m$ *if* w *fulfills the update relation imposed on* w_{m-1}.

Remark 5.24. *For the special case $p = 1$, we obtain the LFSR-bound that was proved by Krause (2002).*

We now turn to the case of Fibonacci FCSRs. Eq. (3.6) implies that we need access to σ_{t-1} in order to check whether the update relation holds for w_t. Therefore, we work with a modified FCSR that essentially outputs the sum σ_t instead of the bit $w_t = \sigma_t \bmod 2$ in each clock. For a Fibonacci FCSR with p bits of additional memory, we let the modified FCSR output for an initial memory state $(b^0_{p-1}, \ldots, b^0_0)$ with $b^0 = \sum_{i=0}^{p-1} b^0_i 2^i$ the values $y^0_t =: \sigma^0_t$ for $t < n-1$, $(b^{p-1}_0, \ldots, b^0_0, y^0_{n-1})$ for $t = n-1$, and $(\sigma^p_t, \sigma^{p-1}_t, \ldots, \sigma^0_t)$ for $t \geq n$ with $\sigma_t = \sum_{i=0}^{p} \sigma^i_t 2^i$ and $w_t = \sigma^0_t$.

Note that a bit w_m in the output of the modified FCSR, $m \geq 0$, then corresponds to the i-th component bit of some intermediate sum σ_t with

$$(i,t) = \tau(m) := \begin{cases} (0, m) & \text{if } m < n-1 \\ (m - (n-1) \bmod (p+1), & \text{otherwise} \\ (m - (n-1) \operatorname{div} (p+1)) + (n-1)) & \end{cases} .$$

Lemma 5.25. *For a Fibonacci FCSR R with p bits of additional memory, an integer $m > 0$, and a reading order π_m, we can construct a π_m-OBDD S_m of width at most 2^{p+1} that tests for the internal bitstream $w \in \{0,1\}^m$ of the modified FCSR with $m = n - 1 + t(p+1)$ whether the last $p+1$ bits fulfill the update relation.*

Proof. In order to check whether $\sigma_t = (\sigma_{t-1} \operatorname{div} 2) + \sum_{i=1}^{n} w_{t-i} \cdot d_{i-1}$, we can equivalently test if

$$\sigma_t = \sum_{i=1}^{p} \sigma^i_{t-1} \cdot 2^i + \sum_{i=1}^{n} \sigma^0_{t-i} \cdot d_{i-1} \ ,$$

since $w_t = \sigma_t \bmod 2 = \sigma^0_t$.

Algorithm 6 describes a read-once algorithm that checks whether the last $p+1$ bits of a bitstream $w \in \{0,1\}^m$ are consistent with the values of the remaining bits in w. Since the algorithm uses exactly $p+1$ bits of additional memory, Observation 5.14 implies that it can be transformed into an OBDD of width at most 2^{p+1}, which implies the claim. □

Note that according to Corollary 3.26, we have $p \approx \log(\mathrm{d})$ for a periodic initial FCSR-state, where d denotes the FCSR's feedback tap vector.

In the case of Galois FCSRs with $a_i \leq d_i$ at all times, we denote by $x_i(t)$ and $a_i(t)$ the value of the register cells x_i and a_i at time t. We think of the main register of the Galois FCSR as producing the bitstream

$$x_0(0), x_1(0), \ldots, x_{n-1}(0), \ldots, x_{i_1}(t), \ldots, x_{i_l}(t), \ldots \ ,$$

where $i_j \in \{1 \leq i < n | d_i = 1\}$, $l = \mathrm{wt}(d) - 1$, $i_j < i_{j'}$ for $j < j'$, and $t > 0$. Similarly, we view the bitstream produced by the carry register as $a_{i_1}(t), \ldots, a_{i_l}(t), \ldots$ for $t \geq 0$.

Lemma 5.26. *For a Galois FCSR R with $a_{i-1} \leq d_{i-1}$ for all $i \in \{1, \ldots, n-1\}$, an integer $m > 0$, and a reading order π_m, we can construct a π_m-OBDD S_m*

Algorithm 6 FibonacciFCSR-$S_m(\pi_m, w)$

```
Let (i,t) := τ(m)
if t < n − 1 then
  return true // Nothing to check for initial bits
end if
σ̃ := 0 // Initialize the (p+1)-bit auxiliary variable
for j = 0 to m − 1 do
  (i′,t′) := τ((π_m)^{−1}(j)) // Determine the σ^{i′}_{t′} that corresponds to the currently read variable w_j
  if t′ ∈ {t − n, ..., t − 1} and i′ = 0 then
    σ̃ := σ̃ + w_j · d_{t−t′−1}
  end if
  if t′ = t − 1 and i′ ∈ {1, ... p} then
    σ̃ := σ̃ + w_j · 2^{i′}
  end if
  if t′ = t then
    if σ̃^{i′} ≠ σ^{i′}_{t′} then
      return false
    end if
  end if
end for
return true
```

of width at most 2 *that tests whether a bit in the bitstream produced by the main register fulfills the corresponding update relations. For a bit in the bitstream of the carry register, we can perform this consistency test in a* π_m*-OBDD of maximum width* 8.

Proof. The definition of Galois FCSRs implies $x_{n-1}(t) = x_0(t-1)$ and for $i \in \{n-2, \ldots, 0\}$ that $x_i(t) = x_{i+1}(t-1)$ if $d_i = 0$ and $x_i(t) = x_{i+1}(t-1) \oplus a_i(t-1) \oplus x_0(t-1)$ if $d_i = 1$. Note that we have $x_{i_j+1}(t-1) = x_{i_{j+1}}(t-(i_{j+1}-i_j))$ and therefore

$$\begin{aligned} x_{i_j}(t) &= x_{i_j+1}(t-1) \oplus a_{i_j}(t-1) \oplus x_0(t-1) \\ &= x_{i_{j+1}}(t-(i_{j+1}-i_j)) \oplus a_{i_j}(t-1) \oplus x_0(t-1) \ . \end{aligned}$$

According to Corollary 5.23, we can test these linear conditions in a π_m-OBDD of width at most 2.

Similarly, Corollary 5.23 yields a maximum width of $2^3 = 8$ in the case of the carry register, since $b_{i_j}(t)$ can be computed as

$$\begin{aligned} a_{i_j}(t) &= x_{i_j+1}(t-1)a_{i_j}(t-1) \oplus a_{i_j}(t-1)x_0(t-1) \oplus x_0(t-1)x_{i_j}(t-1) \\ &= x_{i_{j+1}}(t-(i_{j+1}-i_j))a_{i_j}(t-1) \oplus a_{i_j}(t-1)x_0(t-1) \\ &\oplus x_0(t-1)x_{i_{j+1}}(t-(i_{j+1}-i_j)) \ , \end{aligned}$$

which implies the claim. □

From the bounds on $w(S_m)$ for the different types of FSRs, we can now straightforwardly derive a bound for $w(R_m)$ for an FSR-based keystream generator. Let $\mathcal{K}$ denote an FSR-based keystream generator consisting of k FSRs

$R^0, \ldots, R^{k-1}$ with π_m-OBDDs $S_m^0, \ldots, S_m^{k-1}$ and $w(S_m^i) \leq 2^{p_i}$ for all $i \in \{0, \ldots, k-1\}$. Moreover, let s_i denote the fraction of combined bits that R^i contributes to the internal bitstream.

Corollary 5.27. *There exists a π_m-OBDD R_m of width at most $2^{|\mathrm{CP}(m)| \sum_{i=0}^{k-1} p_i s_i}$ that tests for a potential internal bitstream $w \in \{0,1\}^m$ of an FSR-based keystream generator whether it is an m-extension of the initial bits.*

Proof. The claim follows directly from $R_m = \bigwedge_{i=1}^{m} S_i$ and the OBDD-properties described in Section 5.2. □

5.5 Applications

5.5.1 Self-Shrinking Generator

As discussed in Section 4.2.1, the self-shrinking generator consists of only one LFSR and no memory. It produces at most m keybits from an internal bitstream w_{2m}, i.e., $\gamma \cdot 2m = m$ and $\gamma = 0.5$.

Lemma 5.28. *For all keystreams $z \in \{0,1\}^*$, there exist at most $\binom{m/2}{|z|} 2^{m/2-|z|}$ internal bitstreams $w \in \{0,1\}^m$ such that $C_m(w) = z$.*

Proof. We first observe that due to $\gamma = 0.5$, we have $|C_m(w)| \leq 0.5m$, i.e., no internal bitstream of length m can produce more than $\frac{m}{2}$ keystreams bits. We fix a keystream z of length at most $\frac{m}{2}$ and let $Z_z = \{w \in \{0,1\}^m : C_m(w) = z\}$. For each $w \in Z_z$, there exists a set $I = \{i_1, \ldots, i_{|z|}\} \subseteq \{0, \ldots, \frac{m}{2}\}$ such that $(w_{2i_j}, w_{2i_j+1}) = (1, z_j)$ for $j \in \{1, \ldots, |z|\}$.

Moreover, w must satisfy $w_{2i_j} = 0$ for all $i_j \in \{0, \ldots, \frac{m}{2}\} \backslash I$. There are $\binom{m/2}{|z|}$ possible choices for I and $2^{m/2-|z|}$ possible assignments to the unrestricted variables. Hence,

$$|Z_z| = \binom{m/2}{|z|} 2^{m/2-|z|} \ .$$

□

Corollary 5.29 (Krause (2002)). *The information rate of the self-shrinking generator is $\alpha = 1 - \frac{\log(3)}{2} \approx 0.2075$.*

Proof. Lemma 5.28 implies that for a $z \in \{0,1\}^*$,

$$\begin{aligned} |\{w \in \{0,1\}^m : C_m(w) \text{ is prefix of } z\}| &= \sum_{|z|=0}^{m/2} \binom{m/2}{|z|} 2^{m/2-|z|} \\ &= \sum_{|z|=0}^{m/2} \binom{m/2}{|z|} 1^{|z|} 2^{m/2-|z|} \\ &= (1+2)^{m/2} = 3^{m/2} \ . \end{aligned}$$

Hence,

$$\begin{aligned} \alpha &= -\frac{1}{m} \Pr_w[C_m(w) \text{ is Prefix of } z] \\ &= -\frac{1}{m} \frac{|\{w \in \{0,1\}^m : C_m(w) \text{ is prefix of } z\}|}{2^m} \\ &= 1 - \frac{\log(3)}{2} \approx 0.2075 \ , \end{aligned}$$

which concludes the proof. □

Algorithm 7 tests whether a given internal bitstream w is consistent with a keystream prefix z. Since the algorithm is read-once and uses at most $\lfloor \log(m) \rfloor + 1$ bits of additional memory, Observation 5.14 implies that it can be transformed into a π_m-OBDD Q_m with $w(Q_m) \leq m$ and $|Q_m| \leq m^2$, where π_m denotes the canonical reading order.

Algorithm 7 SelfShrinkingGenerator-$Q_m(w, z)$

```
t := 0
u := 0 // auxiliary variable u, u ≤ ⌊m/2⌋
while t < m − 1 do
  if w_t = 1 then
    if z_u ≠ w_{t+1} then
      return false
    end if
    u := u + 1
  end if
  t := t + 2
end while
return true
```

Altogether, we obtain from Theorem 5.18, Corollary 5.23, and Remark 5.24:

Corollary 5.30 (Krause (2002)). *From a prefix of length $\lceil 2.41n \rceil$ of a keystream $z = C_m(L(x))$ produced by a self-shrinking generator of key length n, an initial state $\tilde{x}$ with $C_m(L(\tilde{x})) = z$ can be computed in time and with space in $\mathcal{O}(n^2 \cdot 2^{0.6563n})$.*

Compared with the attacks mentioned in Section 4.2.1, the BDD-Attack is almost as fast as the currently best short-keystream attack due to Hell and Johansson (2006), but consumes exponentially more memory.

5.5.2 Bluetooth Keystream Generator E_0

Recall from Section 4.2.2 that the E_0 keystream generator is a regularly clocked $(4, 4)$-combiner with 4 LFSRs, a 4-bit memory unit and an internal state size of 128 bits. Therefore, we have $\alpha = \gamma = \frac{1}{4}$.

Algorithm 8 tests whether a given internal bitstream w is consistent with an observed E_0-keystream z, given that the initial memory state is q_0. Exploiting Eqs. (4.1) and (4.2), the algorithm relies on a lookup-table $\delta' : \{0, \ldots, 4\} \times \{0, 1\}^4$ that maps the sum $s'_t = \sum_{i=0}^{3} w_t^i$ and the current memory state q_t to the next memory state $q_{t+1} = \delta'(s'_t, q_t)$. Since the algorithm uses a constant amount of additional memory, it can be transformed into a π_m-OBDD Q_m of constant maximum width, as indicated by Observation 5.14. Similarly to the case of the Self-Shrinking Generator, π_m denotes the canonical reading order.

In summary, Theorem 5.18, Corollary 5.23, and Remark 5.24 imply the following performance figures for the BDD-based attack on E_0.

Corollary 5.31 (Krause (2002)). *From a prefix of length n of a keystream $z = C_m(L(x))$ produced by an E_0 keystream generator of key length n, an initial*

Algorithm 8 E_0-$Q_m(q_0, w, z)$

```
// w is interpreted as w = w_0^0, ..., w_0^3, ..., w_j^0, ..., w_j^3, ..., w_{m-1 div 4}^{m-1 mod 4}
t := 0
s := 0 // auxiliary variable for the integer sum, s ∈ {0, ..., 4}
q := q_0 // auxiliary variable for the memory state
for t = 0 to ⌊m/4⌋ − 1 do
  s := w_t^0 + w_t^1 + w_t^2 + w_t^3
  if (s mod 2) ⊕ ⨁_{i=0}^{3} c^i q ≠ z_t then
    return false
  end if
  q := δ'(s, q)
end for
return true
```

state $\tilde{x}$ with $C_m(L(\tilde{x})) = z$ can be computed in time and with space in $\mathcal{O}(n \cdot 2^{0.6n})$, i.e., with $2^{76.8}$ polynomial-time operations for $n = 128$.

The BDD-Attack slightly improves the attack by Fluhrer and Lucks (2001), which trades off time and the number of required keystream bits. For the minimum number of 132 available keystream bits the attack needs 2^{84} polynomial time operations.

5.5.3 GSM Keystream Generator A5/1

We note that a bit that serves as input for the keystream function f at a particular time has been considered a few clockings earlier by the clock control mechanism. Based on our observations in Section 5.4.1, we therefore duplicate the three registers of A5/1 such that the keystream bits are computed from the first three registers and the clock control operates on the remaining registers. However, the keystream generation algorithm of the resulting generator is still not read-once, since the bits of unclocked registers are reconsidered in subsequent iterations. The read-once Algorithm 9 fixes this problem by introducing auxiliary variables for these unchanged values. It can be straightforwardly checked that this algorithm is equivalent to the original A5/1 algorithm.

Due to the irregular clocking of the A5/1 algorithm, the information rate α is a little less straightforward to compute, but can be determined as follows.

Lemma 5.32 (Krause (2002), Stegemann (2004)). *The information rate of the modified A5/1 keystream generator is given by*

$$\alpha = \frac{1}{2} \log u_1 \approx 0.2193 \ ,$$

where u_1 denotes the positive real root of the polynomial $p(u) = u^3 - 3u^2 + 8$.

Proof. The Independence Assumption has been shown to hold for A5/1 by Krause (2002) and Stegemann (2004). For an arbitrary keystream z, $m \geq 1$, and a randomly chosen internal bitstream w, $|w| = m$, let

$$p(m) := \Pr_w[C_m(w) \text{ is prefix of } z] \ .$$

Algorithm 9 read-once-A5/1(w)

// w is interpreted as $w = w_0^0, \ldots, w_0^5, \ldots, w_j^0, \ldots, w_j^5, \ldots, w_{m-1 \text{ div } 6}^{m-1 \bmod 6}$
$i := [0,0,0]$ // current read positions
$u :=$NIL // unchanged index $\in \{0,1,2,NIL\}$
$v :=$NIL // unchanged control value $\in \{0,1,NIL\}$
$v' :=$NIL // unchanged output value $\in \{0,1,NIL\}$
$t := 0$
while (true) **do**
  if $\exists r \in \{0,1,2\}\backslash\{u\} : 6 \cdot i[r] + r \geq m$ **then**
    stop
  end if
  $Read := \{0,1,2\}\backslash\{u\}$
  Let $r_0, \ldots, r_{|Read|-1}$ the elements of $Read$ in ascending order, i.e. $r_m < r_n$ for $m < n$
  $out[r_0] := w_{i[r_0]}^{r_0}$
  $out[r_1] := w_{i[r_1]}^{r_1}$
  if $u \neq NIL$ **then** // $\exists$ an unchanged index
    $out[u] := v'$ // copy the unchanged output value
  else // all read positions incremented
    $out[r_2] := w_{i[r_2]}^{r_2}$ // read the third output value
  end if
  output $z_t = out[0] \oplus out[1] \oplus out[2]$
  if $\exists r \in \{0,1,2\}\backslash\{u\} : 6 \cdot i[r] + 3 + r \geq m$ **then**
    stop
  end if
  $c[r_0] := w_{i[r_0]}^{3+r_0}$
  $c[r_1] := w_{i[r_1]}^{3+r_1}$
  if $u \neq NIL$ **then** // $\exists$ an unchanged index
    $c[u] := v$ // copy the unchanged control value
  else // all read positions incremented
    $c[r_2] := w_{i[r_2]}^{3+r_2}$ // read the third control value
  end if
  $controlbit := maj_3(c[0], c[1], c[2])$
  if $\exists r \in \{0,1,2\} : c[r] \neq controlbit$ **then**
    // By definition of maj_3, $\exists$ at most one such r
    $u := r$ // set unchanged index
    $v := controlbit \oplus 1$ // set unchanged control value
    $v' := out[r]$ // set unchanged output value
  else // all read positions incremented
    $u := v := v' := NIL$
  end if
  for $l \in \{0,1,2\}\backslash\{u\}$ **do**
    $i[l] := i[l] + 1$ // increment read positions
  end for
  $t := t + 1$
end while

Moreover, define for $m \geq 0$ and $k \leq m$

$$p(m,k) := \Pr_{w \in_u \{0,1\}^m}[|C_m(w)| = k] \ .$$

Since a keystream bit is computed either from 4 or 6 internal bits, we have $\frac{m}{6} \leq |C_m(w)| \leq \frac{m}{4}$. Based on Equation (4.3) we can express $p(m)$ as

$$\begin{aligned} p(m) &= \Pr_w[C_m(w) \text{ is prefix of } z] \\ &= \sum_{i=0}^{\lceil \gamma m \rceil} \Pr_w[|C_m(w)| = i] \cdot \Pr_{|C_m(w)|=i}[C_m(w) = z_0, \ldots, z_{i-1}] \\ &= \sum_{k=\frac{m}{6}}^{\frac{m}{4}} p(m,k) \cdot 2^{-k} \ . \end{aligned}$$

Furthermore, Lemma 4.5 implies $\alpha = -\frac{1}{m}\log_2(p(m))$. Therefore, we now derive a suitable recurrence relation for $p(m)$ and use this expression to compute α.

Let W denote the random variable that stores the number of internal bits that were used for the computation of the first $\frac{m}{6}$ keystream bits. Moreover, let W' store the number of keystream bits that were computed from 6 internal bits. These definitions imply

$$W = 6 \cdot W' + 4 \cdot (\frac{m}{6} - W') \ .$$

The number of internal bits that have not been read after the $\frac{m}{6}$ keystream bits have been produced is given by

$$m - W = m - (6 \cdot W' + 4 \cdot (\frac{m}{6} - W')) = \frac{1}{3}m - 2W' \ .$$

We obtain the following recurrence relation for $p(m)$.

$$\begin{aligned} p(m) &= \sum_{i=0}^{\frac{m}{6}} \left(2^{-\frac{m}{6}} \Pr[W' = i] \cdot \sum_{j=(\frac{1}{3}m-2i)/6}^{(\frac{1}{3}m-2i)/6} 2^{-j} p\left(\frac{1}{3}m - 2i, j\right) \right) \\ &= \sum_{i=0}^{\frac{m}{6}} 2^{-\frac{m}{6}} \Pr[W' = i] \cdot p\left(\frac{1}{3}m - 2i\right) \end{aligned}$$

Since W' is $\left(\frac{m}{6}, \frac{1}{4}\right)$-binomially distributed (cf. Stegemann (2004) for a proof), we can write $\Pr[W' = i]$ as

$$\Pr[W' = i] = \binom{\frac{m}{6}}{i} \left(\frac{1}{4}\right)^i \left(\frac{3}{4}\right)^{\frac{m}{6}-i} \ ,$$

which implies

$$p(m) = \sum_{i=0}^{\frac{m}{6}} 2^{-\frac{m}{6}} \binom{\frac{m}{6}}{i} \left(\frac{1}{4}\right)^i \left(\frac{3}{4}\right)^{\frac{m}{6}-i} \cdot p\left(\frac{1}{3}m - 2i\right)$$

With $p(m) = 2^{-\alpha m}$, we obtain a recurrence relation for α:

$$\begin{aligned} 2^{-\alpha m} &= \sum_{i=0}^{\frac{m}{6}} 2^{-\frac{m}{6}} \binom{\frac{m}{6}}{i} \left(\frac{1}{4}\right)^i \left(\frac{3}{4}\right)^{\frac{m}{6}-i} \cdot 2^{-\alpha\left(\frac{1}{3}m-2i\right)} \\ &= 2^{-(\frac{1}{6}+\frac{\alpha}{3})m} \sum_{i=0}^{\frac{m}{6}} \binom{\frac{m}{6}}{i} \left(\frac{1}{4}\right)^i \left(\frac{3}{4}\right)^{\frac{m}{6}-i} \cdot 2^{2\alpha i} \end{aligned}$$

Lemma 5.45 implies that this equation is equivalent to

$$\begin{aligned} 2^{-\alpha m} &= 2^{-(\frac{1}{6}+\frac{\alpha}{3})m} \left(\frac{1}{4}\right)^{\frac{m}{6}} (3 + 2^{2\alpha})^{\frac{m}{6}} \\ \iff 2^{-6\alpha} &= 2^{-(1+2\alpha)} \frac{1}{4}(3 + 2^{2\alpha}) \\ \iff 2^{1-4\alpha} &= \frac{1}{4}(3 + 2^{2\alpha}) \\ \iff 2(2^{2\alpha})^{-2} - \frac{1}{4}(2^{2\alpha}) - \frac{3}{4} &= 0 \end{aligned}$$

By substituting $u := 2^{2\alpha}$, i.e., $\alpha = \frac{1}{2}\log u$, we finally obtain

$$u^3 + 3u^2 - 8 = 0 \ . \qquad \square$$

The read-once Algorithm 10 tests for a given internal bitstream w and an observed keystream z whether z could have been produced by w. In order to transform Algorithm 10 into an FBDD, we proceed as follows. We first convert the function *output-test* into an $\mathrm{output-FBDD}(i, u, v, v', t)$. Let $\mathrm{out}[r_j]$ denote the value of the variable $w_{i[r_j]}^{r_j}$ for $j \in \{0, 1, 2\}$ and let

$$\hat{v}' := \begin{cases} \mathrm{out}[r_0] \oplus \mathrm{out}[r_1] \oplus \mathrm{out}[r_2] & u = \mathrm{NIL} \\ \mathrm{out}[r_0] \oplus \mathrm{out}[r_1] \oplus v' & u \neq \mathrm{NIL} \end{cases} .$$

Then, $\mathrm{output-FBDD}(i, u, v, v', t)$ is a complete binary tree of depth 2 if $u \neq \mathrm{NIL}$ and a complete binary tree of depth 1 if $u = \mathrm{NIL}$ with

$$q_j = \begin{cases} \mathrm{control-FBDD}(i, u, v, \hat{v}', out, t) & \text{if } \hat{v}' = w_t \\ 0-\mathrm{sink} & \text{otherwise} \end{cases} .$$

If there exists an $r \in \{0, 1, 2\} \backslash \{u\}$ such that $6 \cdot i[r] + r \geq m$, then

$$\mathrm{output-FBDD}(i, u, v, v', t)$$

is identical to the 1-sink. Similarly, in order to convert the function $\mathrm{control-test}$ into a

$$\mathrm{control-FBDD}(i, u, v, v', out, t) \ ,$$

let $c[r_j]$ denote the value of $w_{i[r_j]}^{3+r_j}$, set

$$\mathrm{controlbit} := \begin{cases} \mathrm{maj}_3(c[r_0], c[r_1], c[r_2]) & \text{if } u = \mathrm{NIL} \\ \mathrm{maj}_3(c[r_0], c[r_1], v) & \text{if } u \neq \mathrm{NIL} \end{cases} ,$$

and define $\hat{i}$, $\hat{u}$, $\hat{v}$, and $\hat{v}'$ as in the function $\mathrm{control-test}$.

control − FBDD(i, u, v, v', t) is then a complete binary tree of depth 2 if $u \neq$ NIL and a complete binary tree of depth 1 if $u =$ NIL with

$$q_j = \text{output} - \text{FBDD}(\hat{i}, \hat{u}, \hat{v}, \hat{v}', t+1) \ .$$

As above, if there exists an $r \in \{0,1,2\}\backslash\{u\}$ such that $6 \cdot i[r] + 3 + r \geq m$, then control − FBDD$(i, u, v, v', t)$ is identical to the 1-sink.

Altogether, the FBDD Q_m is given by

$$\text{output} - \text{FBDD}([0,0,0], \text{NIL}, \text{NIL}, \text{NIL}, 0) \ .$$

Since the sub-FBDDs output − FBDD and control − FBDD have constant sizes and there are at most $\mathcal{O}(m^4)$ different sub-FBDDs, the size of Q_m satisfies $|Q_m| \in \mathcal{O}(m^4)$.

Ignoring the current position t in the keystream and omitting the test whether $z_t = \hat{v}'$ in the function output − test, we can straightforwardly derive from Q_m the oracle graph G_m^0 that defines the order in which the bits of the w_j are read in Q_m. It is easy to see that $|G_m^0| \in \mathcal{O}(m^3)$.

Algorithm 10 A5/1-$Q_m(w, z)$

// w is interpreted as $w = w_0^0, \ldots, w_0^5, \ldots, w_j^0, \ldots, w_j^5, \ldots, w_{m-1 \text{ div } 6}^{m-1 \bmod 6}$
export w, m, z // global variables
return output-test([0,0,0],NIL,NIL,NIL,0)

The definition of A5/1 implies that the keystream function reads the bits produced by a fixed LFSR R^j, $j \in \{0, \ldots 5\}$, in canonical order. Hence, the reading order π_m^j defined by $\pi_m^j(i) := i \text{ div } k$ for $i \equiv j \bmod 6$ is consistent with G_m^0 in the sense of Definition 5.11. Observation 5.12 then implies that the π_m^j-OBDD S_m with $j = (m-1) \bmod 6$ constructed according to Corollary 5.23 and Remark 5.24 is a G_m^0-FBDD.

Obviously, at most $\frac{m}{4}$ keybits are produced from an internal bitstream of length m, which implies $\gamma = \frac{1}{4}$.

In summary, we obtain by plugging the computed values into the statements of Theorem 5.18:

Corollary 5.33 (Krause (2002)). *From a prefix of length $\lceil 1.14n \rceil$ of a keystream $y = C_m(L(x))$ produced by an A5/1 keystream generator of key length n, an initial state $\tilde{x}$ with $C_m(L(\tilde{x})) = y$ can be computed in time and with space in $\mathcal{O}(n^{10} \cdot 2^{0.6403n})$, i.e, with $2^{0.6403n} = 2^{41}$ polynomial-time operations for $n = 64$.*

We note that since $\lceil 1.14n \rceil = 73$ and the framelength in GSM is 114 Bits for each direction , we only need the first frame, i.e., the first around 4.6 milliseconds, of a conversation in order to reconstruct the initial state of the keystream generator.

5.5.4 Trivium

Trivium (see Section 4.2.4) is a regularly clocked keystream generator consisting of three interconnected NFSRs R^0, R^1, R^2 of lengths $n^{(0)} = 93$, $n^{(1)} = 84$, and $n^{(2)} = 111$. The 288-bit initial state of the generator is derived from an 80 bit

function output-test(i, u, v, v', t)
// i=current read positions
// u=unchanged index $\in \{0, 1, 2, NIL\}$
// v=unchanged control value $\in \{0, 1, NIL\}$
// v'=unchanged output value $\in \{0, 1, NIL\}$
// t=current position in the keystream
if $\exists r \in \{0,1,2\}\backslash\{u\} : 6 \cdot i[r] + r \geq m$ **then**
  return *true*
end if
$Read := \{0,1,2\}\backslash\{u\}$
Let $r_0, \ldots, r_{|Read|-1}$ the elements of $Read$ in ascending order, i.e. $r_m < r_n$ for $m < n$
$out[r_0] := w^{r_0}_{i[r_0]}$
$out[r_1] := w^{r_1}_{i[r_1]}$
if $u \neq NIL$ **then** // $\exists$ an unchanged index
  $out[u] := v'$ // copy the unchanged output value
else // all read positions incremented
  $out[r_2] := w^{r_2}_{i[r_2]}$ // read the third output value
end if
$\hat{v}' := out[0] \oplus out[1] \oplus out[2]$
if $z_t \neq \hat{v}'$ **then**
  return *false* // keystream inconsistent
end if
return control-test$(i, u, v, \hat{v}', out, t)$

key and an 80 bit IV. The keystream function computes a keystream bit z_t by linearly combining six bits of the internal state, with each NFSR contributing two bits (cf. Section 4.2.4 for details). In order to mount the BDD-attack on TRIVIUM, we write the keystream function as

$$z_t = g_1(s_1, s_{94}, s_{178}) \oplus s_{28} \oplus s_{109} \oplus s_{223}$$

and proceed as described in Section 5.4.1 by adding an LFSR R^3 which computes g_1 to the generator. For π_m equal to the canonical reading order, we have $p_i = \max_{1 \leq t \leq 288}\{|\mathrm{AM}_{\pi_m}(\tilde{f}^i, t)|\} + 1 = 2$ and $s_i = \frac{1}{4}$ for $i \in \{0,1,2\}$ as well as $p_3 = 1$ and $s_3 = \frac{1}{4}$, which implies $p = \sum_{i=0}^{3} p_i s_i = \frac{7}{4}$. Since the modified generator computes one keystream bit from four internal bits, we have $\beta(m) = \frac{1}{4}m$ and $\alpha = \gamma = \frac{1}{4}$. Based on Lemma 5.22, we can obviously construct a π_m-OBDD Q_m with $w(Q_m) \leq 2$ that performs the consistency test for the observed keystream z.

Observation 5.34. *For* TRIVIUM *we have* $\Pr_w[C_m(w) \textit{is prefix of } z] = p_C(m) = 2^{-|C_m(w)|}$*, i.e., the Independence Assumption holds.*

Proof. Let $\tilde{z} = (\tilde{z}_1, \ldots, \tilde{z}_{|\tilde{z}|})$ denote an arbitrary $|\tilde{z}|$-bit keystream. Since $w^{(m)}$ is randomly chosen and uniformly distributed, we have

$$\Pr_w[z_1 = w_1 + w_{28} + w_{94} + w_{109} + w_{178} + w_{223} = \tilde{z}_1] = \frac{1}{2} \ .$$

function control-test(i, u, v, v', out, t)
if $\exists r \in \{0,1,2\}\backslash\{u\} : 6 \cdot i[r] + 3 + r \geq m$ **then**
  return *true*
end if
$Read := \{0,1,2\}\backslash\{u\}$
Let $r_0, \ldots, r_{|Read|-1}$ the elements of $Read$ in ascending order, i.e. $r_m < r_n$ for $m < n$
$c[r_0] := w_{i[r_0]}^{3+r_0}$
$c[r_1] := w_{i[r_1]}^{3+r_1}$
if $u \neq NIL$ **then** // $\exists$ an unchanged index
  $c[u] := v$ // copy the unchanged control value
else // all read positions incremented
  $c[r_2] := w_{i[r_2]}^{3+r_2}$ // read the third control value
end if
$controlbit := maj_3(c[0], c[1], c[2])$
if $\exists r \in \{0,1,2\} : c[r] \neq controlbit$ **then**
  // By definition of maj_3, $\exists$ at most one such r
  $\hat{u} := r$ // set unchanged index
  $\hat{v} := controlbit \oplus 1$ // set unchanged control value
  $\hat{v}' := out[r]$ // set unchanged output value
else // all read positions incremented
  $\hat{u} := \hat{v} := \hat{v}' := NIL$
end if
for $l \in \{0,1,2\}$ **do**
  $\hat{i}[l] := i[l] + \begin{cases} 1 & l \neq u \\ 0 & l = u \end{cases}$
end for
return output-test$(\hat{i}, \hat{u}, \hat{v}, \hat{v}', t+1)$

Three of the six internal bits utilized in the computation of a particular keybit will be reused in later steps, but each step also involves three previously unconsidered bits. Hence, the claim follows by induction. □

By plugging α, γ and p into the statement of Theorem 5.18, we obtain:

Theorem 5.35. *The secret initial state of the* TRIVIUM *automaton can be recovered from the first n keystream bits in time and with space in $\mathcal{O}\left(n \cdot 2^{0.65625n}\right) \approx 2^{189}$ for $n = 288$.*

Theorem 5.35 shows that the BDD-attack is applicable to TRIVIUM, but its performance is not competitive with recently published attacks requiring only 2^{45} operations as described in Section 4.2.4.

5.5.5 Grain-128

The regularly clocked stream cipher Grain-128 (see Section 4.2.5) has a key size of 128 bits and an IV size of 96 bits. The design is based on two interconnected shift registers, an LFSR R^0 and an NFSR R^1, both of lengths $n^{(0)} = n^{(1)} = 128$ and a nonlinear keystream function. We denote the content of the LFSR by $s_t, s_{t+1}, \ldots, s_{t+127}$ and the content of the NFSR by $b_t, b_{t+1}, \ldots, b_{t+127}$. The corresponding update functions and the keystream function are given in Section 4.2.5.

We add to the keystream generator an NFSR R^2 which computes the keystream bits z_t and have the generator output the values produced by R^2 in each clock.

More precisely, we can compute z_t as

$$z_t = g_1(b_{t+2}, b_{t+15}, b_{t+36}, b_{t+45}, b_{t+64}, b_{t+73}, b_{t+89}, b_{t+12}, s_{t+8}, s_{t+13}, s_{t+20}, s_{t+60}, s_{t+79}) \oplus b_{t+95} g_2(s_{t+42}) \oplus g_3(b_{t+12}) b_{t+95} s_{t+95} \ ,$$

where $g_2(s_{t+42}) = s_{t+42}$ and $g_3(b_{t+12}) = b_{t+12}$ and g_1 contains 3 monomials of degree 2.

Hence, we can compute one keystream bit from 3 internal bits, which implies $\beta(m) = \frac{1}{3}m$ and $\alpha = \gamma = \frac{1}{3}$. For π_m equal to the canonical reading order, it is $p_0 = 1$, and we have $p_1 = \max_{0 \le i \le 117}\{|\mathrm{AM}_{\pi_m}(\tilde{f}^1, t+i)|\} + 1 = 4$, and $p_2 = \max_{0 \le i \le 95}\{|\mathrm{AM}_{\pi_m}(\tilde{f}^2, t+i)|\} + 1 = 4$. Hence, $p = \frac{1}{3} + \frac{4}{3} + \frac{4}{3} = 3$. Obviously, the consistency test for an observed keystream can be performed by a π_m-OBDD Q_m with $w(Q_m) \le 2^3 = 8$ according to Lemma 5.22. Since new bits are utilized in the computation of each keybit, we can expect the Independence Assumption to hold.

Hence, the application of Theorem 5.18 yields

Theorem 5.36. *The secret initial state of the Grain automaton can be recovered from the first n keystream bits in time and with space in $\mathcal{O}\left(n \cdot 2^{0.6n}\right) \approx 2^{154}$ for $n = 256$.*

Compared to the attacks listed in Section 4.2.5, although far from the effort required by exhaustive key search, Theorem 5.36 is to the best of our knowledge the first exploitable cryptanalytic result under realistic assumptions besides generic time-memory-data-tradeoff attacks as presented by Biryukov and Shamir (2000).

5.5.6 The F-FCSR Stream Cipher Family

The F-FCSR stream cipher family in its current version consists of the variants F-FCSR-H and F-FCSR-16 (see Section 4.2.6).

F-FCSR-H has key length 80 bits and consists of a single Galois FCSR M of length $n = 160$ and a feedback tap vector d of Hamming weight 83. Memory cells are only present at those 82 positions $i \in \{1, \ldots, n-1\}$, for which $d_i = 1$. At each clock, eight keystream bits b_i are created by taking the XOR-sum of up to 15 variables of the current internal state (cf. Section 4.2.6 for details).

In order to mount the BDD-attack, we split the FCSR into the main register R^0 and the carry register R^1. Since each keystream bit is computed as the sum of up to 15 internal bits, we are in a similar situation as described in Example 5.19 and need additional LFSRs $R^2, \ldots, R^9$ to compute the keystream bits z_i, $0 \leq i < 8$. The modified keystream function simply returns these bits in each clock. With $l := \mathrm{wt}(d) - 1$ we obtain eight keystream bits from $2l + 8$ internal bits, hence $\beta(m) = \frac{8}{2l+1}m$ and $\alpha = \gamma = \frac{8}{2l+8} = \frac{2}{43}$. We have $p_0 = p_1 = l$, $p_i = 1$ for $i \in \{2, \ldots, 9\}$, $s_0 = s_1 = \frac{l}{2l+8}$, and $s_i = \frac{1}{2l+8}$ for $2 \leq i \leq 9$, which implies $p = \frac{2l^2+1}{2l+1}$.

Obviously, the consistency test for the observed keystream z can be performed by an OBDD Q_m with $w(Q_m) \leq 2$. Since the computation of the keybits involves new internal bits in every clock, we can expect the Independence Assumption to hold. Note that we have l additional unknowns from the initial value of the carry register.

Plugging the computed values into the statement of Lemma 5.17 implies the following theorem.

Theorem 5.37. *The secret initial state of the F-FCSR-H automaton can be recovered from the first $n+l$ keystream bits in time and with space in $\mathcal{O}\left(n \cdot 2^{0.9925(n+l)}\right) \approx 2^{241}$ for $n = 160$ and $l = 82$.*

The F-FCSR-16 generator has the same structure as F-FCSR-H, but larger parameters. More precisely, we have key length 128 bits, $n = 256$ and the feedback tap vector has Hamming weight 131 (i.e., $l = 130$), where memory cells are only present at nonzero tap positions as before. Since F-FCSR-16 produces 16 keystream bits per clock, we construct 16 additional LFSRs that produce these bits. Hence, we can compute 16 keystream bits from $2l + 16$ internal bits, which implies $\beta(m) = \frac{16}{2l+16}m$ and $\alpha = \gamma = \frac{16}{2l+16} = \frac{4}{69}$. Analogously to the case of F-FCSR-H, we obtain $p = \frac{2l^2+16}{2l+16}$. The modified generator satisfies the Independence Assumption as before, and we have $l = 130$ additional unknowns.

We obtain by applying Lemma 5.17:

Theorem 5.38. *The secret initial state of the F-FCSR-16 automaton can be recovered from the first $n+l$ keystream bits in time and with space in $\mathcal{O}\left(n \cdot 2^{0.94(n+l)}\right) \approx 2^{363}$ for $n = 256$ and $q' = 8$.*

Our analysis supports the security requirement that the Hamming weight of c should not be too small, which Arnault et al. (2006) motivated by completely different arguments. Although the BDD-attack is to the best of our knowledge the first nontrivial attack on the current version of the F-FCSR family, its efficiency is by no means close to exhaustive key search.

5.6 Divide-and-Conquer Strategies (DCS)

One obvious disadvantage of BDD-based attacks is their high memory consumption, which is essentially determined by the size of the intermediate BDDs P_m. One possible approach are divide-and-conquer strategies (DCS) that divide the search space, i.e., the set $\{0,1\}^m$ of internal bitstreams of length m, into segments and to apply BDD-based attacks to the segments individually.

We represent a segmentation of $\{0,1\}^m$ by a function $\theta_m : \{0,1\}^m \to \{0,1\}^*$ assigning a segment to each internal bitstream. The number of different segments is then given by $|\mathrm{im}(\theta_m)|$. We denote by θ_m^ς the characteristic function of segment $\varsigma \in \mathrm{im}(\theta_m)$, i.e.,

$$\begin{aligned} \theta_m^\varsigma : \{0,1\}^m &\to \{0,1\} \\ w &\mapsto \begin{cases} 1 & \text{if } \theta_m(w) = \varsigma \\ 0 & \text{otherwise} \end{cases} . \end{aligned}$$

Consequently, we denote by Θ_m^ς the G_m^0-FBDD representing θ_m^ς, i.e., the G_m^0-FBDD that accepts exactly those $w \in \{0,1\}^m$ that satisfy $\theta_m^\varsigma(w) = 1$. Based on a segmentation θ_m, we can perform the BDD-based attack as outlined in Algorithm 11.

Algorithm 11 RecoverInitialState-DCS

```
for all ς ∈ im(θ_m) do
  P^ς = Q_{n_min} ∧ Θ^ς_{n_min}
  for m = n_min + 1 to ⌈α^{-1} n⌉ do
        P^ς = MIN(P^ς ∧ Q_m ∧ S_m ∧ Θ^ς_m)
  end for
  if (P^ς)^{-1}(1) ≠ {} then
    return the initial state bits contained in one of the w ∈ (P^ς)^{-1}(1)
  end if
end for
```

In the same way as the original attack described in Algorithm 5, the performance of the DCS-based attack fundamentally depends on the size of the intermediate FBDDs P_m^ς. Compared to the FBDDs P_m in the original attack, P_m^ς satisfies $|(P_m^\varsigma)^{-1}(1)| \leq |(P_m)^{-1}(1)|$ due to its construction. Hence,

$$m \cdot |(P_m^\varsigma)^{-1}(1)| \leq m \cdot |(P_m)^{-1}(1)| \ .$$

On the other hand, the construction of P_m^ς implies

$$|G_m^0| \cdot |Q_m| \cdot |R_m| \cdot |\Theta_m^\varsigma| \geq |G_m^0| \cdot |Q_m| \cdot |R_m| \ ,$$

i.e., one of the two bounds that we have used to estimate $|P_m|$ in the proof of Lemma 5.17 decreases and the other one increases. We conclude that in general, we will only benefit from a divide-and-conquer strategy if $|\Theta_m^\varsigma|$ is small (preferrably polynomial in m) and $|\mathrm{im}(\theta_m)|$, the number of segments, is not too large, as we have to apply the attack to each segment instead of only once. However, since a particular internal bitstream belongs to exactly one segment, the sets of internal bitstreams mapped to the same segment are disjoint, and the attacks on the individual segments can be efficiently parallelized.

We now consider the special case of setting constant the bits at certain initial positions in the internal bitstream. If $V \subseteq \mathrm{IP}(m)$, $|V| \leq n$, denotes the set of initial positions to be set constant, we define the restriction of $w \in \{0,1\}^m$ to the bits at the positions in V as $w|_V := (w_{i_1}, \ldots, w_{i_{|V|}})$ with $i_j \in V$, $i_j < i_k$ for $j < k$, and the corresponding segmentation function as

$$\begin{aligned} \theta_{m,V} : \{0,1\}^m &\to \{0,1\}^{|V|} \\ w &\mapsto w|_V \end{aligned} .$$

We call a position $j \geq 1$ a V-*determined* position in $w = (w_0, \ldots, w_{m-1})$ if $j \in V \cup \{0, \ldots, m-1\}$ or if the value of w_j is determined by the bits w_l, $l \in V$.

We note that $|\mathrm{im}(\theta_{m,V})| = 2^{|V|}$ and that the construction of the G_m^0-FBDD $\Theta_{m,V}^{\varsigma}$ is trivial, while its size is bounded by $|\Theta_{m,V}^{\varsigma}| \in \mathcal{O}(|V|)$. The attack on an individual segment $\varsigma \in \{0,1\}^{|V|}$ only needs to regard the bits at the $n - |V|$ non-constant initial positions as unknowns, and in the worst case, there are no V-determined positions except for those in V. The DCS-based attack on a single segment therefore corresponds (in the worst case) to the original attack with reduced key length. Following the lines of the proof of Lemma 5.17 we straightforwardly obtain

Theorem 5.39. *For an FSR-based keystream generator fulfilling the requirements of Lemma 5.17 and a divide-and-conquer strategy based on setting constant the bits at initial positions $j \in V$, $|V| \leq n$, the divide-and-conquer algorithm (Algorithm 11) will consume time in the order of $\mathcal{O}\left(2^{|V|}\epsilon(n)|Q_n||V|2^{r^*}\right)$ and memory in the order of $\mathcal{O}\left(\epsilon(n)|Q_n||V|2^{r^*}\right)$, with $r^* := \frac{p(1-\alpha)}{p+\alpha}(n - |V|)$.*

5.6.1 DCS for regularly clocked (k,l)-Combiners

Based on Theorem 5.39, we analyze two particular choices for V that are applicable to regularly clocked (k,l)-combiners, e.g., the E_0 keystream generator.

First, we define V to contain exactly the positions of the first s initial bits of each FSR. In the worst case, there are no V-determined positions besides the positions in V.

For a segment $\varsigma \in \mathrm{im}(\theta_{m,V})$, a BDD-based search of the corresponding segment requires as much effort as the original BDD-attack on a (k,l)-combiner of key length $(n-ks)$. We therefore obtain $r^* = \frac{k-1}{k+1}(n-ks)$ and the exponential part of the overall runtime becomes

$$2^{ks+\frac{k-1}{k+1}(n-ks)} = 2^{\frac{k-1}{k+1}n+\frac{2k}{k+1}s} ,$$

which is by a factor of $2^{\frac{2k}{k+1}s}$ worse than in the original attack. On the other hand, the required memory is reduced by a factor of $2^{\frac{k-1}{k+1}ks}$.

We note that we only need to consider the assignments to the positions in V that are consistent with z. If the combination of the first s initial bits in each register determines the values of the first s keystream bits (E_0 has this property, for instance), Lemma 5.15 implies that it is sufficient to consider around $|\{0,1\}^{(1-\alpha)ks}| = 2^{(k-1)s}$ of the $|\mathrm{im}(\theta_{m,V})| = 2^{ks}$ possible segments, which makes the runtime decrease by a factor of 2^s to

$$2^{\frac{k-1}{k+1}n+\frac{k-1}{k+1}s} .$$

Lemma 5.40. *For a regularly clocked (k,l)-combiner fulfilling the requirements of Lemma 5.17 and the divide-and-conquer strategy of setting constant the first s bits produced by each FSR, the divide-and-conquer algorithm (Algorithm 11) will consume time in the order of $\mathcal{O}\left(\epsilon(n)|Q_n| \cdot ks \cdot 2^{\frac{k-1}{k+1}(n+s)+s}\right)$ and memory in the order of $\mathcal{O}\left(\epsilon(n)|Q_n| \cdot ks \cdot 2^{\frac{k-1}{k+1}(n-ks)}\right)$. If the combiner always produces s keystream bits from the registers' first s initial bits, the runtime decreases by a factor of 2^s.*

The E_0 keystream generator is a regularly clocked $(4,4)$-combiner, which implies

Corollary 5.41. *For the E_0 keystream generator with key length $n = 128$, choosing V to contain the positions of the first s initial bits of each LFSR yields a runtime of the DCS-based attack of $2^{0.6(128+s)}$ polynomial time operations and a memory consumption in the order of $2^{0.6(128-4s)}$.*

As a second example, we choose as V the set of all initial positions that belong to the shortest FSR, w.l.o.g. the FSR R^0. If we denote by $n_0 \leq \frac{n}{k}$ the length of R^0, $\{0,1\}^{n_0}$ is the set of all possible initial states of R^0. Since every k-th position of an internal bitstream w is V-determined, the attack on a particular segment corresponds to the performance of the original BDD-attack on a $(k-1,l)$-combiner of key length $n-n_0$, hence $r^* = \frac{k-2}{k}(n-n_0)$. It is easy to see that for $n_0 \leq \frac{n}{k+1}$, we have

$$|V| + r^* = n_0 + \frac{k-2}{k}(n-n_0) \leq \frac{k-1}{k+1}n \ ,$$

which means that for sufficiently small n_0, we even obtain a runtime improvement in addition to the significantly reduced space consumption.

Lemma 5.42. *For a regularly clocked (k,l)-combiner fulfilling the requirements of Lemma 5.17, n_0 the length of the shortest FSR, and the divide-and-conquer strategy of setting constant the shortest FSR, the divide-and-conquer algorithm (Algorithm 11) will consume time in the order of $\mathcal{O}\left(\epsilon(n)|Q_n| \cdot n_0 \cdot 2^{n_0+\frac{k-2}{k}(n-n_0)}\right)$ and memory in the order of $\mathcal{O}\left(\epsilon(n)|Q_n| \cdot n_0 \cdot 2^{\frac{k-2}{k}(n-n_0)}\right)$.*

In the case of the E_0 keystream generator, we have $n_0 = 25 \leq 25.6 = \frac{128}{4+1}$ and obtain

Corollary 5.43. *For the E_0 keystream generator with key length $n = 128$, choosing V to be the set of all initial positions that belong to the LFSR of length $n_0 = 25$ (the shortest LFSR) yields a runtime of the DCS-based attack of $2^{25+\frac{1}{2}103} = 2^{76.5}$ polynomial-time operations and a memory consumption in the order of $2^{51.5}$.*

Compared to the original BDD-attack, we have improved the memory consumption by a factor of about 2^{25} and the runtime by a factor of $2^{0.3}$.

Shaked and Wool (2006) set constant the last parts of the LFSRs in E_0 (60 bits in total) and thereby lowered the memory requirements to 2^{23} while increasing the runtime to the order of 2^{83}.

5.6.2 DCS for the A5/1 Generator

In the following, we compute the information rate of the A5/1 generator with respect to a family of choices for the set V, particularly those defined by setting constant the initial states of one or more LFSRs. As stated in Section 5.5.3, in the unmodified definition of the A5/1 generator, each of the three LFSRs is divided into two, approximately equally long halfs, a value-half consisting of the output cell and the cells between output and clock-control cell and a control half consisting of the clock-control cell and the rest of the register. Since the value-LFSRs and the control-LFSRs in the modified setting correspond to the value-halfs and the control-halfs in the unmodified case, setting constant the initial states of LFSRs or half-LFSRs in the original definition is equivalent to fixing the corresponding LFSRs in the modified case.

For all natural numbers $i \geq 1$, we denote by Z_i and W_i the random variables corresponding to the i-th keystream bit and the number of internal bits processed for the production of the i-th keystream bit, respectively, taken over the probability space of all random internal bitstreams. In all cases, Z_i and W_i will fulfill the following conditions.

- For all $i > 1$, W_i is independent of $W_1, \ldots, W_{i-1}$, and Z_i is independent of $Z_1, \ldots, Z_{i-1}$.
- $Pr[Z_i = 0] = Pr[Z_i = 1] = \frac{1}{2}$.
- There are natural numbers $a > b > c$ and probabilities p, q and $r = 1-p-q$ such that $Pr[W_i = a] = p$, $Pr[W_i = b] = q$, and $Pr[W_i = c] = r$.

We denote the situation that Z_i and W_i fulfill the above conditions as case $[(p, a), (q, b), (r, c)]$. It can be easily checked that the unrestricted A5/1 generator corresponds to case $[(1/4, 6), (3/4, 4), (0, 0)]$. We will see below that all generators derived from the A5/1 generator by setting constant one or more of the six LFSRs correspond to $[(p, a), (q, b), (r, c)]$ for some p, q, r, a, b, c. We may then compute the information rate α with the help of the following Theorem.

Theorem 5.44. *In the case $[(p, a), (q, b), (r, c)]$, the information rate equals α, where $t = 2^{\alpha}$ is the unique positive real solution of $pt^a + qt^b + rt^c - 2 = 0$.*

Note that for the special case $[(1, k), 0, 0]$ the information rate is $1/k$.

In order to prove Theorem 5.44, we need the following technical result.

Lemma 5.45 (Krause (2002)). *All natural numbers $N \geq 1$, probabilities $p \in (0, 1)$ and real numbers $\beta > 0$ satisfy $\sum_{i=0}^{N} \binom{N}{i} p^i (1-p)^{N-i} 2^{\beta i} = \left(1 - p + p2^{\beta}\right)^N$.*

Proof (Theorem 5.44). Since we can obtain the information rate α from $\alpha = -\frac{1}{m} \log_2 p_C(m)$ following Assumption 4.4, we now compute the probability $p_C(m) = \Pr_w[C_m(w) \text{ is prefix of } z]$ for the cases that parts of the LFSRs are set constant.

Case $[(p, a), (q, b), (r, c)]$ implies that from all random internal bitstreams of length m, m divisible by a, at least m/a keystream bits are produced. The number of internal bits remaining from m internal bits after the production of m/a keystream bits can be computed as

$$m - aU - bV - c\left(\frac{m}{a} - U - V\right) = \frac{a-c}{a} m - (a-c)U - (b-c)V \ ,$$

where U and V denote the number of keystream bits among the first m/a keystream bits for which a and b internal bits are processed, respectively. Note that U is $(p, m/a)$-binomially distributed and that V, under the condition that $U = i$, is $(q/(q+r), m/a - i)$-binomially distributed. We obtain the following relation for $p_C(m)$.

$$p_C(m) = 2^{-\frac{m}{a}} \sum_{i=0}^{\frac{m}{a}} \sum_{j=0}^{\frac{m}{a}-i} Pr[U = i, V = j] p\left(\frac{a-c}{a} m - (a-c)i - (b-c)j\right) \quad \text{, i.e.,}$$

$$\begin{aligned} 2^{-\alpha m} &= 2^{-\frac{m}{a}} \sum_{i=0}^{\frac{m}{a}} \binom{\frac{m}{a}}{i} p^i (1-p)^{\frac{m}{a}-i} \\ &\quad \cdot \sum_{j=0}^{\frac{m}{a}-i} \binom{\frac{m}{a}-i}{j} \left(\frac{q}{q+r}\right)^j \left(\frac{r}{q+r}\right)^{\frac{m}{a}-i-j} \cdot 2^{-\alpha(\frac{a-c}{a}m-(a-c)i-(b-c)j)} \quad , \end{aligned}$$

which is equivalent to

$$\begin{aligned} 2^{(1-a\alpha+(a-c)\alpha)\frac{m}{a}} &= \sum_{i=0}^{\frac{m}{a}} \binom{\frac{m}{a}}{i} p^i (1-p)^{\frac{m}{a}-i} \cdot 2^{(a-c)\alpha i} \\ &\quad \cdot \sum_{j=0}^{\frac{m}{a}-i} \binom{\frac{m}{a}-i}{j} \left(\frac{q}{1-p}\right)^j \left(\frac{r}{1-p}\right)^{\frac{m}{a}-i-j} \cdot 2^{(b-c)\alpha j} \quad . \end{aligned}$$

Now, we apply Lemma 5.45 to the inner sum and obtain

$$2^{(1-n\alpha)\frac{m}{a}} = \sum_{i=0}^{\frac{m}{a}} \binom{\frac{m}{a}}{i} p^i (1-p)^{\frac{m}{a}-i} \cdot 2^{(a-c)\alpha i} \cdot \left(\frac{r}{1-p} + \frac{q}{1-p} 2^{(b-c)\alpha}\right)^{\frac{m}{a}-i} .$$

Setting $s := \frac{r}{1-p} + \frac{q}{1-p} 2^{(b-c)\alpha}$, we get

$$\begin{aligned} \left(\frac{2}{s 2^{c\alpha}}\right)^{\frac{m}{a}} &= \sum_{i=0}^{\frac{m}{a}} \binom{\frac{m}{a}}{i} p^i (1-p)^{\frac{m}{a}-i} \cdot 2^{((a-c)\alpha - \log(s))i} \\ &= \left(1 - p + p 2^{(a-c)\alpha - \log(s)}\right)^{\frac{m}{a}} . \end{aligned}$$

Consequently, we obtain by setting $t := 2^\alpha$

$$\frac{2}{s t^c} = 1 - p + p \frac{t^{a-c}}{s} \quad \Leftrightarrow \quad 2 = (1-p) s t^c + p t^a \quad .$$

$s = \frac{r}{1-p} + \frac{q}{1-p} t^{b-c}$ implies $2 = r t^c + q t^b + p t^a$, which in turn implies the claim. □

We now compute the information rates for restrictions of type $(v_1 v_2 v_3 | c_1 c_2 c_3) \in \{0,1\}^6$, which means that those output LFSRs i in the modified version of the generator for which $v_i = 1$ and the control LFSRs j for which $c_j = 1$ are set constant. Note that the unrestricted case corresponds to (000|000). We do not

consider the case of 5 constant LFSRs, since the initial state of the remaining unknown LFSR from a given keystream can be computed in linear time.

For symmetry reasons, certain choices for $(v_1v_2v_3|c_1c_2c_3)$ are equivalent. First, it is easy to see that for all permutations π_m of $\{1,2,3\}$, restriction $(v_1v_2v_3|c_1c_2c_3)$ is equivalent to restiction $(v_{\pi_m(1)}, v_{\pi_m(2)}v_{\pi_m(3)}|c_{\pi_m(1)}c_{\pi_m(2)}c_{\pi_m(3)})$. Furthermore, we observe that with respect to restriction $(v|c)$, $v, c \in \{0,1\}^3$, the number of internal bits $W(u,V,C)$ processed for the production of the next keystream bit assuming the current values in the control LFSRs are $u \in \{0,1\}^3$ equals

$$W(u,v,c) = \sum_{i,c_i=0} f_i(u) + \sum_{i,v_i=0} f_i(u) \ , \tag{5.3}$$

where for $i \in \{1,2,3\}$ the Boolean function $f_i : \{0,1\}^3 \to \{0,1\}$ is defined to output 1 on u iff the i-th LFSR will be clocked w.r.t. u, more precisely

$$f_i(u) = (u_i \oplus u_{((i+1) \bmod 3)} \oplus 1) \vee (u_i \oplus u_{((i+2) \bmod 3)} \oplus 1) \ .$$

Equation (5.3) implies that for all $v,c,u \in \{0,1\}^3$ and $i \in \{1,2,3\}$, $W(u,v,c) = W(u,v',c')$, where v', c' are obtained from v, c by exchanging the i-th component. Hence, restriction $(v|c)$ is equivalent to restriction $(v'|c')$. It is therefore sufficient to analyze the restrictions (000|100), (100|100), (100|010), (100|110), (000|111), (100|111), and (110|110).

We first consider the restriction (100|100). If the actual content of the output cells of the two non-constant control LFSRs is 00 or 11, then four internal bits will be processed, otherwise two internal bits will be processed. Hence, the corresponding case is $[(1/2,4),(1/2,2),0]$ and therefore $\alpha \approx 0.3215$.

Under restriction (100|010), four internal bits will be processed if the actual content of the output cell of the constant control LFSR is $b \in \{0,1\}$ and the actual content of the two non-constant control LFSR is bb. If we have $b\bar{b}$ then two, and in all remaining cases 3 internal bits will be processed. Therefore, we are in the case $[(1/4,4),(1/2,3),(1/4,2)]$ and obtain $\alpha \approx 0.3271$.

Under restriction (110|110), two internal bits will be processed if the assignment to the output cells of the constant control LFSRs is 01 or 10 or if all three output cells of the control LFSRs coincide. If the assignment to the output cells of the constant control LFSRs is bb for some $b \in \{0,1\}$ and the random assignment to the remaining control cell is $\bar{b}$, then the next keystream bit depends only on the constant assignments, and no internal bit will be processed. Hence, in contrast to the above cases, $p_C(m)$ and α are not independent of the constant LFSRs and the given keystream. Therefore, we compute only the average information rate over all possible assignments to the constant control and output LFSRs. According to the above observation, the probability that two internal bits are processed for the next keystream bit is 3/4, and the probability that 0 internal bits are processed for the next ouput bit is 1/4. In total, we obtain $[(3/4,4),(1/4,0),0]$ and therefore $\alpha \approx 0.6113$.

We can handle the remaining cases with similar arguments.

The information rates for the discussed cases are summarized in Table 5.1.

Lemma 5.46. *For the A5/1 keystream generator and the devide-and-conquer strategy of setting constant particular sub-LFSRs as indicated in Table 5.1, the divide-and-conquer algorithm (Algorithm 11) will consume time in the order of* $\mathcal{O}\left(2^{|V|} \cdot n^{11} \cdot 2^{r^*}\right)$ *and memory in the order of* $\mathcal{O}\left(n^{11} \cdot 2^{r^*}\right)$ *with* $r^* = \frac{1-\alpha}{1+\alpha}n$.

Table 5.1: Information rates α for the restricted A5/1

$\lvert V\rvert$	restriction	α	r^*	$\lvert V\rvert + r^*$
$\frac{2}{3}n$	(100|111)	0.6430	$0.2173n$	$0.8840n$
	(110|110)	0.6113	$0.2412n$	$0.9079n$
$\frac{1}{2}n$	(000|111)	0.4386	$0.3902n$	$0.8902n$
	(100|110)	0.4261	$0.4024n$	$0.9024n$
$\frac{1}{3}n$	(000|110)	0.3271	$0.5070n$	$0.8403n$
	(100|100)	0.3215	$0.5134n$	$0.8467n$
$\frac{1}{6}n$	(000|100)	0.2622	$0.5840n$	$0.7507n$
0	(000|000)	0.2193	$0.6403n$	$0.6403n$

5.7 Simulations and Experimental Results

In order to provide a fast implementation of the FBDD algorithms, the FBDD-library developed by Stegemann (2004) based on the publicly available OBDD package CUDD (Somenzi, 2001) was extended to support divide-and-conquer strategies. We used this library for our experiments on a standard Linux PC with a 2.7 GHz Intel Xeon processor and 4 GB of RAM. All implementations were done in C using the gcc-compiler.

Since the runtime of the cryptanalysis fundamentally depends on the maximum size of the intermediate FBDDs P_m, we investigate how much experimentally obtained values of $|P_m|$ deviate from the theoretical figures.

We first consider the basic BDD-based attack. For the self-shrinking generator, the E_0 generator and the A5/1 generator, we analyzed several thousands of reduced instances with random primitive feedback polynomials and random initial states for various key lengths. For each considered random generator, we computed the actual maximum BDD-size of the intermediate results

$$P_{\max}(n) = \max_{1 \le m \le \lceil \alpha^{-1} n \rceil} \{|P_m|\} \quad ,$$

the theoretical upper bound

$$P^t_{\max}(n) = \max_{1 \le m \le \lceil \alpha^{-1} n \rceil} \left\{ \epsilon(m) \cdot |Q_m| \cdot 2^{\frac{p(1-\alpha)}{p+\alpha} n} \right\}$$

that was obtained in Lemma 5.17, as well as the quotient $q(n) = \frac{\log(P_{\max}(n))}{\log(P^t_{\max}(n))}$.

Similarly, we tested for E_0 and A5/1 the divide-and-conquer strategy of setting constant the shortest LFSR (denoted by strategy s1), and we considered fixing the first $s = \lfloor \frac{n_0}{2} \rfloor \le \frac{n}{8}$ bits of each of the four LFSRs in E_0 (denoted by strategy s2), with n_0 the length of the shortest LFSR. Note that $(s1)$ corresponds to the case (100|100) for the A5/1 generator. Since the $q(n)$-values did not noticeably decrease with increasing n in all our simulations, we estimate the attack's performance in dependence of n by multiplying the theoretical figures by $2^{q(n)}$. Particularly, we can obtain conjectures about the attack's performance on real-life instances of E_0 and A5/1 by replacing n with the actual key lengths. Tables 5.2 and 5.3 shows the results of these computations along with details about our experiments. We observe that our results are consistent with an

earlier analysis of the basic BDD-based attack on E_0 and the self-shrinking generator which was conducted by Schleer (2002).

On average, the attack based on DCS (s1) took 87 minutes for E_0 with $n = 37$ and 54 minutes for A5/1 with $n = 30$. The longest key lengths that we were able to tackle with the resources described at the beginning of this section were $n = 46$ for E_0 and $n = 37$ for A5/1. These attacks used up almost all of the available memory and took 60.5 and 25.1 hours to complete on average.

5.8 Discussion of the BDD-Attack

We observe that when considering only the keystream generator (without the key/IV setup procedure), the BDD-Attack is an efficient generic initial state recovery attack that is faster than exhaustive search for a broad class of stream ciphers. This leads straightforwardly to an efficient attack for older designs like E_0 and A5/1, whose key length is roughly equal to the internal state size.

From the extended BDD-Attack, e.g. on TRIVIUM, we observe that the BDD approach may still be applied to recover the initial internal state of a keystream generator under the following generalizations.

1. Instead of LFSRs, feedback shift registers with nonlinear update functions are used. This is in contrast to algebraic attacks (to be discussed in the next chapter), which fundamentally depend on the linearity of the update function. The only requirement that we have is that the update relation be balanced.

2. The shift registers influence each other via exchanging update bits.

However, since the adaption of time/memory/data tradeoffs to stream ciphers by Biryukov and Shamir (2000), modern designs incorporate keystream generators whose size is at least twice the key length. Hence, for the BDD-Attack in its current form to yield an actual attack on the whole cipher (including the key/IV setup), we would need $\frac{p(1-\alpha)}{p+\alpha} < \frac{1}{2}$, which means $\alpha > \frac{1}{3}$ for $p = 1$. This will rarely be the case for practical designs.

Hence, the generic nature of the BDD-Attack is at the same time its drawback: Currently, we cannot make use of any IV-knowledge and we see no way to efficiently tacke the key/IV setup (usually performing many operations without producing observable output) with our method. Similarly, it seems hard to

Table 5.2: Simulation parameters of the BDD-based attack

generator	DCS	key length interval	avg $q(n)$	no. of samples
E_0	–	$[19, 37]$	0.85	2000
E_0	s1	$[19, 37]$	0.95	2700
E_0	s2	$[19, 37]$	0.9	2700
A5/1	–	$[15, 30]$	0.9	3000
A5/1	s1	$[19, 37]$	0.77	2400
SSG	–	$[10, 35]$	0.8	3300

Table 5.3: Performance of the BDD-based attack in practice

generator	DCS	estimated practical performance			
		Time		Space	
E_0	–	$2^{0.51n}$	$2^{65.28}$	$2^{0.51n}$	$2^{65.28}$
E_0	s1	$2^{0.475(n+n_0)}$	$2^{72.68}$	$2^{0.475(n-n_0)}$	$2^{48.93}$
E_0	s2	$2^{0.54n+0.27n_0}$	$2^{75.87}$	$2^{0.54n-1.08n_0}$	$2^{42.12}$
A5/1	–	$2^{0.5763n}$	$2^{36.88}$	$2^{0.5763n}$	$2^{36.88}$
A5/1	s1	$2^{0.3953n+0.77n_0}$	$2^{39.93}$	$2^{0.3953n}$	$2^{25.30}$
SSG	–	$2^{0.525n}$		$2^{0.525n}$	

incorporate the specialities in the cipher operations that are heavily exploited by other, more specific attacks.

It therefore remains as an open question whether the BDD-Approach can be combined with other strategies (e.g., correlation attacks and algebraic attacks which are to be considered in the next chapter) in order to obtain attacks on modern designs whose internal state is much larger than the secret key.

Chapter 6

Other Generic Attacks on Stream Ciphers

In this section, we consider two other prominent generic attacks on stream ciphers – correlation attacks and algebraic attacks.

A correlation attack consists of finding and exploiting linear functions

$$L(X_t, \ldots, X_{t+r-1}, z_t, \ldots, z_{t+r-1})$$

which are biased, i.e., equal to zero with some probability $\neq 1/2$. Algebraic attacks, in a way, mark the opposite. Here, non-linear equations of preferably low degree that are true with probability one are used to describe the secret information by a system of equations.

The basic ideas of these attacks have been known for quite a few years. The first appearance of correlation attacks dates back to the mid-80s (Siegenthaler, 1985), while algebraic attacks have been discovered around the year 2003 (Armknecht and Krause, 2003, Courtois, 2003, Courtois and Meier, 2003).

In this thesis, we focus on particular variants of correlation attacks and algebraic attacks on LFSR-based combiners with memory, which we describe in Sections 6.1 and 6.2, respectively. We indicate ways to reduce the efficiency of these attacks in Section 6.3 and apply our findings in Section 6.4 to improve the security of the Bluetooth keystream generator E_0 by relatively small modifications of the original design.

6.1 Correlation Attacks

6.1.1 The Basic Idea

Inspired by Zenner (2004), we first describe the basic ideas behind correlation attacks.

Definition 6.1. *We define the* bias $\lambda(X)$ *of a binary random variable* X *as*

$$\lambda(X) := \Pr[X = 0] - \Pr[X = 1] = \mathbb{E}[(-1)^X]$$

and the correlation *between two random variables* X *and* Y *as* $\lambda(X \oplus Y)$*. We call* X unbiased *if* $\lambda(X) = 0$*, and we say that* X *and* Y uncorrelated *if* $\lambda(X \oplus Y) = 0$*.*

For $p_e(X,Y)$ defined as

$$p_e(X,Y) := \Pr[X \oplus Y = 0] = \Pr[X = Y] \ ,$$

we have $\lambda(X \oplus Y) = 2p_e(X,Y) - 1$, or equivalently $p_e(X,Y) = \frac{1}{2} + \frac{\lambda(X \oplus Y)}{2}$. Note that X and Y are uncorrelated if and only if $p_e(X,Y) = \frac{1}{2}$.

We first consider the case of combination generators without memory consisting of k FSRs $R^1, \ldots, R^{k-1}$ and a keystream function $C : \{0,1\}^k \to \{0,1\}$ (see Section 4.1.1).

The fundamental property that correlation attacks are based on is that the keystream $(z_t)_{t \geq 0}$ and the bitstream $(w_t^j)_{t \geq 0}$ produced by FSR R^j, $j \in \{0, \ldots, k-1\}$ are correlated, more precisely

$$\Pr[z_t \oplus w_t^j = 0] = \Pr[z_t = w_t^j] = p_e(z_t, w_t^j) = \lambda' \neq \frac{1}{2} \text{ for all } t \ . \tag{6.1}$$

The original correlation attack proposed by Siegenthaler (1985) then proceeds as follows.

1. Make a guess $\tilde{\omega}^j(0)$ for the initial state $\omega^j(0)$ of R^j and compute from $\tilde{\omega}^j(0)$ the sequence $(\tilde{w}_t^j)_{t \geq 0}$.

2. For a suitably chosen n, compute the sum

$$\tilde{D} := \sum_{i=0}^{n-1} (\tilde{w}_i^j \oplus z_i)$$

over the integers.

We now distinguish two cases. If the guess in step (1) was correct, $\tilde{D}$ is (n, λ')-binomially distributed with expected value $\mu = \lambda' n$ and variance $\sigma^2 = n\lambda'(1 - \lambda')$. On the other hand, if the guess was wrong, $(z_t \oplus \tilde{w}_t^j)_{t \geq 0}$ behaves like a random sequence, i.e., $\tilde{D}$ is $(n, \frac{1}{2})$-binomially distributed with $\mu = \frac{n}{2}$ and $\sigma^2 = \frac{n}{4}$.

Hence, if we can tell which distribution $\tilde{D}$ was drawn from, we can deduce if our guess $\tilde{\omega}^j(0)$ was correct. A straightforward approach is to set a threshold D' and to accept $\tilde{\omega}^j(0)$ if $\tilde{D} > D'$. Otherwise, we assume the guessed initial state was wrong and we try the next one.

With $|R^j|$ denoting the number of cells in register R^j, this method will require a number of steps in the order of $2^{|R^j|}$ for recovering $\omega^j(0)$, and about $2^{\sum_{i \neq j} |R^i|}$ operations for computing the remaining $k-1$ initial states, which adds up to an overall effort of $2^{|R^j|} + 2^{\sum_{i \neq j} |R^i|}$, whereas exhaustive search on the whole initial state of the generator would require $2^{\sum_{i=0}^{k-1} |R^i|} = 2^{|R^j|} \cdot 2^{\sum_{i \neq j} |R^i|}$ operations. The number n of required keystream bits to tell apart the two distributions (with a fixed error probability) depends on the value $|\frac{1}{2} - \lambda'|$, i.e., the absolute distance between λ' and $\frac{1}{2}$, and will shrink as this distance increases.

This strategy can be straightforwardly extended to correlations of linear combinations of FSR output bits and the keystream implied by

$$\Pr[z_t = \oplus_{j=0}^{k-1} \gamma_j w_t^j] \neq \frac{1}{2} \text{ with } \gamma_j \in \{0,1\} \ . \tag{6.2}$$

However, we now have to guess the initial states of all R^j with $\gamma_j = 1$ simultaneously, which leads to an overall effort in the order of

$$2^{\sum_{j=0}^{k-1} \gamma_j |R^j|} + 2^{\sum_{j=0}^{k-1} (\gamma_j \oplus 1) |R^j|} \ .$$

It is interesting to note that if indeed $z_t = \oplus_{j=0}^{k-1} \gamma_j w_t^j$ for a clock cycle t, then the update relations of the FSRs (and sums thereof) continue to hold if we replace the term $\oplus_{j=0}^{k-1} \gamma_j w_t^j$ by z_t. Conversely, if a keystream bit z_t satisfies a large number of such relations, it is reasonable to assume that $\oplus_{j=0}^{k-1} \gamma_j w_t^j = z_t$ and to assume $\oplus_{j=0}^{k-1} \gamma_j w_t^j \neq z_t$ if z_t satisfies only few. In this way, we can obtain a candidate guess for the registers R^j with $\gamma_j \neq 0$, which can be incrementally improved in order to obtain the true values. This idea was proposed and formalized by Meier and Staffelbach as *Fast Correlation Attack* and has constantly been extended and improved since its original publication in 1988 (see, e.g., Hell (2007) as a recent example).

As natural countermeasure against correlation attacks, we would try to use combination functions that induce the lowest possible correlations between the keystream and the FSR bitstreams.

Definition 6.2. *A Boolean function* $g : \{0,1\}^k \to \{0,1\}$ *is said to be* r-th order correlation immune *if no linear function* L *depending on up to* $r < k$ *input variables exists such that* $\Pr[L(x) = g(x)] \neq \frac{1}{2}$.

However, there exist at least two tradeoffs that limit the effect of a correlation immune combination function on the overall security of the generator.

Firstly, Siegenthaler (1984), Xiao and Massey (1988) showed that an increase in correlation immunity leads to a lower linear complexity and vice versa. Hence, the output keystream of a highly correlation immune generator will be efficiently reproducible by an LFSR.

For the second tradeoff, we need the following definition.

Definition 6.3. *Let* $\{L_i | 1 \leq i \leq 2^k\}$ *denote the set of linear functions in up to* k *variables. The* correlation coefficient *between a Boolean function* $g : \{0,1\}^k \to \{0,1\}$ *and* L_i *is defined as* $c_i = 2 \cdot p_i - 1$ *with* $p_i = \Pr[L_i(x) = g(x)]$.

Meier and Staffelbach (1989) observed that

$$\sum_{i=1}^{2^k} c_i^2 = 1 \ , \tag{6.3}$$

i.e., if g is not correlated to any low-weight linear function, it is at the same time even stronger correlated to linear functions with larger weight. Hence, correlations itself can never be prevented.

Rueppel (1986) showed that LFSR-based combiners with memory are able to overcome the tradeoff between correlation immunity and linear complexity, but it turns out that a tradeoff similar to Eq. (6.3) is still possible as soon as correlations are regarded that span several consecutive clock cycles (Golić, 1993, 1996, Lu and Vaudenay, 2005, 2004, Salmasizadeh et al., 1997). This is the setting that we are going to consider in the following.

6.1.2 Analysis of the Special Case $C(x_t, q_t) = \alpha(x_t) \oplus \beta(q_t)$

We focus on the special case of LFSR-based (k, l)-combiners with memory whose keystream function C can be written as the sum of two functions $\alpha : \{0,1\}^k \to \{0,1\}$ and $\beta : \{0,1\}^l \to \{0,1\}$, i.e.,

$$C(x_t, q_t) = \alpha(x_t) \oplus \beta(q_t) \tag{6.4}$$

in the LFSR output bits $x_t = (x_t^0, \ldots, x_t^{k-1})$ and the memory state $q_t = (q_t^0, \ldots, q_t^{l-1})$ at time t. Moreover, we are going to consider only biased linear combinations of $\beta(q_t)$.

Therefore, we look for coefficients $\gamma = (\gamma_0, \ldots, \gamma_{r-1})$ such that

$$\lambda(\gamma) := \left(Pr\left[\bigoplus_{i=0}^{r-1} \gamma_i \cdot \beta(q_{t+i}) = 0\right] - Pr\left[\bigoplus_{i=0}^{r-1} \gamma_i \cdot \beta(q_{t+i}) = 1\right]\right) \neq 0 \ . \tag{6.5}$$

Lu and Vaudenay (2005, 2008) showed that the bias $\lambda(\gamma)$ is related to the correlation of the keystream $(z_t)_{t\geq 0}$ and the sequence $(x_t^0)_{t\geq 0}$ produced by the shortest LFSR (assume R^0 for simplicity) by

$$\Pr\left[\bigoplus_{i=1}^{w} \left(\gamma_0(x_{t+v_i}^0 \oplus z_{t+v_i}) \oplus \ldots \oplus \gamma_{r-1}(x_{t+v_i+r-1}^0 \oplus z_{t+v_i+r-1})\right)\right] = \frac{1}{2} + \frac{(\lambda(\gamma))^w}{2} \ ,$$

with w and $v_1, \ldots, v_w$ depending on the initial state polynomials of the LFSRs. Hence, biased linear combinations of $\beta(q_t)$ imply a vulnerability to a correlation attack, and for the attack to be as efficient as possible, we are interested in coefficient vectors γ yielding

$$\lambda_{\max} := \max\{|\lambda(\gamma)|\} \ .$$

General methods to systematically compute $\lambda_{\max}$ and the corresponding equations exist (e.g., see Golić (1993)), but since their resource consumption is exponential in k, l and r, these methods are only feasible for small parameters.

However, our special case allows for a closed formula for the bias $\lambda(\gamma)$, which we are going to derive in the following.

We assume that for each time $t \geq 1$, there is a separate Boolean function $\beta_t : \{0,1\}^l \times \{0,1\}^k \to \{0,1\}$ revealing information about q_t and x_t and define

$$\begin{array}{rccl} F^r : & \{0,1\}^l \times (\{0,1\}^k)^r & \to & \{0,1\} \\ & (q_1, x_1, \ldots, x_r) & \mapsto & \beta_1(q_1, x_1) \oplus \ldots \oplus \beta_r(q_r, x_r) \ , \end{array}$$

where $q_{t+1} := \delta(q_t, x_t)$.

Definition 6.4. *We define the* bias *of a Boolean function* $f : \{0,1\}^n \to \{0,1\}$ *as*

$$\lambda(f) := \Pr[f(x) = 0] - \Pr[f(x) = 1] \ .$$

We call f unbiased *if* $\lambda(f) = 0$.

Note that if all inputs x are equally likely, we have $\lambda(f) = 2^{-n}\left(|f^{-1}(0)| - |f^{-1}(1)|\right)$.

The value $\lambda(F^r)$, for which we now derive a matrix-based expression, corresponds to Eq. (6.5) after setting $\beta_t(q_t, x_t) := \beta(q_t)$ if $\gamma_t = 1$ and $\beta_t :\equiv 0$ otherwise.

Definition 6.5. *For all states $q, q' \in \{0,1\}^l$, let $p(q,q')$ denote the probability that state q will change into q', i.e., $p(q,q') = 2^{-k}\,|\{x|\delta(q,x) = q'\}|$. Additionally, define*

$$b_t(q,q') := \frac{1}{2^k}\left(|\{x|\beta_t(q,x) = 0 \wedge \delta(q,x) = q'\}| - |\{x|\beta_t(q,x) = 1 \wedge \delta(q,x) = q'\}|\right) \ .$$

We call the matrix $P = (p(q,q'))_{q,q' \in \{0,1\}^l}$ the transition matrix *of the memory update function δ and the matrix $B_t = (b_t(q,q'))_{q,q' \in \{0,1\}^l}$ the* bias matrix *of δ and β_t w.r.t. to time t.*

Theorem 6.6. *For all $r \geq 1$,*

$$\lambda(F^r) = 2^{-l}\left(e^T\right) \circ B_1 \circ \cdots \circ B_r \circ e \ ,$$

where e denotes the constant-1 vector of length 2^l and M^T denotes the transpose of matrix M.

In order to prove Theorem 6.6, we first collect some observations on computing biases.

- For a given finite set S and functions $f, g : S \to \mathbb{R}$ we denote by $(f,g) = \frac{1}{|S|}\sum_{s\in S} f(s)g(s)$ a positive definite scalar product on $\mathbb{R}^S$. Note that each Boolean function $f : \{0,1\}^n \to \{0,1\}$ satisfies $\lambda(f) = \left((-1)^f, 1\right)$.

- Consider two disjoint finite sets S and S', functions $f : S \to \mathbb{R}$ and $g : S' \to \mathbb{R}$, and let $h : S \times S' \to \mathbb{R}$ be defined by $h(s,s') = f(s)g(s')$. Then

$$\begin{aligned}(h,1) &= \frac{1}{|S||S'|}\sum_{s\in S, s'\in S'} f(s)g(s') \\ &= \frac{1}{|S|}\sum_{s\in S} f(s)\frac{1}{|S'|}\sum_{s'\in S'} g(s') \\ &= (f,1)(g,1) \ .\end{aligned}$$

  This implies that for Boolean functions $f : \{0,1\}^n \to \{0,1\}$, $g : \{0,1\}^m \to \{0,1\}$, and $h : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$, defined by $h(s,s') = f(s) \oplus g(s')$, we have $\lambda(h) = \lambda(f) \cdot \lambda(g)$.

Now let us denote by $f^r : \{0,1\}^l \times \left(\{0,1\}^k\right)^r \to \mathbb{R}$ the function $(-1)^{F^r}$. For all $r \geq 1$ and $q \in \{0,1\}^l$, we define an additional function

$$\begin{array}{rccl} f_q^r : & \{0,1\}^l \times \left(\{0,1\}^k\right)^r & \to & \{-1,0,1\} \subseteq \mathbb{R} \\ & (q_1, x_1, \ldots, x_r) & \mapsto & \begin{cases} f^r(q_1, x_1, \cdots, x_r) & \text{if } \delta(q_r, x_r) = q \\ 0 & \text{otherwise} \end{cases} \ , \end{array}$$

with $q_{t+1} = \delta(q_t, x_t)$ for $i \in \{1, \ldots, r-1\}$, and let $\Gamma_q^r = (f_q^r, 1)$ and $\Gamma^r = (\Gamma_q^r)_{q\in\{0,1\}^l}$. Then $f^r = \sum_{q\in\{0,1\}^l} f_q^r$ and $\lambda(F^r) = \sum_{q\in\{0,1\}^l} \Gamma_q^r$.

Theorem 6.6 is now a straightforward consequence of the following Lemma.

Lemma 6.7. *For $q \in \{0,1\}^l$ and $r \geq 1$, the bias matrices B_t from Definition 6.5 satisfy*

$$(\Gamma^r)^T = 2^{-l}(e^T) \circ B_1 \circ \cdots \circ B_r \ .$$

Proof. For all $q, q' \in \{0,1\}^l$ we define

$$g_{q,q'} : \begin{array}{rcl} \{0,1\}^k & \to & \{0,1\} \\ x & \mapsto & \begin{cases} 1 & \text{if } \delta(q,x) = q' \\ 0 & \text{otherwise} \end{cases} \end{array} .$$

Observe that for each $t \geq 1$,

$$((-1)^{\beta_t(q,\cdot)} g_{q,q'}, 1) = b_t(q,q') \ . \tag{6.6}$$

We prove the claim by induction on r. Note that, due to Eq. (6.6), for all $q \in \{0,1\}^l$

$$\Gamma^1_q = 2^{-(k+l)} \sum_{q_1,x_1} (-1)^{\beta_1(q_1,x_1)} g_{q_1,q}(x_1) = 2^{-l} \sum_{q_1} b_t(q_1,q) \ .$$

Consequently, $\Gamma^1 = 2^{-l}(e^T) \circ B_1$. For $r > 1$, the function f^r_q can be written as

$$f^r_q(q_1, x_1, \cdots, x_r) = \sum_{q' \in \{0,1\}^l} f^{r-1}_{q'}(q_1, x_1, \cdots, x_{r-1})(-1)^{\beta_r(q',x_r)} g_{q',q}(x_r) \ .$$

Hence, by Eq. (6.6), we obtain

$$\Gamma^r_q = \sum_{q' \in \{0,1\}^l} \Gamma^{r-1}_{q'} b_r(q',q) \quad \textit{and} \quad (\Gamma^r)^T = (\Gamma^{r-1})^T \circ B_r \ . \qquad \square$$

Note that the formula given by Theorem 6.6 can be efficiently evaluated and therefore permits an exhaustive search for the best correlations even for large values of r up to the length of the shortest LFSR.

6.2 Algebraic Attacks

6.2.1 The Basic Idea

Algebraic attacks (Armknecht and Krause, 2003, Courtois, 2003, Courtois and Meier, 2003) are based on solving systems of equations and were, just like correlation attacks, targeted at LFSR-based combination generators without memory, in our notation consisting of k LFSRs $R^0, \ldots, R^{k-1}$ and a nonlinear keystream function C that produces from the LFSR output $x_t = (x^0_t, \ldots, x^{k-1}_t)$ a keystream bit $z_t = C(x_t)$ in each clock cycle t. At this point, we only describe the basic ideas behind algebraic attacks and refer the interested reader to Armknecht (2006) for a thorough treatment of the subject.

The core of algebraic attacks is to find Boolean functions $F : \{0,1\}^{k \cdot r} \to \{0,1\}$ of low preferably degree such that for all clocks t,

$$F(x_t, \ldots, x_{t+r-1}, z_t, \ldots, z_{t+r-1}) = 0 \ . \tag{6.7}$$

Since the combiner's FSRs are LFSRs by assumption, we can express the bit x^j_t that LFSR R^j produces at time t by a linear function L^j_t in the initial

state $\omega^j(0)$ of R^j as $x_t^j = L_t^j(\omega^j(0))$. Hence, collecting equations of the type of Eq. (6.7) yields a system of equations in the secret initial states of the LFSRs.

However, since the keystream function is non-linear, solving the system and thereby recovering the secret initial state is NP-hard in general, so we should not hope for an efficent generally applicable key recovery algorithm based on this strategy, but we may still be fortunate enough to encounter special cases that are sufficiently easy to solve.

This might especially be the case if the number of known keystream bits and therefore the number of equations increases. Let R denote the number of accessible equations and μ the number of occurring monomials. If $R \ll \mu$, a promising method is to compute Groebner bases.

Unfortunately, it seems hard to predict the required time effort, albeit simulations indicate that the neccessary amount of time drops with increasing number of equations (Armknecht and Ars, 2009, Faugère and Ars, 2003).

In the case of $R \approx \mu$, linearization (Courtois et al., 2000) seems to be the first choice. The idea of linearization is to substitute each occurring monomial by a new variable and to treat the whole system as a system of linear equations, making it easily solvable by Gaussian elimination.

For the case that the number of equations exceeds the number of monomials, one might reduce the degree of the equations in a precomputation step. This idea is known as *fast algebraic attacks*, which have been introduced by Courtois (2003) and further improved by, e.g., Armknecht (2004a), Hawkes and Rose (2004). However, the attack scenario is more restrictive as it requires the attacker to know many successive keystream bits and Eq. (6.7) to have a special structure.

All theses approaches have in common that their runtime strongly depends on the degree d of the incorporated equations. The lower the degree, the faster the attacks. Hence, a natural countermeasure against such attacks is to prevent the existence of low-degree equations.

6.2.2 Analysis of a restricted Scenario

For our analysis, we will concentrate on algebraic attacks where $R \approx \mu$ and μ is approximately $\binom{n}{d}$. If φ denotes the number of functions F of degree d fulfilling Eq. (6.7) and n denotes the total length of the LFSRs, then the amount of data is $\approx \binom{n}{d}/\varphi$, and the required memory and runtime are in $O\left(\binom{n}{d}^2\right)$ and $O\left(\binom{n}{d}^3\right)$, respectively. Moreover, we now consider the case of (k,l)-combiners with memory, i.e., we have an additional l-bit memory, the keystream bits z_t are computed from the LFSR state $x_t = (x_t^0, \ldots, x_t^{k-1})$ and the memory state $q_t = (q_t^0, \ldots, q_t^{l-1})$ as $z_t = C(x_t, q_t)$, while the memory is updated in each clock cycle according to $q_{t+1} = \delta(x_t, q_t)$.

In order to formalize that an LFSR output vector $(x_t, \ldots, x_{r+r-1}) \in (\{0,1\}^k)^r$ may (in conjunction with a suitable memory state q_t) yield a given keystream piece $(z_t, \ldots, z_{t+r-1})$, we use the notion of an *extended output function* introduced by Armknecht (2006) (only that we call it the extended keystream function in order to be more consistent with the rest of our notation).

Definition 6.8. *For the keystream function $C : \{0,1\}^k \times \{0,1\}^l \to \{0,1\}$ of a (k,l)-combiner with memory and an integer $r > 0$, we define the* extended

keystream function C_Ψ *by*

$$\begin{array}{rccc} C_\Psi: & \{0,1\}^l \times (\{0,1\}^k)^r & \to & \{0,1\}^r \\ & (q, x_1, \ldots, x_r) & \mapsto & (z_1, \ldots, z_r) \end{array}$$

with $q_{i+1} = \delta(x_i, q_i)$ *for* $0 \le i < r$ *and* $z_i = C(x_i, q_i)$ *for* $1 \le i \le r$.

With this notion, Armknecht and Krause (2003) adapted the structure of Eq. (6.7) to the setting of combiners with memory.

Definition 6.9. *For a* $Z = (z_1, \ldots, z_r) \in \{0,1\}^r$*, we call a Boolean function* $F_Z : (\{0,1\}^k)^r \to \{0,1\}$ *a* Z*-function (with respect to* C_Ψ*) if it is not constant zero and satisfies*

$$C_\Psi(q, x_1, \ldots, x_r) = Z \;\Rightarrow\; F_Z(x_1, \ldots, x_r) = 0 \tag{6.8}$$

for all $q \in \{0,1\}^l$ *and all* $(x_1, \ldots, x_r) \in (\{0,1\}^k)^r$.

Note that this definition implies that F_Z vanishes on all combinations of LFSR-inputs over r clock cycles and starting states q that yield the keystream piece Z.

The algebraic attack of Armknecht and Krause (2003) now consists in computing for each $Z \in \{0,1\}^r$ a Z-function F_Z of the lowest possible degree and to set up the system of equations

$$F_{(z_t, \ldots, z_{t+r-1})}(x_t, \ldots, x_{t+r-1}) = 0, \quad t = 0, 1, \ldots \;,$$

express the x_t in terms of the initial LFSR states and solve the system.

As mentioned earlier, the efficiency of the attack fundamentally depends on the degree of the Z-functions. Therefore, we want to bound the lowest possible Z-function degree that can occur for a given (k,l)-combiner.

Definition 6.10. *For a* $Z \in \{0,1\}^r$*, we define*

$$X_{Z,Q} := \left\{ x \in \left(\{0,1\}^k\right)^r \,|\, C_\Psi(Q, x_1, \ldots, x_r) = Z \right\}$$

$$X_Z := \left\{ x \in \left(\{0,1\}^k\right)^r \,|\, \exists q \in \{0,1\}^l : C_\Psi(q, x_1, \ldots, x_r) = Z \right\} = \bigcup_{Q \in \{0,1\}^l} X_{Z,Q}$$

From Eq. (6.8) we deduce that F_Z is a Z-function if and only if $F(x) = 0$ for all $x \in X_Z$. This leads directly to the notion of annihilators.

Definition 6.11. *We say that a Boolean function* $p : \{0,1\}^n \to \{0,1\}$*,* $p \not\equiv 0$*, is an* annihilator *of a subset* $A \subseteq \{0,1\}^n$ *if* $p(x) = 0$ *for all* $x \in A$*. We denote the set of annihilators of* A *by* $\mathrm{Ann}(A)$*. Furthermore, we define for* $A \subset \{0,1\}^n$

$$\mathrm{mindeg}(A) := \min\{\deg(f) | f \in \mathrm{Ann}(A)\} \;.$$

If $A = \{0,1\}^n$*, we set* $\mathrm{mindeg}(A) := \infty$.

We observe that if we can prove a lower bound for $\mathrm{mindeg}(X_Z)$ for all Z, this gives a lower bound for Z-function degrees and hence the effort required by an algebraic attack. In the following, we will propose a construction which enables us to derive such a lower bound.

We first show that under certain conditions, each special lower bound for $\mathrm{mindeg}(X_{z_r,Q})$ is also a general lower bound for $\mathrm{mindeg}(X_{(z_1,\ldots,z_r)})$.

Theorem 6.12. *If the keystream function C can be expressed as*

$$C(x,q) = \alpha(x) \oplus \beta(q) \ , \tag{6.9}$$

with $\alpha : \{0,1\}^k \to \{0,1\}$ satisfying $\text{mindeg}\left(\alpha^{-1}(0)\right) = \text{mindeg}\left(\alpha^{-1}(1)\right) = d$ *and $\beta : \{0,1\}^l \to \{0,1\}$ then*

$$\text{mindeg}(X_Z) \geq \text{mindeg}(X_{Z,Q}) = d$$

for all $r \geq 1$, $Z = (z_1, \ldots, z_r) \in \{0,1\}^r$, and $Q \in \{0,1\}^l$.

Proof. Because of $X_{Z,Q} \subseteq X_Z$, each annihilator of X_Z is also an annihilator of $X_{Z,Q}$. This shows the first inequality.

Moreover, all choices $z \in \{0,1\}$ and $Q \in \{0,1\}^l$ satisfy $X_{z,Q} = \alpha^{-1}(\beta(Q) \oplus z)$ and therefore $\text{mindeg}(X_{z,Q}) = d$.

Let $r \geq 1$, $Z = (z_1, \ldots, z_r) \in \{0,1\}^r$, $q_1 \in \{0,1\}^l$ and $f(Y_1) \in \mathbb{F}_2[Y_1]$ be an annihilator of X_{z_1,q_1}. Then f can be seen as an element in $\mathbb{F}_2[Y_1, \ldots, Y_r]$ which annihilates X_{Z,q_1}, too. This shows that $\text{mindeg}\left(x_{Z,q_1}\right) \leq \text{mindeg}\left(x_{z_1 q_1}\right) = d$.

We prove now by induction over r that $\text{mindeg}(X_{Z,q_1}) \geq d$ for all choices of q_1 and Z. For $r = 1$, the claim is certainly true. Now let $r > 1$ and the claim be true for all $r' < r$. Fix $Z = (z_1, \ldots, z_r)$ and q_1 and $f(Y_1, \ldots, Y_r) \in \text{Ann}(X_{Z,Q})$ having the minimal degree $\text{mindeg}(X_{Q,Z})$. Choose an arbitrary value $(x_1, \ldots, x_r) \in (\{0,1\}^k)^r$ and set $q_2 := \delta(x_1, q_1)$. Then

$$f^*(Y_2 \ldots, Y_r) := f(x_1, Y_2, \ldots, Y_r)$$

annihilates $X_{(z_2,\ldots,z_r),q_2}$. Hence,

$$\text{mindeg}\left(x_{Z,q_1}\right) = \deg(f) \geq \deg(f^*) \geq \text{mindeg}\left(x_{(z_2,\ldots,z_r),q_2}\right) \geq d \ ,$$

where the last inequality is true by assumption. □

6.3 Countermeasures and Design Principles

6.3.1 Increasing the Resistance against Correlation Attacks

Theorem 6.6 allows to compute the biases which are relevant for correlation attacks against combiners with memory with a keystream function as in Eq. (6.4) and to derive corresponding design criteria to immunize them against attacks that exploit these biases. In particular, Theorem 6.6 yields two different criteria for δ and β_t in order to achieve that $\lambda(F^r) = 0$ for all $r \geq 1$.

The first one assumes the situation that β_t is independent of $x \in \{0,1\}^k$, i.e., $\beta_t(q,x) = \beta_t(q)$ for all x, which holds, e.g., for E_0.

Definition 6.13. *We say that δ is* balanced *if $k = l$ and $|\{x|\delta(q,x) = q'\}| = 1$ for all q, q'.*

Note that for a balanced δ, $p(q,q') = 2^{-k}$ for all q, q'.

Theorem 6.14. *Let β_t either be constant zero or, at least at one time t, depend only on q and be balanced. If δ is also balanced, then $\lambda(F^r) = 0$.*

Proof. If $\beta_t \equiv 0$, then B_t equals P, the transition matrix of δ. Due to $(e^T) \cdot P = e^T$, we can assume w.l.o.g. that $\beta_1 \not\equiv 0$. Observe that the property of β_t being balanced implies that $\sum_q (-1)^{\beta_1(q)} = 0$. Let $x_{(q,q')} := \{x | \delta(q,x) = q'\}$. If β_t depends only on q, then $b_t(q,q')$ can be rewritten to

$$b_t(q,q') = \left\{ \begin{array}{cc} 0 & \text{if } x_{(q,q')} = \emptyset \\ |x_{(q,q')}|/2^k & \text{if } x_{(q,q')} \neq \emptyset \wedge \beta(q) = 0 \\ -|x_{(q,q')}|/2^k & \text{if } x_{(q,q')} \neq \emptyset \wedge \beta(q) = 1 \end{array} \right\}$$
$$= (-1)^{\beta_t(q)} \cdot p(q,q') \ .$$

Let $v^T := (e^T) \cdot B_1$. We show that v is already the all-zero vector, which concludes the proof. Let $(v^T)_q$ denote the q-th entry of v^T. We have

$$(v^T)_q = \sum_q (-1)^{\beta_1(q)} p(q,q') = 2^{-k} \cdot \underbrace{\sum_q (-1)^{\beta_1(q)}}_{=0} = 0 \ . \qquad \square$$

In the case that the functions β_t are not independent of x, it is also possible to entirely avoid correlations if we put some additional restrictions on β_t.

Definition 6.15. *The function $\beta : \{0,1\}^l \times \{0,1\}^k \to \{0,1\}$ is called* q-balanced *if all states $q \in \{0,1\}^l$ satisfy*

$$\left|\left\{x \in \{0,1\}^k |\ \beta(q,x) = 0\right\}\right| = \left|\left\{x \in \{0,1\}^k |\ \beta(q,x) = 1\right\}\right| \ .$$

Lemma 6.16. *Let B denote the bias matrix of the state transition function $\delta : \{0,1\}^l \times \{0,1\}^k \to \{0,1\}^l$ and a q-balanced function $\beta : \{0,1\}^l \times \{0,1\}^k \to \{0,1\}$. Then $B \circ e = \vec{0}$.*

Proof. It can be easily checked that for all $q \in \{0,1\}^l$,

$$(B \circ e)_q = \frac{\left|\left\{x \in \{0,1\}^k,\ \beta(q,x) = 0\right\}\right| - \left|\left\{x \in \{0,1\}^k,\ \beta(q,x) = 1\right\}\right|}{2^k} \ ,$$

which, by definition, vanishes if β is q-balanced. $\square$

Theorem 6.17. *Let $r \geq 1$ and β_t be either q-balanced or constant zero for all t, $1 \leq t \leq r$. Then $\lambda(F^r) = 0$.*

Proof. Note that for $\beta_t \equiv 0$, the bias matrix B_t equals the transition matrix P. As each row of P corresponds to a probability distribution over $\{0,1\}^l$, we obtain $P \circ e = e$. The rest follows straightforwardly from Theorem 6.6. $\square$

We want to point out that the previous statements are only true as long as the corresponding input words x_t are independent values in $\{0,1\}^k$. In the case that LFSRs are used as driving devices, this is only the case as long as r is at most the length of the shortest LFSR. This imposes no serious drawback, because so far, no feasible methods are known to compute the bias while considering the LFSR-structure.

As we have seen, our results immediately imply two different design criteria to avoid any biased linear combinations in the expressions $\beta(q_t)$. Actually, they have even wider applications. For example, the keystream function

$$f\left(\left(c^1, c^2, c^3, c^4\right), \left(x^1, x^2, x^3, x^4\right)\right) = c^2 \oplus x^1 \oplus x^2 \oplus x^3 \oplus x^4$$

used in E_0 is q-balanced. This guarantees that no biased linear combinations of the keystream bits z_t exist for $r \leq 25$, the length of the shortest LFSR.

6.3.2 Increasing the Resistance against Algebraic Attacks

We have seen that the efficiency of algebraic drops with increasing minimum degree of the Z-functions. Theorem 6.12 then implies the following strategy. Choose a keystream function $C(x,q) = \alpha(x) \oplus \beta(q)$ such that mindeg $\left(\alpha^{-1}(0)\right)$ and mindeg $\left(\alpha^{-1}(1)\right)$ is the maximum possible value. This will guarantee the same lower bound for all Z-functions, as long the values $x_1, \ldots, x_r$ are independent elements in $\{0,1\}^k$. In the case that they are the outputs of LFSRs, this condition holds if r is no larger than the length of the shortest LFSR (e.g., 25 in the case of E_0). This restriction is not critical, since currently known methods (e.g., Armknecht et al. (2006), Didier and Tillich (2006)) are only able to practically derive Z-functions if r is not much larger than 20.

The value d is equivalently known under the term *algebraic immunity*, which was introduced by Meier et al. (2004) in the context of memoryless combiners, extended to combiners with memory by Armknecht (2004b), and examined in several papers since then.

Observation 6.18 (Courtois and Meier (2003)). *Any Boolean function in n variables has an algebraic immunity of at most $\lceil \frac{n}{2} \rceil$.*

This means that any proposal for a function with optimum algebraic immunity $\lceil \frac{n}{2} \rceil$ can be incorporated in our design.

Proposals on how to construct functions with maximum (or at least high) algebraic immunity have been made, e.g., by Armknecht and Krause (2006), Carlet (2008), Dalai et al. (2005). A rather straightforward candidate is the (generalized) majority-function.

Corollary 6.19. *Let $k \geq 1$. The majority function $maj : \{0,1\}^k \to \{0,1\}$ defined by*

$$maj(x) = \begin{cases} 0 & \text{if } \mathrm{wt}(x) < k/2 \text{ or } \mathrm{wt}(x) = k/2 \text{ and } x_1 = 0 \\ 1 & \text{otherwise} \end{cases},$$

satisfies $\mathrm{mindeg}(maj^{-1}(0)) = \mathrm{mindeg}(maj^{-1}(1)) = k/2$.

A proof can be found in (Braeken and Lano, 2005). The authors pointed out that *maj* has a very low nonlinearity, making it a bad choice for memoryless combiners. However, this is no problem in our setting, as long as high biases λ are avoided (e.g., using the principles described in Section 6.3.1).

Using our design principle and a Boolean function with optimum algebraic immunity, it is possible to exclude the existence of Z-functions having a degree less than $\lceil k/2 \rceil$. In fact, experiments have shown by exhaustive search that the actual values of mindeg are often higher, showing that $\lceil k/2 \rceil$ seems to be a rather coarse estimation. Moreover, one can easily increase this bound, even without increasing the number of LFSRs by using several different bits per LFSR and clock cycle. For example, in the case of E_0, one could use the modified output and update functions $z_t := maj(x_{2t-1}, x_{2t})$ and $q_{t+1} := \delta(\delta(q_t, x_{2t-1}), x_{2t})$. The bitrate is halfed, but the existence of Z-functions of degree less than 4 can be excluded.

6.4 Application to E_0

In this section, we apply the results from the previous sections to improve the security of the E_0 keystream generator. Consequently, we assume that $k = l = 4$ and that the keystream bit z_t is computed by $z_t = f(q_t, x_t) = \alpha(x_t) \oplus \beta(q_t)$, with $\alpha(x_t) = x_t^0 \oplus x_t^1 \oplus x_t^2 \oplus x_t^3$ and $\beta(q_t) = q_t^1$. Recall from Section 4.2.2 that the state transition function of E_0 is defined as

$$\delta_0(q_t, x_t) = \left(\mathcal{S}_{t+1}^1 \oplus q_t^0 \oplus q_t^3, \mathcal{S}_{t+1}^0 \oplus q_t^1 \oplus q_t^2 \oplus q_t^3, q_t^0, q_t^1\right) \quad ,$$

where $\mathcal{S}_{t+1} = (\mathcal{S}_{t+1}^1, \mathcal{S}_{t+1}^0) = \left\lfloor \frac{x_t^0 + x_t^1 + x_t^2 + x_t^3 + 2 \cdot q_t^0 + q_t^1}{2} \right\rfloor$.

Lu and Vaudenay (2005, 2008) proved that $\lambda_{\max} = 25/256$ for $r \leq 25$, where 25 is the length of the shortest LFSR. This observation and the exploit of a synchronization flaw led to the currently best attack on the Bluetooth cipher (Lu and Vaudenay, 2004). Table 6.1 shows the resource requirements of this attack.

The currently best algebraic attack on E_0 in this scenario uses Z-functions of degree 4 over 4 clocks (Armknecht and Krause, 2003). The corresponding performance data are given in Table 6.2. Courtois (2003) proposed a method to obtain equations of degree 3, however with the enormous value $r \approx 8.822.188$. It is still an open question whether Z-functions exist of degree < 4 and $r \ll 8.822.188$ for E_0.

We now try to improve the resistance of E_0 to correlation attacks and algebraic attacks of the described types by carefully modifying its components.

First, using our C-implementation of Theorem 6.6 based on the ATLAS linear algebra library (Whaley and Petitet, 2005), we computed the maximum absolute biases over 25 clock cycles (the length of E_0's shortest LFSR) for all 16 E_0-variants in which β is defined as $\beta_{(a_1,a_2,a_3,a_4)}(q_t) = a_1 \cdot q_t^0 \oplus a_2 \cdot q_t^1 \oplus a_3 \cdot q_t^2 \oplus a_4 \cdot q_t^3$ for $a = (a_1, a_2, a_3, a_4) \in \{0,1\}^4$. Note that the original β corresponds to $\beta_{(0,1,0,0)}$. As Table 6.3 shows, the minimum absolute bias $\lambda = 0.024414$ is obtained for $a = (0, 1, 1, 1)$. We denote the corresponding generator by E_0^1. However, with the help of a toolkit developed by Brandeis (2004) that determines Z-functions by exhaustive search, we have computed Z-functions of degree 3 for E_0^1, which makes it weaker against algebraic attacks than the original E_0. However, choosing $a = (1, 0, 1, 1)$, i.e., the a-value with the second best minimum absolute bias, yields mindeg $= 6$. We call the corresponding generator E_0^2.

In the next step, we exploit our theory to completely avoid biases. Starting from the original definition of E_0, we obtain the generator E_0^3 by replacing the state transition function by δ_1, which we define as the integer addition modulo

Table 6.1: The resource consumption of the fastest correlation attack on E_0 as presented by Lu and Vaudenay (2004)

$\lambda_{\max}$	Frames	Data	Time	Space
λ	$m = \max(\frac{1}{\lambda^{10}}, \frac{236.59}{\lambda^8})$	$24m$	$36m + 3 \cdot 2^{18} \cdot \min(m, 2^{18})$	m
$\frac{25}{256}$	$2^{34.74}$	$2^{39.32}$	$2^{40.17}$	$2^{34.74}$

Table 6.2: The resource consumption of an algebraic attack on E_0 with key size n and an equation of degree d

	$\#F$	Data	Time	Space
General	φ	$O\left(\binom{n}{d}/\varphi\right)$	$O\left(\binom{n}{d}^3\right)$	$O\left(\binom{n}{d}^2\right)$
E_0: $n = 128$, $d = 4$	1	$2^{23.35}$	$2^{70.04}$	$2^{46.69}$

2^4, i.e.,

$$\delta_1\left(q_t^0, \ldots, q_t^3, x_t^0, \ldots, x_t^3\right) = \left(\sum_{i=0}^{3} q_t^{3-i} 2^i + \sum_{j=0}^{3} x_t^{3-j} 2^j\right) \bmod 16 \ .$$

Since δ_1 is balanced, Theorem 6.14 implies $\lambda = 0$. However, we computed Z-functions of degree 3 for E_0^3.

We therefore replace the function α of E_0^3 by the majority function described in Corollary 6.19. For the resulting generator E_0^4, we obtain mindeg $= 5$.

If we additionally replace the function β by the majority function, mindeg even increases to 6. Note that the $\lambda = 0$ property is still preserved by these modifications. Thus, we obtain a keystream generator E_0^5 with $\lambda_{\max} = 0$ whose resistance against algebraic attacks is significantly increased compared to the original E_0.

For all our variants of E_0, Table 6.4 lists the minimum degree and the respective number of Z-functions over r clock cycles. For Example, for E_0^4, the minimum degree of Z-functions over up to 5 clock cycles is 5, and there are 40, 264, 896, and 2528 Z-functions over $2, 3, 4$ and 5 clock cycles, respectively.

The computation of the number of Z-functions over 6 clocks for E_0^5 could not be completed with the resources at our disposal. Since in all our experiments, the minimum degree of the Z-functions never decreased with increasing r, we suspect that mindeg $= 6$ will also hold for E_0^5 and $r = 6$.

Note that in all cases, the values of mindeg were actually higher than the theoretical lower bound $\lceil k/2 \rceil = 2$.

The constructions of the considered generators and the respective performances of algebraic and correlation attacks are summarized in Table 6.6 and illustrated in Figure 6.1.

We note that the generator E_0^2, which is just a slight modification of E_0 (we only made β depend on two more state bits), already yields a similar resistance against algebraic attacks as E_0^5 and significantly decreases the vulnerability against correlation attacks.

Table 6.3: Maximum absolute biases and performance of correlation attacks for β_a-generators

a	max (λ)	Frames	Data	Time	Space
$(0,0,0,1)$	0.097656	$2^{34.74}$	$2^{39.32}$	$2^{40.17}$	$2^{34.74}$
$(0,0,1,0)$	0.244141	$2^{24.16}$	$2^{28.74}$	$2^{37.59}$	$2^{24.16}$
$(0,0,1,1)$	0.156250	$2^{29.31}$	$2^{33.90}$	$2^{37.74}$	$2^{29.31}$
$(0,1,0,0)$	**0.097656**	$\mathbf{2^{34.74}}$	$\mathbf{2^{39.32}}$	$\mathbf{2^{40.17}}$	$\mathbf{2^{34.74}}$
$(0,1,0,1)$	0.097656	$2^{34.74}$	$2^{39.32}$	$2^{40.17}$	$2^{34.74}$
$(0,1,1,0)$	0.156250	$2^{29.31}$	$2^{33.90}$	$2^{37.74}$	$2^{29.31}$
$(0,1,1,1)$	**0.024414**	$\mathbf{2^{53.56}}$	$\mathbf{2^{58.15}}$	$\mathbf{2^{58.73}}$	$\mathbf{2^{53.56}}$
$(1,0,0,0)$	0.244141	$2^{24.16}$	$2^{28.74}$	$2^{37.59}$	$2^{24.16}$
$(1,0,0,1)$	0.250000	$2^{23.89}$	$2^{28.47}$	$2^{37.59}$	$2^{23.89}$
$(1,0,1,0)$	0.097656	$2^{34.74}$	$2^{39.32}$	$2^{40.17}$	$2^{34.74}$
$(1,0,1,1)$	**0.038528**	$\mathbf{2^{46.98}}$	$\mathbf{2^{51.56}}$	$\mathbf{2^{52.15}}$	$\mathbf{2^{46.98}}$
$(1,1,0,0)$	0.156250	$2^{29.31}$	$2^{33.90}$	$2^{37.74}$	$2^{29.31}$
$(1,1,0,1)$	0.156250	$2^{29.31}$	$2^{33.90}$	$2^{37.74}$	$2^{29.31}$
$(1,1,1,0)$	0.152588	$2^{29.58}$	$2^{34.17}$	$2^{37.77}$	$2^{29.58}$
$(1,1,1,1)$	0.097656	$2^{34.74}$	$2^{39.32}$	$2^{40.17}$	$2^{34.74}$

Table 6.4: mindeg and number of Z-functions for the candidate generators

Cipher	E_0	E_0^1	E_0^2	E_0^3	E_0^4	E_0^5
mindeg	4	3	6	3	5	6
Clocks	Number of equations					
$r=2$	0	12	0	4	40	12
$r=3$	0	48	24	40	264	318
$r=4$	16	144	160	144	896	1416
$r=5$	64	384	544	416	2528	>0
$r=6$	192	?	>0	?	?	?

Table 6.5: Definitions of the candidate generators

	δ	$\alpha\left(x_t^1,x_t^2,x_t^3,x_t^4\right)$	$\beta\left(c_t^1,c_t^2,c_t^3,c_t^4\right)$
E_0	δ_0	$x_t^1 \oplus x_t^2 \oplus x_t^3 \oplus x_t^4$	c_t^2
E_0^1	δ_0	$x_t^1 \oplus x_t^2 \oplus x_t^3 \oplus x_t^4$	$c_t^2 \oplus c_t^3 \oplus c_t^4$
E_0^2	δ_0	$x_t^1 \oplus x_t^2 \oplus x_t^3 \oplus x_t^4$	$c_t^1 \oplus c_t^3 \oplus c_t^4$
E_0^3	δ_1	$x_t^1 \oplus x_t^2 \oplus x_t^3 \oplus x_t^4$	c_t^2
E_0^4	δ_1	$\mathrm{maj}(x_t^1,x_t^2,x_t^3,x_t^4)$	c_t^2
E_0^5	δ_1	$\mathrm{maj}(x_t^1,x_t^2,x_t^3,x_t^4)$	$\mathrm{maj}(c_t^1,c_t^2,c_t^3,c_t^4)$

Table 6.6: Performance of algebraic and correlation attacks on the candidate generators

	Algebraic Attack		Correlation Attack	
	mindeg	Time	λ	Time
E_0	4	$2^{70.18}$	0.097656	$2^{40.17}$
E_0^1	3	$2^{55.25}$	0.024414	$2^{58.73}$
E_0^2	6	$2^{97.22}$	0.038528	$2^{52.15}$
E_0^3	3	$2^{55.25}$	0	n/a
E_0^4	5	$2^{84.11}$	0	n/a
E_0^5	6	$2^{97.22}$	0	n/a

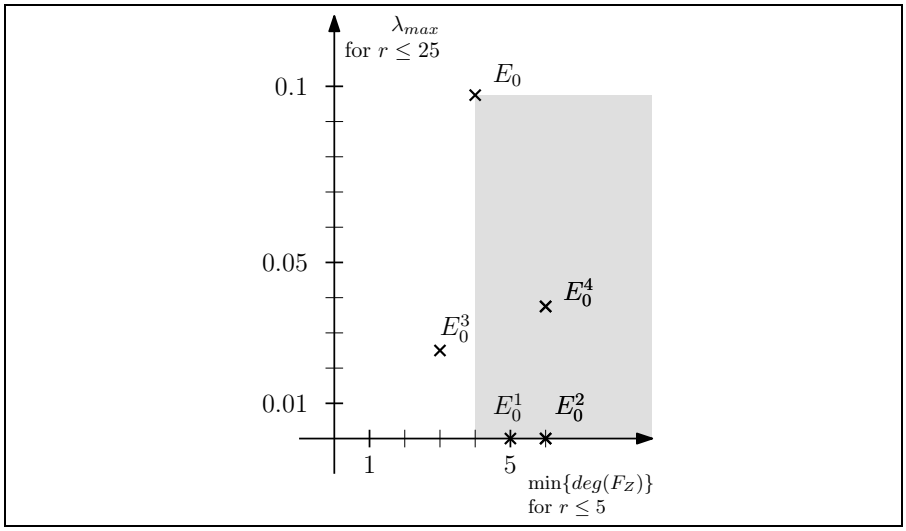

Figure 6.1: Comparison of the candidate generators to E_0

Part II

Authenticity with Linear Protocols

Chapter 7

Algorithms for Entity and Message Authentication

7.1 Security Definitions and Attacker Models

We have seen in the first part of this thesis that in order for people to call a particular system *secure*, this system should allow for confidential communication, which is usually achieved by encrypting the messages that are exchanged between communication partners.

Let us revisit our two-party communication scenario from Section 2.1. Two parties, Alice and Bob, communicate over a channel that is accessible to an adversary. Besides the confidentiality of exchanged messages, it may also be beneficial for Alice to ensure that she is really talking to Bob instead of an adversary masquerading as Bob, and that a message that claims to come from Bob was in fact sent by Bob and has not been modified during the transmission.

These requirements may seem less obvious than message confidentiality at first glance, but turn out to be equally, if not even more important in many practical systems. Consider, for example, banking transactions. It is certainly desirable to hinder an adversary observing how much money a customer withdraws from his account or to whom he transfers how much money, but it seems even more important to prevent an adversary from withdrawing or transferring money from somebody else's account.

In the process of ensuring these properties, Alice has to gain confidence in the identity of Bob as communication partner or as originator of a message. An *identity* is a set of information that distinguishes a specific entity from every other within a particular environment, e.g., a given and family name, an e-mail address, or a URI (Adams, 2005). This implies that the mapping $r : A \to I$ from an entity set A to an identity space I should be injective, i.e., no two entities $a_1, a_2 \in A$ are mapped to the same identity $i \in I$. Note that the mapping between entities and identities can also be modeled as a relation $R \subseteq A \times I$ with $(a, i) \in R$ if and only if identity i is associated to entitiy a. We say that a *has* identity i, or that identity i *is bound to* entity a, and call the tuple (a, i) an *identity binding* for a. An identity may also be bound to another identity from a different identity space, e.g., an international bank account number (IBAN, see ISO/IEC (2007)) to an e-mail address.

7.1.1 Entity Authentication

Consider a bank customer who uses an automated teller machine (ATM) to withdraw money from his account. In the course of the transaction, the customer is usually required to plug his bank card into the machine and to enter his personal identification number (PIN). If PIN and bank account number match the information that is stored in the bank's database, the ATM is convinced that the account number in fact belongs to the person standing in front of the machine.

In our more abstract communication setting, Bob (the *claimant* or the *prover*) claims to have a certain identity, e.g., a bank account number. In order for Alice (the *verifier*) to believe that the presented identity really belongs to Bob, she will usually require some corroborating evidence of his claim, e.g. a PIN. The process of obtaining and verifying this evidence is called *entity authentication* (Adams, 2005), and a particular algorithm that implements entity authentication is called an *entity authentication scheme* or *entity authentication protocol*. As described by Zuccherato (2005), the corroborating evidence (sometimes also termed *credentials*) is usually computed based on

- someting knwon, e.g., a password or personal identification number (PIN),
- something possessed, e.g., physical devices such as mechanical keys or smart cards,
- something inherent, e.g., biometric information such as a fingerprint or the structure of the iris.

If the verifier is convinced by the corroborating evidence, we say that the authentication was successful. After a successful authentication, the prover is said to be *authenticated*. If the prover is in fact who he claims to be, then we call the prover *authentic*.

Identification is often used as a synonym for entity authentication. However, some authors define identification as the action of merely claiming an identity without providing corroborating evidence. We tend to favour the latter definition, but will avoid the term identification altogether whenever possible.

7.1.2 Entity Recognition

Entitiy authentication usually assumes that the identities of the communication partners are long-term identities that are bound to the entities during a system setup phase independently of actual communication sessions. This assumption is reasonable in systems that are rather static, e.g., a corporate IT infrastructure, but less suitable for low-end sensor network scenarios in which nodes join and leave systems dynamically and are limited in their computing power and storage capacities. In such scenarios, it is often sufficient to ensure that an entity can recognize a communication partner that she has talked to before (*entity recognition*, see Hammell et al. (2005)). Schemes that solve this problem usually can make do with short-term identities that are established dynamically when the entities start communicating for the first time, as we will see in Section 7.5.

7.1.3 Message Authentication

The way most ATMs work is based on the assumption that once the customer is authenticated, he and not the adversary will be the one talking to the ATM for the remainder of the communication session. Therefore, a customer is usually asked for corroborating evidence only once per session. On the other hand, in order to prevent an attacker from taking over the session of a customer who has left the ATM without logging out (thereby breaking the assumption), sessions are usually aborted after a relatively short period of customer inactivity.

The ATM assumption translates into our abstract setting by requiring that the adversary have no access to communication channels that have been established between legitimate communication partners. This may be valid in the ATM scenario, but is a lot less reasonable if we consider an online banking use case, in which a customer issues a credit transfer order to his bank over the internet. In general, we can make no reliable assumptions on the route an internet message takes to reach its destination, and the probabilities of a message being read or even modified on the way have to be considered non-negligible. Hence, the banking server should require corroborating evidence of each received message in fact originating from the claimed sender.

In our abstract model, Bob (the prover) would attach corroborating evidence of his creatorship to each message he sends to Alice (the verifier). As with entity authentication, the process of obtaining and verifying this corroborating evidence is called *message authentication*, an algorithm that implements message authentication is called *message authentication algorithm* or *message authentication scheme*, and a message for which the authentification was successful is called *authenticated*. If the message in fact originates from the claimed sender and was not altered during transmission, we say that the message is *authentic*.

In addition to message authenticity, many applications have additional uniqueness and timeliness requirements that duplicate or lost messages as well as messages that are received in the wrong order be detected and handled appropriately. We note that the presence of these properties is implied by some authors' definitions of authenticity (see, e.g., Menezes et al. (2001)). However, we choose to separate uniqueness and timeliness from our authenticity definition since they are sometimes covered by transport layers in communication stacks rather than by authenticity schemes in the narrow sense. An example is the widely used TLS protocol (see Dierks and Rescorla (2008)), which relies on TCP to ensure these properties.

7.1.4 Message Recognition

Similarly to entity recognition (see Section 7.1.2), message recognition as a weaker form of message authentication only requires to ensure that a message originates from a particular sender that has been talked to before, and (unlike most message authentication schemes) not consider or check any long-term sender identities. Instead, the communicating parties generate short-term identities just before starting the conversation.

7.1.5 Attacker Models

As in the case of confidential communication, we relate the security of an entity authentication scheme to an attacker (or adversary) model. The most prominent attacker goal in the entity authentication/recognition setting is to impersonate an entity, i.e., to convincingly masquerade as somebody else. Attacks that are targeted at this goal are usually called *impersonation attacks*. Another attacker goal may be to prevent the authentication of a legitimate prover or message (*denial of service*) by disturbing the authentication.

The most pessimistic assumption is an active attacker who has full control of the communication channel, as suggested in the Dolev-Yao security model (Dolev and Yao, 1983). More precisely, an active attacker may

- read all exchanged messages,
- modify exchanged mesages, especially delay or suppress their delivery or alter their content,
- introduce additional messages into the communication channel, especially replay previously recorded messages.

We note that this model particularly allows the attacker to

- present a previously recorded legitimate evidence to the verifier (*replay attack*),
- interleave several authentication sessions (running in parallel or sequentially) by using information obtained from one session in the context of another,
- disobey the authentication scheme by sending messages which the receiver does not expect in the current protocol state.
- act as a man-in-the-middle (MITM), i.e., intecept messages from one communication partner, possibly modify them, and pass them on to the receiver.

Of special interest in our analysis is a special class of active attackers, which we call *detection attackers*.

Definition 7.1. *A* detection attacker *on an entity authentication protocol is an active attacker who is restricted to the following disjoint attack stages.*

1. *Interact with a legitimate prover in any desired way.*
2. *Interact with a legitimate verifier and try to impersonate the prover.*

In the message authentication/recognition setting, we commonly assume that the attacker has active control over the communication channel as above, and is additionally able to force Bob to send message payload data x_i of his choice (which will then be accepted by Alice as authentic). Thereby, his choices of the x_i may be adaptive, i.e., x_i may depend on the information that was obtained for $x_{i'}$ for $i' < i$. We define that he has reached his goal if he is able to generate a message with payload $x \neq x_i$ for all i that is authenticated by Alice (existential forgery in a chosen message scenario, see, e.g., Lucks et al. (2008)).

Sometimes security is evaluated also with respect to passive attackers who are able to read exchanged messages, but cannot influence the communication channel in any way.

In all cases, we follow Kerckhoffs' principle (Kerckhoffs, 1883) also in the authentication setting and assume the attacker to know the entire specification of the authentication scheme and all information that the scheme processes except for the data that it explicitly requires to be kept confidential.

7.2 Message Authentication Codes

Definition 7.2. *A* Message Authentication Code *(MAC) is a mapping*

$$\begin{array}{rccc} \mathrm{MAC}: & \{0,1\}^* \times \{0,1\}^n & \to & \{0,1\}^l \\ & (x,k) & \mapsto & m \end{array} \tag{7.1}$$

that computes for a message x an authentication code m under an n-bit key k.

A MAC is commonly used in our two-party communication scenario in the following way (see Fig. 7.1). Alice and Bob agree on a symmetric key k prior to the communication. Bob computes for a message x the value $m = \mathrm{MAC}(x,k)$ and transmits $(x,m) = (x, \mathrm{MAC}(x,k))$ to Alice. Alice computes for a received message (x',m') the value $\mathrm{MACverify}(x',m',k)$ with

$$\begin{array}{rccc} \mathrm{MACverify}: & \{0,1\}^* \times \{0,1\}^l \times \{0,1\}^n & \to & \{0,1\} \\ & (x,m,k) & \mapsto & \begin{cases} 1 & \text{if } \mathrm{MAC}(x,k) = m \\ 0 & \text{otherwise} \end{cases} \end{array},$$

and believes the message to come from Bob if $\mathrm{MACverify}(x',m',k) = 1$. Hence, the value $m = \mathrm{MAC}(x,k)$ serves as corroborating evidence of the authenticity of x.

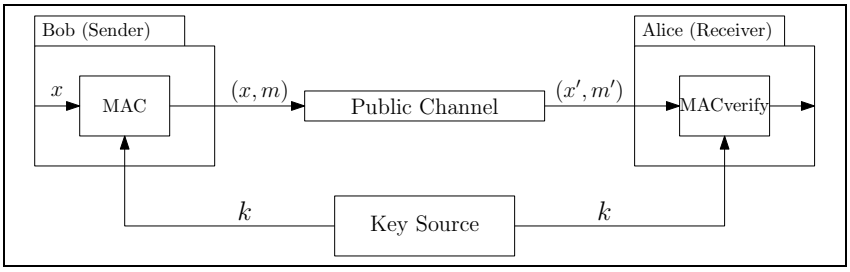

Figure 7.1: Message authentication with message authentication codes

A MAC is considered secure if it is infeasible to perform an existential forgery under an adaptive chosen message attack (see Section 7.1.5), i.e., an attacker who may obtain $\mathrm{MAC}(x_i,k)$ under the secret key k for messages x_i of his choice is not able to produce with a realistic amount of resources a pair (x,m) with $x \neq x_i$ for all i such that $\mathrm{MACverify}(x,m,k) = 1$. Obviously, recovering the secert key k that is used to generate the authentication code is sufficient for an existential forgery.

Similarly to the cipher systems described in Section 2.1, since Alice and Bob both use the same key in the production and verification of m, message

authentication codes are said to belong to the set of symmetric authentication schemes.

We note that a MAC by itself does not provide assurance of message timeliness nor uniqueness.

7.2.1 Message Authentication Codes based on Block Ciphers

The structure of Eq. (7.1) suggests to use block ciphers as building blocks for message authentication codes. This idea is implemented, e.g., in the CBC-MAC (ISO/IEC, 1999), which is based on the CBC mode of a block cipher $E : \{0,1\}^l \times \{0,1\}^n \rightarrow \{0,1\}^l$ (see Section 2.2). If the CBC mode encryption of an m-block message $b = (b_1, \ldots, b_m)$ is given by $E^{\text{CBC}}((b_1, \ldots, b_m), k, \text{IV}) := (c_1, \ldots, c_m)$, the CBC-MAC value for this message is computed as

$$\text{MAC}^{\text{CBC}}((b_1, \ldots, b_m), k) := c_m$$

with $(c_1, \ldots, c_m) = E^{\text{CBC}}((b_1, \ldots, b_m), k, 0)$.

Note that the CBC-MAC is insecure if we allow the messages to have different lenghts, since we can forge an authentic message by appending arbitrary blocks to observed authentic messages for which the CBC-MAC value is known. This issue is addressed by variants of the CBC-MAC construction such as CMAC (see, e.g., Preneel (2005) for a discussion).

If a message authentication code is built from a block cipher, this cipher will dominate the resource requirements of the MAC. The AES block cipher (see Menezes et al. (2001)) is widely used for building message authentication codes, and efficient implementations, particularly for resource-constraint environments are continously being developed and optimized (Moradi et al., 2011). In addition to general purpose block ciphers, lightweight block ciphers such as PRESENT (Bogdanov et al., 2007) and KATAN/KTANTAN (De Cannière et al., 2009) are specifically targeted at low-end devices.

7.2.2 Message Authentication Codes based on Cryptographic Hash Functions

Definition 7.3. *A* cryptographic hash function *is a mapping* $H : \{0,1\}^* \rightarrow \{0,1\}^l$ *which maps an input of arbitrary length to a fixed-length output.*

Cryptographic hash functions are usually required to be

- collision resistant, i.e., it is infeasible for an adversary to find two inputs $x \neq x'$ such that $h(x) = h(x')$,
- preimage resistant, i.e., it is infeasible for an adversary to find for a given output $y \in \{0,1\}^l$ an input x such that $h(x) = y$, and
- 2nd preimage resistant, i.e., it is infeasible for an adversary to find for a given input x another input x' such that $h(x) = h(x')$.

We note that collision resistance implies 2nd preimage resistance. However, preimage resistance does not imply 2nd preimage resistance, nor does 2nd preimage resistance imply preimage resistance. A cryptographic hash function that is both preimage resistant and 2nd preimage resistant is said to be *one-way*.

A collision resistant cryptographic hash function H for arbitraty inputs $x \in \{0,1\}^*$ may be constructed from a collision resistant *compression function* $h : \{0,1\}^c \times \{0,1\}^d \to \{0,1\}^c$ with $c < d$ by expanding x to L blocks of length d (with the last block only containing the bitlength of x), i.e., $x = (M_1, \ldots, M_L)$, and computing the output as $H(x) := H_L$ with

$$H_i := \begin{cases} C & \text{for } i = 0 \\ h(H_{i-1}, M_i) & \text{for } 0 < i \leq L \end{cases} \quad \text{with } C \in \{0,1\}^c \text{ constant.}$$

This construction is attributed to Merkle and Damgård (Damgård, 1990, Merkle, 1979, 1990). In particular, block ciphers may be used as compression functions, e.g., as in the Davies-Meyer scheme (Davies and Price, 1984) by computing the values H_i based on a block cipher $E : \{0,1\}^l \times \{0,1\}^n \to \{0,1\}^l$ with $l = c$, $n = d$ as

$$h(H_{i-1}, M_i) := E(H_{i-1}, M_i) \oplus H_{i-1} \ ,$$

see, e.g., Black et al. (2002), Preneel et al. (1994).

Constructing a cryptographic hash function from a block cipher may be particularly beneficial on low-end devices with too little capacity to implement both a block cipher and a dedicated cryptographic hash function.

Potentially the most prominent examples of cryptographic hash functions are the MD5 hash function (Rivest, 1992) and the SHA hash function family (NIST, 2008).[1] Similarly to block ciphers, also dedicated lightweight cryptographic hash functions exist, see, e.g., Guo et al. (2011).

We omit further details and refer the interested reader to Preneel (1993) for an introduction to cryptographic hash functions and to Fleischmann et al. (2008), Preneel (2009) for information on more recent hash function proposals and their properties.

Cryptographic hash functions can readily produce a fixed-length fingerprint (or *digest*) of an arbitrarily long message, but in order to turn a hash function into a message authentication code in the sense of Eq. (7.1), it has to be specified how to handle the secret key that the authentication relies on. A common approach is the HMAC construction (Krawczyk et al., 1997) that derives an l-bit message authentication code from a cryptogrpahic hash function $H : \{0,1\}^* \to \{0,1\}^l$ and a key $k \in \{0,1\}^n$ as

$$\begin{array}{rccc} \mathrm{MAC}^H : & \{0,1\}^* \times \{0,1\}^n & \to & \{0,1\}^l \\ & (x,k) & \mapsto & H\left((k \oplus \mathrm{opad}) || H\left(k \oplus \mathrm{ipad} || x\right)\right) \end{array}$$

with publicly known constants opad and ipad.

7.3 Message Authentication with Digital Signatures

Corroborating evidence of a message's authenticity can also be computed with asymmetric cipher systems as defined in Section 2.4. Therefore, we use the

[1] Since the security of these algorithms is increasingly under question, at the time of writing of this thesis, a competition is being held by the National Institute of Standards and Technology (NIST) to select a successor algorithm for the SHA hash function family (see NIST (2010)), which is stimulating the development of many new designs and intensive research in this field.

decryption operation, which employs the private key, to produce the evidence, and the encryption operation, which is based on the public key, to verify the evidence. Since in contrast to message authentication codes the evidence is publicy verifiable without the need to establish a common secret, an evidence based on an asymmetric system is usually called *digital signature* (see, e.g., Vaudenay (2006) for an introduction).

The security definition of digital signatures is similar to the definition for message authentication codes. A digital signature is considered secure if it is infeasible for an adversary who may obtain signatures for messages x_i of his choice under the secret signature key to produce a signature for a message $x \neq x_i$ that will be accepted by a legitimate verifier. As in the MAC case, recovering the signature key is sufficient for being able to forge signatures for arbitrary messages.

In order to spend less effort on rather costly asymmetric operations, the evidence is typically computed for a digest of the message (derived with a cryptographic hash function) rather than for the message as a whole. But still, as with cipher systems, message authentication based on asymmetric digital signature schemes usually requires substantially more effort than symmetric message authentication codes, and this effort has to be considered too large in many low-end device applications.

7.4 Challenge-Response based Entity Authentication

Entity authentication is usually performed by proving the posession of some object (often called *key*), either a piece of information like a password or a PIN, or a physical object such as a mechanical key (see also Section 7.1.1).[2] In our attacker model, presenting the key itself to the prover as corroborating evidence is not an option in most cases of electronic communication, since it would disclose it also to the attacker and immediately allow for impersonation attacks. Hence, the corroborating evidence needs to be some information that is derived from the key, but not the key itself. However, if this derived information does not change from one authentication to another, it is as valuable as the key that is was derived from because the attacker could just replay it to impersonate the prover.

We see that the corroborating evidence should be some information that is derived from the object and is valid only for a short period of time, ideally only for one authentication session, such that the verifier can decide whether he is presented a recent (or fresh) evidence or some outdated information, which he would then assume to have been replayed by an attacker.

The most common techniques to implement freshness verification of a corroborating evidence are timestamps and verifier-supplied challenges. Timestamps are included into the evidence in an agreed way to document its creation time. While being rather straightforward to include, timestamps require a means for prover and verifier to agree on the current time (be it UTC time or some abstract

[2] Strictly speaking, we could also model physical objects as pieces of information by identifying them with their specification. However, obtaining this specification may not always be feasible, e.g., as in the case of physical uncloneable functions (PUFs) that are inherently determined by electrical or mechanical properties of a device (see, e.g., Pappu (2001))

counter-based time), which may not always be feasible especially in low-end device applications.

Verifier-supplied challenges are independent of time synchronization, but require the verifier to provide a challenge (usually a binary string) that the prover has to include in the computation of the evidence. If the verifier keeps track of the challenges he supplies to provers, he can reject evidences that are based on out-dated challenges. Entity authentication schemes based on verifier-supplied challenges are usually called *challenge-response authentication schemes*. Whether a challenge-response scheme is suitable for a particular resource-constraint application primarily depends on the severity of the communication overhead for transmitting the challenge to the prover.

We note that we can rather straightforwardly build challenge-response entity authentication schemes from message authentication schemes by requiring the prover to provide as corroborating evidence of his identity a message with the verifier-supplied challenge as payload and corroborating evidence of this message's authenticity. Alternatively, an encryption of the challenge can be used as corroborating evidence (ISO/IEC, 1993).

7.5 Authentication Schemes based on Hash Chains

A slightly different flavour of entity authentication protocols which has been proposed by Lamport (1981) is based on a one-way function $h : \{0,1\}^l \to \{0,1\}^l$. Prover and verifier agree on a value n, the prover chooses an arbitrary value x_0, computes the sequence $(x_i)_{1 \leq i \leq n}$ with $x_i = h(x_{i-1})$, and transmits the value x_n to the verifier in a tamper-proof, but not necessarily confidential way. As corroborating evidence in the i-th authentication session, $1 \leq i \leq n$, the prover presents the value x_{n-i} (i.e., the preimage of x_{n-i+1} under h) and is authenticated if and only if $h(x_{n-i}) = x_{n-i+1}$.

Since the sequence (x_i) is produced by repeated (chained) application of h, and h is often implemented as a cryptographic hash function or its compression function, this authentication technique is known as *hash chain* based authentication.

Due to the construction, the number of possible authentications is limited to the length of the hash chain, which, in the absence of auxiliary techniques, makes the construction slightly less suited for authenticating long-term identities. Therefore, hash chain based authentication is more often used for entity recognition than for entity authentication. On the other hand, compared to digital signatures, the scheme is computationally much more efficient on typical low-end devices, and since the transmission of the chain endpoint x_n does not have to be confidential, it requires less effort than conventional key establishment techniques in the initialization phase.

Message authentication schemes can be built based on hash chains, e.g., as in the Guy Fawkes protocol suggested by Anderson et al. (1998). The Jane Doe protocol by Lucks et al. (2008) uses the elements of the hash chain as keys for a message authentication code, and the chain elements are successively disclosed to the verifier such that he can perform the authentication. In order to prevent forgery attacks, care has to be taken not to disclose these values too soon, which is ensured by a second hash chain that is produced on the verifier's side and stepwise disclosed to the prover.

7.6 Authentication based on the Hardness of Learning Problems

A special case of lightweight challenge-response based entity authentication as described in Section 7.4 is the following generic strategy:

1. Construct from a lightweight function E a basic challenge-response protocol and reduce the security of the basic protocol against passive attackers to the hardness of a suitable learning problem.
2. Define a protocol P over E and try to reduce the security of P against active attackers to the security of the basic protocol against passive attackers.

For a function $E : X \times K \to Y$ with suitably chosen input space X, key space K and output space Y, the basic protocol is defined as follows. The verifier (Alice) and the prover (Bob) share a secret key $k \in K$. A basic round consists of the following steps.

- Alice and Bob exchange challenge information. As a special case, this step may only consist of Alice sending a publicly known constant value (a *hello* message) that is just used as a trigger to initiate the communication.
- Based on the challenges, Bob chooses a random element $x \in X$ which is distributed according to a publicly known probability distribution $\Pr_B$ and sends $z = E(x, k)$ as corroborating evidence of his knowledge of k to Alice.
- Alice verifies z based on the challenges and the common secret k.

After r such rounds and depending on the number of rounds with successful verifications, Alice decides whether to authenticate Bob.

In the following, we consider two entity authentication protocol families of this type, the HB family and the family of linear (n, k, L) protocols.

Chapter 8

The HB Family of Authentication Protocols

8.1 The HB Protocol

The HB protocol was proposed by Juels and Weis (2005) as an authentication protocol that can be executed by humans. The use case that they had in mind was securely logging into a terminal in the presence of adversaries eavesdropping on what the user types into the keyboard.

The verifier (Alice) and the prover (Bob) share a common secret $k \in \{0,1\}^n$ and a public noise parameter $\eta \in \left(0, \frac{1}{2}\right)$. A basic round of the HB protocol works as follows. Alice transmits a random challenge $a \in \{0,1\}^n$ to Bob, who chooses a value $\nu \in \{0,1\}^n$ with $\Pr[\nu = 1] = \eta$ and transmits the value $z := (a \cdot k) \oplus \nu$ as corroborating evidence to Alice, where $x \cdot y \in \{0,1\}$ for $x, y \in \{0,1\}^n$ denotes the inner product of x and y over GF(2). She accepts z if and only if z is equal to $(a \cdot k)$. Fig. 8.1 illustrates the basic round of HB.

Note that in the terminology of Section 7.6, we have $K = \mathrm{GF}(2)^n$, $X = Y = \mathrm{GF}(2)^n \times \mathrm{GF}(2)$, $y = \mathrm{GF}(2)$, and the basis operation is defined by $E((x,\nu),k) = (x,y)$, where $y = x \cdot k \oplus \nu$.

The entire protocol consists of r basic rounds, and Alice authenticates Bob

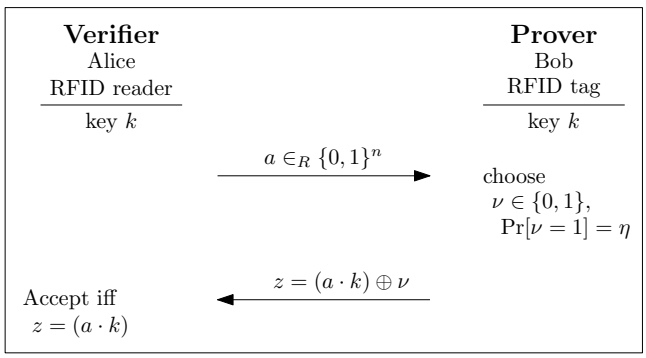

Figure 8.1: Basic round of the HB protocol

if and only if the number of rounds in which the received value z was rejected is less than ηr, or equivalently, for A denoting a random $(r \times n)$ matrix over $\{0,1\}$ and k written as $(n \times 1)$ matrix, Alice authenticates Bob if and only if

$$|(A \circ k) \oplus z| < \eta r \ .$$

Hence, a passive attacker who observes the communication between Alice and Bob and wants to impersonate Bob is faced with the problem of finding a $k' \in \{0,1\}^n$ such that

$$|(A \circ k') \oplus z| < \eta r \ .$$

This problem corresponds to the *Learning Parity in the Presence of Noise* (LPN) problem, which is defined as follows.

Definition 8.1 (Learning Parity in the Presence of Noise). *Let A be a random $(r \times n)$ matrix , let k be a random n-bit vector, let $\eta \in \left(0, \frac{1}{2}\right)$ be a constant noise parameter, and let ν be a random r-bit vector such that $|\nu| \leq \eta r$. Given A, η, and $z = (A \circ k) \oplus \nu$, find an n-bit vector k' such that $|(A \circ k') \oplus z)| \leq \eta r$.*

The LPN problem is well-established in the literature (see, e.g., Hopper and Blum (2001), Juels and Weis (2005) for an overview). It has been shown to be NP-hard, but its difficulty on random instances is still an open question. The best known algorithm to date is due to Blum et al. (2003) and has a running time of $2^{O\left(\frac{n}{\log n}\right)}$, with tighter analyses and implementation improvements proposed by Fossorier et al. (2006), Levieil and Fouque (2006).

Hopper and Blum (2001), Juels and Weis (2005) showed that the security of the HB protocol against passive attackers can be reduced to the hardness of the LPN problem in the sense that a passive attacker who can impersonate the prover in the HB protocol can be used to solve the LPN problem.

However, a detection attacker Eve (see Definition 7.1) can break the HB protocol and impersonate the prover as follows (Juels and Weis, 2005). Eve repeatedly sends the same challenge a to Bob in order to learn the error-free value of $a \cdot k$ for the unknown secret k. She repeats this procedure for n linearly independent values a (e.g., the standard basis $\{e_1, \ldots, e_n\}$ of $\{0,1\}^n$) and can then recover k by Gaussian elimination.

8.2 The HB$^+$ Protocol

HB$^+$ was proposed by Juels and Weis (2005) to strenghen HB against active attackers. It introduces an additional message, a prover-supplied blinding factor, and works as follows.

Verifier (Alice) and prover (Bob) share two secret values $k_1, k_2 \in \{0,1\}^n$ and a public noise paramter $\eta \in \left(0, \frac{1}{2}\right)$. At the beginning of a basic protocol round, Bob sends a randomly chosen blinding factor $b \in \{0,1\}^n$ to Alice, and she replies with a challenge $a \in \{0,1\}^n$. Bob then computes his corroborating evidence as $z = (a \cdot k_1) \oplus (b \cdot k_2) \oplus \nu$ with $\nu \in \{0,1\}$ such that $\Pr[\nu = 1] = \eta$ and transmits z to Alice, who accepts z if and only if z is equal to $(a \cdot k_1) \oplus (b \cdot k_2)$. The basic protocol round of HB$^+$ is illustrated in Fig. 8.2.

As in the HB protocol, the basic protocol round is repeated r times, and Alice authenticates Bob if and only if the number of basic rounds ending in rejection of the evidence is less than ηr.

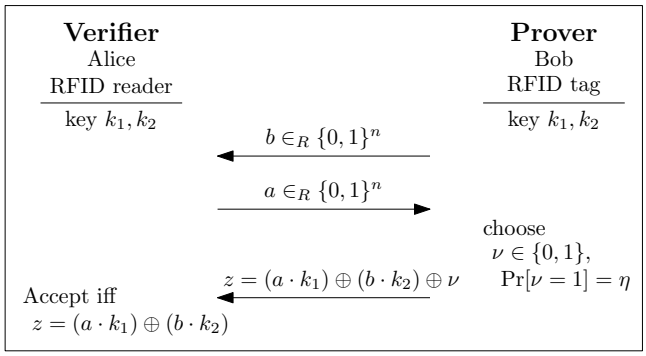

Figure 8.2: Basic round of the HB$^+$ protocol

Juels and Weis (2005) were able to show that a detection attacker (see Definition 7.1) on HB$^+$ can be used to attack the HB protocol in a passive attacker scenario, and this attack can in turn be used to solve the LPN problem. Hence, the hardness of LPN implies the security of HB$^+$ against the active adversary that HB is not able to resist.

However, Gilbert et al. (2005) observed that this reduction cannot be extended to general active adversaries, who can also act as man-in-the-middle between legitimate prover and legitimate verifier. More precisely, an HB$^+$ man-in-the-middle attacker Eve could proceed as follows. She chooses a value $\delta \in \{0,1\}^n$ and replaces a verifier-supplied challenge a by $a \oplus \delta$. Bob will then produce a corroborating evidence z' as

$$z' = (a \cdot k_1) \oplus (\delta \cdot k_1) \oplus (b \cdot k_2) \oplus \nu \ ,$$

and Eve observes whether Alice accepts z'. If this is the case, then $(\delta \cdot k_1) \oplus \nu = 0$, which implies $\delta \cdot k_1 = 0$ with probability $1 - \eta$. Conversely, if Alice rejects z', then $\delta \cdot k_1 = 1$ with probability $1 - \eta$. Similarly to the attack on HB, Eve can repeat this procedure for n linearly independent values δ and recover k_1. With this knowledge, Eve can already impersonate Bob by choosing $b = 0$. If she wants to recover also k_2, she can choose an arbitrary b and interact with a legitimate verifier, supply $z' = a \cdot k_1$ as corroborating evidence and deduce $b \cdot k_2 = 0$ if and only if the verifier accepts z'. Again, repeating these steps for n linearly independent blinding factors b yields k_2.

In the following, we will refer to this attack on HB$^+$ as Gilbert/Robshaw/Sibert attack (GRS attack).

8.3 Variants of the HB$^+$ Protocol

After its publication in 2005, HB$^+$ has recieved considerable attention in the cryptographic community. Especially its simplicity, its efficiency on the prover's side, and its provable resistance against passive attacks (albeit relying on the hardness of LPN) while being vulnerable to the rather simple GRS attack, motivated a number of follow-up proposals that aim to avoid this shortcoming while preserving as many of the advantages as possible.

However, it turns out that resisting GRS-style attacks — modifying the verifier's challenge and learning from his reaction to the evidence that the prover

produces from the perturbed input — does not seem to be an easy task. Particularly Gilbert et al. (2008b) showed that many of the follow-up proposals end up being less efficient than HB$^+$ while not providing considerably more security.

In the following, we describe the most prominent ones of these proposals and discuss their security properties.

8.3.1 The HB^{++} Protocol

The HB^{++} protocol was proposed by Bringer et al. (2006) and consists, just as HB$^+$, of a number of repetitions of a basic protocol round. However, at the beginning of an authentication session, four secrets k_1, k_1', k_2, k_2' are derived from a shared secret master key K, a prover-supplied blinding factor $B \in \{0,1\}^{80}$ and a verifier's challenge $A \in \{0,1\}^{80}$. This is done by applying a publicly known hash function to K, A and B. Bringer et al. propose a particular function $h : \{0,1\}^{768} \times \{0,1\}^{80} \times \{0,1\}^{80} \to \{0,1\}^{320}$, which implies a master secret size of 768 bits and four session keys of size 80 bits each (see Fig. 8.3).

In a basic protocol round, verifier (Alice) and prover (Bob) exchange a blinding factor b and a challenge a as in HB$^+$. Then Bob chooses two noise parameters ν and ν' with $\nu, \nu' \in \{0,1\}$ and $\Pr[\nu = 1] = \Pr[\nu' = 1] = \eta$. The corroborating evidence consists of two components (z, z') with

$$\begin{aligned} z &= (a \cdot k_1) \oplus (b \cdot k_2) \oplus \nu \\ z' &= (\mathsf{ROT}_i(f(a)) \cdot k_1') \oplus (\mathsf{ROT}_i(f(b)) \cdot k_2') \oplus \nu' \ , \end{aligned}$$

where $\mathsf{ROT}_i(x)$ denotes the rotation of $x \in \{0,1\}^*$ by i positions to the left, and f denotes a permutation. Analogously to HB$^+$, Alice checks whether z and z' satisfy

$$\begin{aligned} z &= (a \cdot k_1) \oplus (b \cdot k_2) \ , \\ z' &= (\mathsf{ROT}_i(f(a)) \cdot k_1') \oplus (\mathsf{ROT}_i(f(b)) \cdot k_2') \ . \end{aligned}$$

Figure 8.4 illustrates a basic protocol round of HB^{++}.

The basic protocol is repeated r times, and Alice authenticates Bob if both the number of erroneous z evidences and the number of erroneous z' evidences do not exceed a threshold t.

Bringer et al. (2006) showed that the resistance of HB^{++} against passive attacks can be reduced to the hardness of LPN, and that the protocol is able to resist the GRS attack if f is carefully chosen.

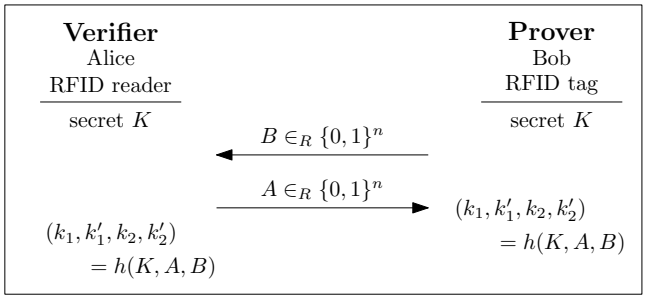

Figure 8.3: Initialization of the HB^{++} protocol

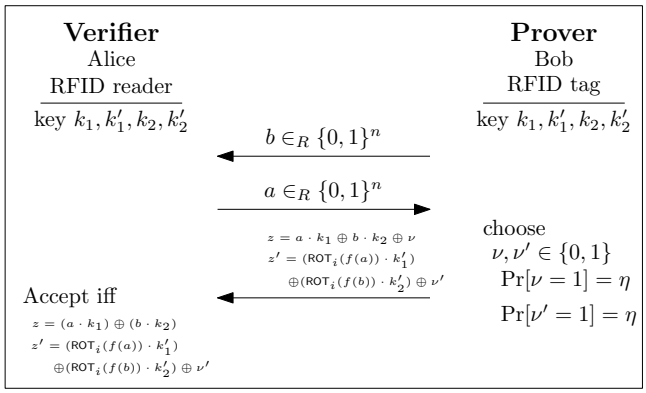

Figure 8.4: Basic round of the HB^{++} protocol

But still, HB^{++} remains vulnerable to a special extension of the GRS attack. Gilbert et al. (2008b) discovered that by disturbing the challenge a in s out of r rounds of an authentication session, exploiting the observed verification result on the verifier's side and the special structure of h, an adversary can deduce linear equations in a number of bits of k_1. The resulting system can be expressed as an LPN instance and solved with moderate effort. From the recovered k_1, the session key k_1' can be derived in a similar manner. The knowledge of k_1 and k_1' is already sufficient for impersonating Bob since the adversay can reuse blinding factors b from successful authentications of Bob along with k_1, k_1' to correct z and z' appropriately.

8.3.2 The HB* Protocol

Another variant of HB$^+$, the HB* protocol, was proposed by Duc and Kim (2007). As in HB$^+$, verifier (Alice) and prover (Bob) share two secret values k_1 and k_2. Additionally, there is a shared secret s that is used to confidentially transmit an auxiliary value γ from Bob to Alice.

At the beginning of a basic protocol round, Bob chooses $\gamma \in \{0,1\}$ with $\Pr[\gamma = 1] = \eta'$, and $\nu \in \{0,1\}$ with $\Pr[\nu = 1] = \eta$, and transmits a randomly chosen blinding factor $b \in \{0,1\}^n$ and the encrypted value of γ, which is computed as $w = (b \cdot s) \oplus \gamma$, to Alice. As in HB$^+$, Alice replies with a challenge $a \in \{0,1\}^n$. Bob then computes his corroborating evidence as

$$z = \begin{cases} (a \cdot k_1) \oplus (b \cdot k_2) \oplus \nu & \text{if } \gamma = 0 \\ (a \cdot k_2) \oplus (b \cdot k_1) \oplus \nu & \text{if } \gamma = 1 \end{cases},$$

and Alice checks whether $(a \cdot k_1) \oplus (b \cdot k_2)$ equals z if $(b \cdot s) = w$ (i.e., $\gamma = 0$), and whether $(a \cdot k_2) \oplus (b \cdot k_1)$ equals z if $(b \cdot s) \neq w$ (i.e., $\gamma = 1$). Again, Bob is authenticated if the verification fails for less than a threshold t out of r basic protocol rounds. The basic protocol round of HB* is illustrated in Fig. 8.5.

Duc and Kim claim resistance against the GRS attack, but Gilbert et al. (2008b) observed that although in each basic round one of the two protocol modes is secretly selected by the value γ, a modified GRS attack remains applicable. This attack is again based on adding a vector δ to the verifier's challenge a and exploiting the information that the result of the verification leaks about

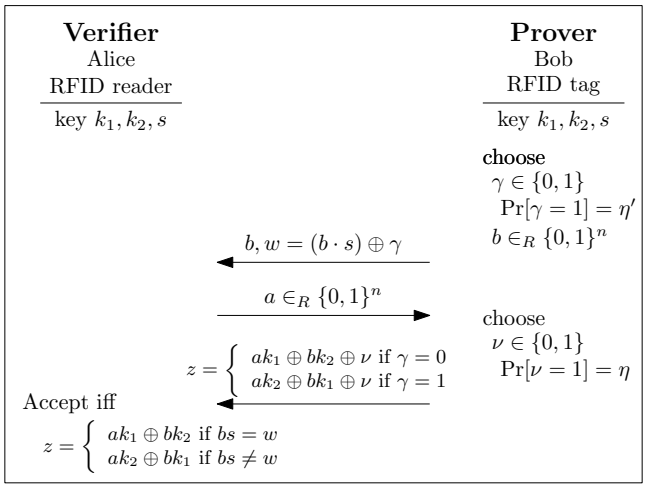

Figure 8.5: Basic round of the HB* protocol

the secrets k_1 and k_2. A case distinction shows that the acceptance probability of z varies depending on the values of $\delta \cdot k_1$ and $\delta \cdot k_2$ in such a way that the attacker can discriminate between (sets of) the cases. Depending on the choices of η and η', the attacker may either recover k_1 as in the GRS attack and impersonate Bob by sending $(b, w) = (0, 0)$ as first message, or learn the two-dimensional vectorial space $< k_1, k_2 >$, which can similarly be exploited to impersonate Bob.

8.3.3 The HB-MP Protocols

Munilla and Peinado (2007) proposed the HB-MP protocol as an HB$^+$-variant that presumably resists the GRS attack. It uses a two-pass basic protocol as follows. Both verifier (Alice) and prover (Bob) share a secret (k_1, k_2). In the i-th execution of the basic protocol, Alice sends a challenge $a \in_R \{0,1\}^m$ to Bob, who chooses a $\nu \in_R \{0,1\}^m$ such that $\Pr[\nu_i = 1] = \eta$ for all $i \in [1, m]$. Then he computes $k_1 := \text{rotate}(k_1, (k_2)_i)$, where $(k_2)_i$ denotes the i-th bit of k_2 and $\text{rotate}(x, y)$ the rotation of x by y positions. He computes $z := (a \cdot (\lfloor k_1 \rfloor_m)) \oplus \nu$ with $\lfloor k_1 \rfloor_m$ denoting the m least significant bits of k_1. Finally, he chooses a value b that satisfies $(b \cdot (\lfloor k_1 \rfloor_m)) = z$ and transmits b to Alice. Alice accepts the evidence if and only if $b \cdot (\lfloor k_1 \rfloor_m)$ equals $a \cdot (\lfloor k_1 \rfloor_m)$ which is equivalent to

$$(a \oplus b) \cdot (\lfloor k_1 \rfloor_m) = 0 \ . \tag{8.1}$$

Figure 8.6 illustrates the basic round of HB-MP.

As with HB$^+$, Bob is authenticated if the number of failed basic protocol rounds is less than some threshold t.

Despite the claim in the original proposal, HB-MP is vulnerable to a passive attack as observed by Gilbert et al. (2008b). Equation (8.1) implies that a basic protocol round is always passed if a and b are equal. Munilla and Peinado recommend immediate rejection for this case, but nevertheless suitable evidences can be constructed from an observed r-round authentication session with exchanged messages (a_i, b_i), $i \in [1, r]$. In order to impersonate the prover, we

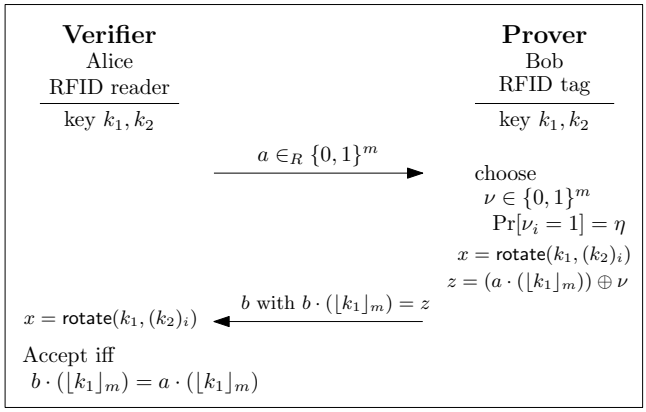

Figure 8.6: Basic round of the HB-MP protocol

compute for the verifier's challenge a'_i an evidence b'_i as $b_i := a'_i \oplus a_i \oplus b_i$. Then we have $b'_i \neq a'_i$ since $a_i \neq b_i$, and

$$(a_i \oplus b_i) \cdot (\lfloor k_1 \rfloor_m) = (a'_i \oplus b'_i) \cdot (\lfloor k_1 \rfloor_m) \quad .$$

Hence, with this strategy we can successfully impersonate the prover if and only if the observed authentication session was successful.

8.3.4 The HB$^\#$ Protocol

In the light of the little resistance of the above described HB$^+$ variants against GRS-type attacks, Gilbert et al. (2008a) introduced HB$^\#$ as an improvement of HB$^+$ that is provably resistant against the GRS attack. It can be seen as a compressed version of HB$^+$ and works as follows.

Verifier (Alice) and prover (Bob) share two secret matrices K_1 and K_2. Bob sends an l_2-bit blinding value $b \in_R \{0,1\}^{l_2}$ to Alice, who replies with an l_1-bit challenge $a \in_R \{0,1\}^{l_1}$. Bob then chooses an m-bit vector $\nu = (\nu_1, \ldots, \nu_m)$ such that $\Pr[\nu_i = 1] = \eta$ for $i \in [1, m]$ and transmits to Alice as corroborating evidence the value $z = aK_1 \oplus bK_2 \oplus \nu$. Alice accepts an evidence z if and only if $\mathrm{wt}(a \cdot K_1 \oplus b \cdot K_2 \oplus z) \leq t$ for some threshold t (see Fig. 8.7).

In contrast to HB$^+$, an authentication session based on HB$^\#$ consists only of a single round. Hence, the protocol is similar to m executions of HB$^+$ with individual secrets in each basic round.

Although provably resistant against the GRS attack and certain extensions, HB$^\#$ has been shown by Ouafi et al. (2008) not to resist general man-in-the-middle attacks. Particularly, Ouafi et al. show how to deduce the Hamming weight of the error vector ν in a particular authentication session, which can in turn be used to set up a system of linear equations to recover K_1 and K_2. The attack has resonable success probabilities for practical parameter choices of HB$^\#$ and is generally applicable also to other members of the HB protocol family.

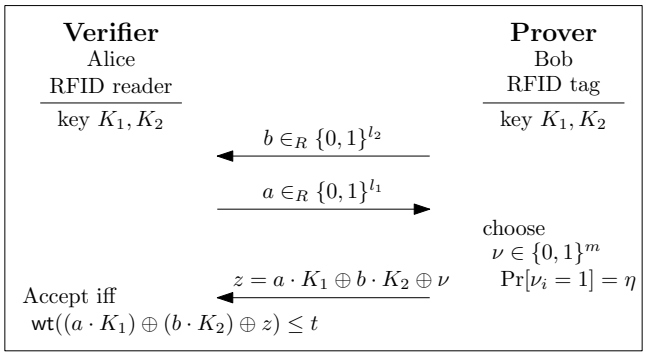

Figure 8.7: One round of the HB$^\#$ protocol

8.3.5 The Trusted-HB Protocol

A generic way to prevent man-in-the-middle attacks on HB$^+$-like protocols is to have the prover send a signature of his view of the communication transcript at the end of the authentication protocol. An adversary who is not able to forge the signature will thereby be prevented from impersonating the prover. Obviously, the choice of signature schemes is restricted by the resource contraints of the HB$^+$ target application environment, which rules out standard message authentication schemes (that could by themselves be used for entity authentication, see Section 7.4). The proposal Trusted-HB by Bringer and Chabanne (2008) tries to solve this trade-off by using a family of universal hash functions that are represented by Toeplitz matrices, particularly by a subset of Toeplitz matrices that can be generated by LFSRs.

Definition 8.2. *A finite collection $\mathcal{H}$ of hash functions $h : \{0,1\}^m \to \{0,1\}^n$ is called* family of universal hash functions *if for each pair of values $x, y \in \{0,1\}^m$, the number of hash functions $h \in \mathcal{H}$ for which $h(x) = h(y)$ is precisely $\frac{|\mathcal{H}|}{m}$, i.e., for a randomly chosen $h \in \mathcal{H}$, $\Pr[h(x) = h(y)] = \frac{1}{m}$ for all $x, y \in \{0,1\}^m$.*

Definition 8.3. *An $(n \times m)$ Boolean* Toeplitz matrix *U contains a fixed value in each left-to-right diagonal, i.e., U is a Toeplitz matrix if $U_{i,j} = U_{i+k,j+k}$ for every $0 \leq i, i+k < n$ and $0 \leq j, j+k < m$.*

Mansour et al. (1990) showed that a family $\mathcal{H}$ of universal hash functions can be constructed by representing the $h \in \mathcal{H}$ as Toeplitz matrices U and computing $h(M)$ as $h(M) := U \circ M$ with M written as $(m \times 1)$ matrix.

The idea of Krawczyk (1994), which is also used in Trusted-HB, was to restrict the family of Toeplitz matrices to those whose consecutive columns can be represented as the consecutive states of an LFSR with irreducible connection polynomial. This restriction trades off reduced security guarantees and compact matrix representations, which are especially useful in resource-constrained environments. The signature for a message $M \in \{0,1\}^m$ is then computed as $\mathrm{MAC}(M) := h(M) \oplus e^{(i)}$, where $e^{(i)} \in \{0,1\}^n$ denotes the i-th unused one-time pad, while h and $e^{(i)}$, $i \geq 0$, are the secret key shared by prover and verifier.

Therefore Trusted-HB consists of two stages:

1. Prover and verifier execute the standard HB$^+$ protocol.

2. The prover computes a signature on the communication transcript of the first stage based on LFSR-based Toeplitz matrices and transmits it to the verifier for verification.

However, the particular implementation of step (2) in Trusted-HB turned out to be flawed (Frumkin and Shamir, 2009), particularly because it seems hard in practice to keep h confidential and provide values $e^{(i)}$ that are sufficiently close to the one-time pad assumption. How the signature can be implemented in a both a secure and efficient way is therefore an open problem to the present day.

Chapter 9

The (n, k, L) Family of Authentication Protocols

9.1 Introduction and Overview

As a possible alternative to HB-type protocols, another class of lightweight authentication protocols (so-called CKK protocols) were introduced by Cichoń et al. (2008). These protocols can be generalized to linear (n, k, L) protocols, in which the secret key consists of the specification of L n-dimensional linear subspaces $V_1, \ldots, V_L$ of $\mathrm{GF}(2)^{n+k}$, while the identification is performed by collaboratively generating an element $v \in V_l$ for a random $l \in \{1, \ldots, L\}$. Cichoń et al. (2008) suggested the CKK^2 protocol, a special linear $(n, k, 2)$ protocol, and the $\mathsf{CKK}^{\sigma,L}$ protocol, a special linear (n, k, L) protocol, for practical application.

Compared to HB-type protocols, the advantages of (n, k, L) protocols and especially their improvements $(n, k, L)^+$ and $(n, k, L)^{++}$ are that fewer bits have to be communicated, computational effort and memory requirements are lower on the prover's side (essentially, the prover has to generate random elements from L different n-dimensional subspaces of $\mathrm{GF}(2)^{n+k}$), and that (n, k, L)-type protocols seem to be more resistant against active attacks. The drawback is that we cannot prove the security of (n, k, L) protocols by reduction to a well-established problem like the LPN-problem yet. However, we show that similarly to HB-type protocols, the security of (n, k, L)-type protocols can be related to the hardness of a certain learning problem, the *Learning Unions of L linear subspaces* (LULS) problem.

We have experimentally confirmed the correctness and efficiency of our attacks and algorithms with the computer algebra system Magma (Bosma et al., 1997).

9.2 The Linear (n, k, L) Protocol

In a linear (n, k, L) protocol, verifier (Alice) and prover (Bob) share as common secret the specifications of L injective linear functions $F_1, \ldots, F_L : \mathrm{GF}(2)^n \longrightarrow \mathrm{GF}(2)^{n+k}$, i.e., each F_i corresponds to an n-dimensional subspace V_i of $\mathrm{GF}(2)^{n+k}$.

After receiving an arbitrary challenge from Alice, Bob computes as cor-

roborating evidence an element $w = F_l(u)$ for $l \in_R [L]$ and $u \in_R \mathrm{GF}(2)^n$. Alice accepts an evidence w if there is an $l \in [L]$ such that $w \in V_l$, where $[L] := \{1, \ldots, L\}$ (see Fig. 9.1).

Obviously, this protocol is vulnerable to a simple passive attack, since an adversary can store a number of proofs and then impersonate Bob by presenting these proofs to Alice.

Moreover, an active adversary can successfully recover the key as follows.

1. Collect a set of messages $O = \{v^1, \ldots, v^s\}$ sent by Bob, with s large enough for O to contain a basis for V_l for all $l \in [L]$ with high probability.

2. Construct an $s \times s$-matrix M over $\{0, 1\}$, where $M_{i,j} = 1$ iff Alice accepts $v^i \oplus v^j$.

Note that if v^i and v^j belong to the same subspace V_l, $\Pr[M_{i,j} = 1] = 1$. If $\{v^i, v^j\} \not\subseteq V_l$ for all $l \in [L]$, then

$$\Pr[M_{i,j} = 1] = \Pr\left[v_i \oplus v_j \in \bigcup_{l=1}^{L} V_l\right] \leq (L-2)2^{-k} \ .$$

The expected number of messages needed for constructing O_r can be estimated based on the following experiment.

Set $B := \emptyset$.
repeat
  Choose a random $v \in \mathrm{GF}(2)^n$ (w.r.t. the uniform distribution).
  $V := V \cup \{v\}$.
until V is a generating system of $\mathrm{GF}(2)^n$.

Lemma 9.1 (Gołębięwski et al. (2008)). *Consider the experiment of repeatedly choosing a random element $v \in \mathrm{GF}(2)^n$ and adding v to an initially empty set V until V contains a generating system of $\mathrm{GF}(2)^n$. Let $p(n)$ denote the probability that the experiment stops after n iterations (i.e., V is a basis of $\mathrm{GF}(2)^n$), and $E(n)$ denote the expected number of iterations of the experiment. Then $p(n) \approx 0.2887$ and $E(n) \approx n + 1.6067$.*

Hence, $s \in \Theta(L \cdot E(n)) = \Theta(Ln)$, i.e., it is possible to efficiently compute specifications of $V_1, \ldots, V_L$ and to impersonate Bob by replying with $w \in V_l$ for arbitrary $l \in [L]$.

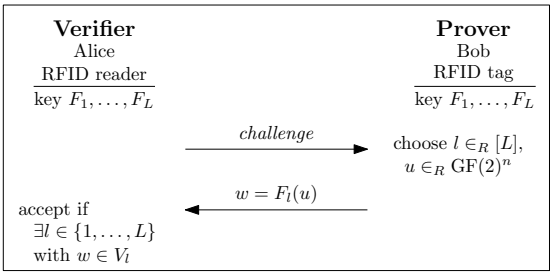

Figure 9.1: Basic round of the (n, k, L) protocol

9.3 The Linear $(n, k, L)^+$ Protocol

In order to prevent the described attacks on the linear (n, k, L) protocol, we consider the following communication mode, which, analogously to the HB^+ protocol (see Section 8.2), defines $(n, k, L)^+$ protocols.

Alice starts by sending an $a \in_R \mathrm{GF}(2)^{n/2}$ to Bob. Bob chooses values $b \in_R \mathrm{GF}(2)^{n/2}$ and $l \in_R [L]$ and sends $w = F_l(a, b)$ to Alice. Alice accepts a $w \in \mathrm{GF}(2)^{n+k}$ if there is some $l \in [L]$ with $w \in V_l$ and the prefix of length $n/2$ of $F_l^{-1}(w)$ is equal to a (see Fig. 9.2).

However, $(n, k, L)^+$ protocols can be broken by the man-in-the-middle attack outlined in Algorithm 12. In this attack, s is chosen large enough for $\{w_1, \ldots, w_s\}$ to contain a basis of V_l with high probability (see Lemma 9.1). The attack is repeated until specifications of all $V_1, \ldots, V_L$ have been computed.

Algorithm 12 $(n, k, L)^+$_MITM-Attack(n, k, L)

Fix $a_1 \neq \vec{0}$ in $\mathrm{GF}(2)^{n/2}$.
Send a_1 to Bob and receive $w_1 \in V_l$ for some unknown $l \in [L]$.
for $r = 2, \ldots, s$ **do**
  repeat
    Intercept a from Alice.
    Send $a' := a \oplus a_1$ to Bob and receive w'.
  until Alice accepts $w' \oplus w_1$ (which happens with probability at least $1/L$)
  Define $a_r := a'$ and $w_r := w'$.
end for
return $\{w_1, \ldots, w_s\}$ (which allows to compute V_l)

9.4 The Linear $(n, k, L)^{++}$ Protocol

The parameters n, k, L as well as V_l, F_l for $l \in [L]$ are defined as above. Let $n = 2N$. The $(n, k, L)^{++}$ protocol works similarly to the $(n, k, L)^+$ protocol, but uses an additional publicly known invertible function $f : \mathrm{GF}(2)^n \longrightarrow \mathrm{GF}(2)^n$, which we call connection function.

In a basic protocol round, Alice chooses a random $a \in \mathrm{GF}(2)^N$, $a \neq \vec{0}$, moves to inner state a, and sends a to Bob. Bob chooses random values $b \in \mathrm{GF}(2)^N$

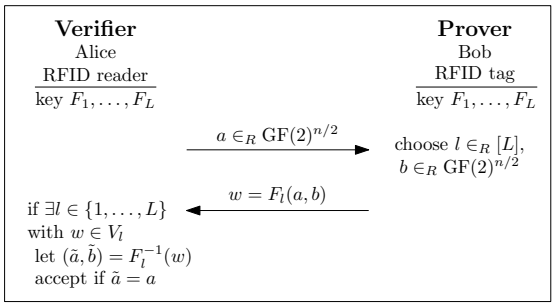

Figure 9.2: Basic round of the $(n, k, L)^+$ protocol

and $l \in [L]$ and sends $w = F_l(f(a, b))$ back to Alice. Alice accepts a message $w \in \mathrm{GF}(2)^n$ in inner state a if $w \neq \vec{0}$, and $\exists l \in [L]$ such that $w \in V_l$, and $f^{-1}(F_l^{-1}(w))$ has the form (a, b) for some $b \in \mathrm{GF}(2)^N$. The basic protocol round of $(n, k, L)^{++}$ is illustrated in Fig. 9.3. Note that choosing f to be the identity yields the $(n, k, L)^{+}$ protocol.

For the $(n, k, L)^{++}$ protocol, we consider a special type of man-in-the-middle attack which we call (x, y)-*equality attack*. The aim of an (x, y)-equality attacker Eve is to generate two messages $w \neq w' \in \mathrm{GF}(2)^{n+k}$ and to efficiently test by man-in-the-middle access to the protocol whether w and $w \oplus w'$ belong to the same linear subspace V_l for some $l \in [L]$. As described above, such an attack can be used to efficiently compute specifications of the subspaces $V_1, \ldots, V_L$. Eve works in three phases:

1. Send a message $y \in \mathrm{GF}(2)^N$ to Bob and receive $w' = F_l(f(y, b'))$.
2. Observe a challenge $a \in \mathrm{GF}(2)^N$ sent by Alice.
3. Compute a value $x = x(y, w', a) \in \mathrm{GF}(2)^N$, send it to Bob, receive the message $w = F_r(f(x, b))$, and send $w \oplus w'$ to Alice.

The success probability of the attack is equal to the probability that Alice accepts $w \oplus w'$ given that $l = r$. Note that if f is $\mathrm{GF}(2)$-linear (as in the $(n, k, L)^{+}$ protocol), setting $x = a \oplus y$ yields an attack with success probability one.

We now define a connection function which yields provable security against (x, y)-equality attacks. In the following we identify $\{0, 1\}^N$ with the finite field $K = \mathbb{F}_{2^N}$ and denote by $+, \cdot$ the addition and multiplication in K. We define a connection function f by

$$\begin{array}{rcl} f : K \times K & \to & K \times K \\ (a, b) & \mapsto & (ab, ab^3) \end{array} \quad . \tag{9.1}$$

Hence, Alice accepts a message w with $F_l^{-1}(w) = (u, v) \in K^2$ in inner state $a \in K^*$ if $(a^{-1}u)^3 = a^{-1}v$, which is equivalent to $u^3 = a^2 v$.

Theorem 9.2. *The success probability of an (x, y)-equality attacker against the $(n, k, L)^{++}$ protocol with connection function f defined in Eq. (9.1) is at most $\frac{3}{2^N - 1}$.*

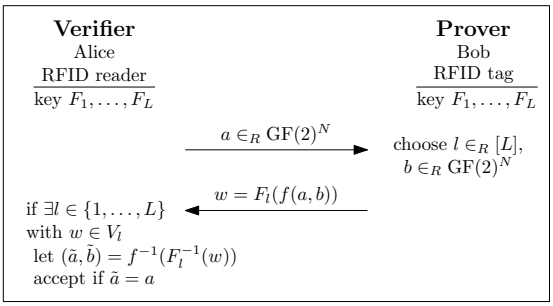

Figure 9.3: Basic round of the $(n, k, L)^{++}$ protocol

Proof. For given $y, a \in K^*$, Eve has to choose an element $x \in K^*$ such that $w + w' = (u, v) \in K \times K$ will be accepted by Alice in inner state a, where $w = F_l(x, b)$ and $w' = F_l(y, b')$ for some $l \in [L]$, and $b, b' \in K^*$. Note that Eve has no information about b, b', and that $u = xb + yb'$ and $v = xb^3 + yb'^3$.

Consequently, Eve's choice for the value x has to satisfy

$$\begin{aligned} & (xb + yb')^3 = a^2(xb^3 + yb'^3) \\ \Leftrightarrow \quad & (x + yc)^3 = a^2(x + yc^3) \text{ with } c := b'(b^{-1}) \\ \Leftrightarrow \quad & P(x, c) = 0, \end{aligned}$$

with $P(x, d)$ for all $d \in K^*$ defined as

$$P(x, d) := x^3 + (yd)x^2 + (y^2d^2 + a^2)x + d^3(y^3 + y^2a^2) \ .$$

Note that there are $|K^*| = 2^N - 1$ different polynomials $P(x, d)$ with respect to the variable x. For all $x \in K^*$ let $P(x) := \{d | P(x, d) = 0\}$. Note that $P(x, d)$ is a polynomial of degree 3 also in the unknown d. This implies that $|P(x)| \leq 3$ for all $x \in K^*$.

Eve has to choose an x that satisfies $c \in P(x)$. Since she does not have any information about c, her success probability is at most $\frac{3}{2^N - 1}$. □

9.5 Special Cases of Linear (n, k, L) Protocols

The definition of the (n, k, L) protocol family was inspired by two earlier proposals, the CKK^2 protocol and the $\mathsf{CKK}^{\sigma,L}$ (Cichoń et al., 2008) protocol, which can be seen as restricted (n, k, L) protocols.

In our notation, the protocol CKK^2 is an $(n + k, k, 2)$ protocol with the additional properties that $F_1(u, a) = (u, f(u), a)$ and $F_2(u, a) = (u, a, f(u))$ for all $u \in \mathrm{GF}(2)^n$ and $a \in \mathrm{GF}(2)^k$, where f denotes a secret linear function $f : \mathrm{GF}(2)^n \longrightarrow \mathrm{GF}(2)^k$. The protocol $\mathsf{CKK}^{\sigma,L}$ is an (n, k, L) protocol with the restriction $F_l(u) = \sigma^l(u||f(u))$ for all $l \in [L]$, where σ denotes a secret permutation $\sigma \in \mathcal{S}_{n+k}$ and f a secret linear function $f : \mathrm{GF}(2)^n \longrightarrow \mathrm{GF}(2)^k$. Hence, the secret keys have the form (f, σ). The parameters $n = 128$ and $k = 30$ were suggested by Cichoń et al. (2008) for practical applications of CKK^2 and $\mathsf{CKK}^{\sigma,L}$.

Gołębiewski et al. (2008) presented an attack against the CKK^2 protocol, which cannot be applied to general (n, k, L) protocols. Its running time is proportional to $\sum_{s=0}^{k-1} \binom{n}{s}$, i.e., of order $n^{\Theta(k)}$. As an improvement of this result, we now describe a very fast attack against the CKK^2 protocol with parameters (n, k) whose running time is dominated by the effort required for inverting k $(n \times n)$-matrices.

Let $f : \mathrm{GF}(2)^n \longrightarrow \mathrm{GF}(2)^k$ denote the secret key and recall that

$$\begin{aligned} V_1 &= \{(v, f(v), a),\ v \in \mathrm{GF}(2)^n, a \in \mathrm{GF}(2)^k\} \ , \\ V_2 &= \{(v, a, f(v)),\ v \in \mathrm{GF}(2)^n, a \in \mathrm{GF}(2)^k\} \ . \end{aligned}$$

Let the functions $f^1, \ldots, f^k : \mathrm{GF}(2)^n \longrightarrow \mathrm{GF}(2)$ denote the component functions of the secret function f, i.e., $f(v) = (f^1(v), \ldots, f^k(v))$ for all $v \in \mathrm{GF}(2)^n$. The attack is based on the simple fact that if an observation (v, a, b) satisfies

Algorithm 13 CKK^2_Attack(n, k)

Let $\{e_1, \ldots, e_n\}$ denote the standard basis of $\mathrm{GF}(2)^n$.
for $r \in [k]$ **do**
  Consider a set of messages produced by Bob and extract from it a set $O_r = \{(v_{r,1}, a_{r,1}, b_{r,1}), \ldots, (v_{r,n}, a_{r,n}, b_{r,n})\}$ such that $v_{r,1}, \ldots, v_{r,n}$ form a basis of $\mathrm{GF}(2)^n$ and $a_{r,i}(r) = b_{r,i}(r) = f^r(v_{r,i})$ for all $i \in [n]$.
  Derive $f^r(e_1), \ldots, f^r(e_n)$ from O_r.
  return $f^1, \ldots, f^k$
end for

$a_r = b_r$ for some $r \in [k]$, which is true with probability $1/2$, then we know that $f^r(v) = a_r = b_r$. The attack works as described in Algorithm 13.

The correctness of the attack follows straightforwardly from the definitions. Lemma 9.1 implies that the expected number of messages needed for constructing O_r is $2 \cdot E(n) \approx 2n + 3.2134$. For the parameter choices proposed for practical applications, the attack is very efficient already on standard PC hardware (Magma V2.15-9 on a 3.4 GHz Intel Pentium IV with 4 GB RAM), see Table 9.1.

9.6 Security of Linear (n, k, L)-type Protocols and the Learning Unions of L Linear Subspaces Problem

9.6.1 The Search-for-a-Basis Heuristic

There are several exhaustive search strategies for computing specifications of the secret subspaces $V_1, \ldots, V_L$.

As an example, we describe the search-for-a-basis heuristic, which tries to construct a set Q of examples which form a basis of V_l for some $l \in L$. For all linearly independent sets Q of n examples let $p(Q)$ denote the probability that an example coming from the oracle belongs to the linear span $\langle Q \rangle$ of Q. It is quite obvious that $p(Q)$ is maximal if Q is a basis of V_l for some $l \in L$. If $p(Q)$ is not too small, we can compute an approximation $\tilde{p}(Q)$ of $p(Q)$ by testing for $w \in \langle Q \rangle$ for a sufficiently large number of examples w. For $v \in Q$ and $w \notin Q$ we denote by $Q(v, w)$ the set obtained by replacing v by w in Q.

The idea of the heuristic is to start with an arbitrary linearly independent set Q of n examples and to try to improve this set by finding $v \in Q$ and $w \notin Q$ such that $\tilde{p}(Q) < \tilde{p}(Q(v, w))$. Iterating this procedure at most n times yields a basis for V_l for some $l \in [L]$.

Table 9.1: Performance of the passive attack on CKK^2

(n, k)	approx. number of observations	approx. attack time
$(128, 30)$	311	0.3 s
$(1024, 256)$	2197	179 s

This kind of heuristic is infeasible if the following condition is fulfilled. For a random linear independent set Q of n examples the probability $p(Q)$ is negligibly small with probability $1 - \epsilon$, ϵ negligibly small. The parameters n, k should be chosen such that this condition is guaranteed.

We estimate the probability $p(Q)$ for the case $L = 2$. For a linear independent set Q of n examples let $Q = Q_1 \cup Q_2$, where $Q_1 \subseteq V_1$ and $Q_2 \subseteq V_2 \setminus V_1$. Without loss of generality, let $|Q_1| = n/2 + s$ and $|Q_2| = n/2 - s$. The event $w \in \langle Q \rangle$ happens iff $w \in V_1 \cap < Q_1 >$ or $w \in V_2$ and $w \in V_2 \cap < Q_1 >$, i.e.,

$$p(Q) \leq \frac{1}{2}\left(2^{s-n/2} + 2^{-}k\right) \quad .$$

Note that $\dim(V_1 \cap V_2) = n - k$ for random n-dimensional subspaces V_1, V_2. If n, k are chosen such that 2^{-k}, $2^{-n/4}$ and the probability that $|v| \notin [n/4, 3n/4]$ are negligibly small, then the above condition is fulfilled (note that the expected value of s is $2^{-k}n/2$).

The parameters (n, k) should be chosen such that these attacks become infeasible. Moreover, k should be large enough such that the probability p of a random $v \in \mathrm{GF}(2)^{n+k}$ belonging to $\bigcup_{l=1}^{L} V_l$ is negligibly small. Note that $p < L2^{-k}$.

The subspaces $V_1, \ldots, V_L$ should have the property $V_i \oplus V_j = \mathrm{GF}(2)^{n+k}$ for all $i \neq j \in [L]$, otherwise the effective key length would be reduced. This implies $n \geq k$.

9.6.2 The Learning Unions of L Linear Subspaces Problem

The Learning Unions of L Linear Subspaces (LULS) Problem refers to the following communication game between a learner and an oracle. The oracle holds the specifications of L n-dimensionial linear subspaces $V_1, \ldots, V_L$ of $\mathrm{GF}(2)^{n+k}$. The learner can send requests *hello* to the oracle. If the oracle receives *hello*, it chooses randomly and uniformly an $l \in [L]$ and $v \in V_l$ and sends the (positive) example v to the learner. The aim of the learner is to compute specifications of $V_1, \ldots, V_L$ from a sufficiently large set $v^1, \ldots, v^s$ of examples produced by the oracle. Note that this corresponds to a passive key recovery attack against (n, k, L)-type protocols. A possible strategy is the search-for-a-basis heuristic described in Section 9.6.1.

An active adversary who is able to solve the LULS problem efficiently can break the $(n, k, L)^+$ protocol. In particular, knowing specifications of the secret subspaces $V_1, \ldots, V_L$, he can generate specifications of the subspaces $V_l(a)$ (i.e., the image of $F_l(a, \cdot)$), for arbitrary $a \in \mathrm{GF}(2)^{n/2}$ and $l \in [L]$ by repeatedly sending a to Bob. Then the adversary uses $N = n/2$ subspaces $V_l(a_i), \ldots, V_l(a_N)$ for $\{a_1, \ldots, a_N\}$ linearly independent to forge a response for

a challenge $a = \sum_{i=1}^{N} \alpha_i a_i$ by computing

$$\begin{aligned} w &= \sum_{i=1}^{N} \alpha_i v_i \text{ with } v_i \in_R V_l(a_i) \\ &= \sum_{i=1}^{N} \alpha_i F_l(a_i, b_i) \\ &= F_l(a, b') \text{ with } b = \sum_{i=1}^{N} b_i \ . \end{aligned}$$

In the case of the $(n, k, L)^{++}$ protocol, the adversary cannot just return a random $w \in V_l(a)$, but has to make sure that the first half of $f^{-1}(F_l^{-1}(w))$ corresponds to a. How such a w can be found efficiently (possibly based on the specifications of the subspaces $V_l(a)$) is a matter of further research.

In the following, we present and discuss an algebraic learning algorithm for LULS.

9.6.3 On Solving the LULS Problem

A Learning Algorithm for the LULS Problem

Recall that the LULS problem with parameters n, k, L consists in computing specifications of L secret n-dimensional linear subspaces of $\mathrm{GF}(2)^{n+k}$ from positive examples v produced by an oracle which chooses randomly and uniformly $l \in [L]$ and $v \in V_l$. In this thesis we treat the case $L = 2$ and consider the special case that $V_l = \{(v, f(v)), v \in \mathrm{GF}(2)^n\}$, $l \in \{1, 2\}$, for secret linear functions $f_1, f_2 : \mathrm{GF}(2)^n \longrightarrow \mathrm{GF}(2)^k$. Our algorithm computes for all $i \in [k]$ specifications of the i-th component functions $f_1^i, f_2^i : \mathrm{GF}(2)^n \longrightarrow \mathrm{GF}(2)$ separately, i.e., it suffices to consider the case $k = 1$. The learning algorithm is based on the following reasoning.

1. Take a set $O = \{(v^1, w_1), \ldots, (v^n, w_n)\} \subseteq \mathrm{GF}(2)^{n+1}$ of examples such that $B = \{v^1, \ldots, v^n\}$ forms a basis of $\mathrm{GF}(2)^n$. For all $i \in [n]$ let x_i and y_i denote the variables corresponding to $f_1(v^i)$ and $f_2(v^i)$, respectively.

2. For $b \in \{0, 1\}$ let $I_b = \{i \in [n], w_i = b\}$.

3. For all $i \in [n]$ let $t_i = x_i \oplus y_i$, and for all $i < j \in [n]$ let $t_{i,j} = x_i y_j \oplus x_j y_i$.

4. Observe that for all $i \in [n]$ the equality $(w_i \oplus x_i)(w_i \oplus y_i) = 0$ holds. This implies
$$x_i y_i = 0 \text{ if } i \in I_0 \text{ and } x_i y_i = 1 \oplus t_i \text{ if } i \in I_1 \ . \tag{9.2}$$

5. Observe that each example $(v, w) \in \mathrm{GF}(2)^{n+1}$, $v \notin B$ satisfies the following: If $v = \bigoplus_{i \in I} v_i$, (i.e., $I \subseteq [n]$ defines the unique representation of v w.r.t. B), then
$$\left(w \oplus \bigoplus_{i \in I} x_i \right) \left(w \oplus \bigoplus_{i \in I} y_i \right) = 0 \ . \tag{9.3}$$

Observe that Eq. (9.3) can be rewritten as a relation $T_B(I, w)$ in the variables t_i and $t_{i,j}$ in the following way. If $w = 0$ then Eq. (9.3) is equivalent to $\bigoplus_{i\in I} x_i y_i \oplus \bigoplus_{i<j\in I} t_{i,j} = 0$. Together with Eq. (9.2) this implies $\bigoplus_{i\in I_1\cap I}(t_i \oplus 1) \oplus \bigoplus_{i<j\in I} t_{i,j} = 0$ for $w = 0$. Consequently, for $w = 0$ we define $T_B(I, w)$ as

$$\bigoplus_{i\in I\cap I_1} t_i \oplus \bigoplus_{i<j\in I} t_{i,j} = \begin{cases} 0 & \text{if } |I \cap I_1| \text{ is even} \\ 1 & \text{if } |I \cap I_1| \text{ is odd} \end{cases} .$$

If $w = 1$ then Eq. (9.3) is equivalent to $1 \oplus \bigoplus_{i\in I} t_i \oplus \bigoplus_{i\in I\cap I_1}(t_i \oplus 1) \oplus \bigoplus_{i<j\in I} t_{i,j} = 0$. Hence, for $w = 1$ we define $T_B(I, w)$ as

$$\bigoplus_{i\in I\cap I_0} t_i \oplus \bigoplus_{i<j\in I} t_{i,j} = \begin{cases} 0 & \text{if } |I \cap I_1| \text{ is odd} \\ 1 & \text{if } |I \cap I_1| \text{ is even} \end{cases} .$$

Note that a relation similar to Eq. (9.3) was also exhibited by Blass et al. (2008) for designing an algebraic attack against F_f protocols.

The learning algorithm now proceeds as described in Algorithm 14.

Algorithm 14 LULS-solve(O)

Let initially the system LES of linear equations in the $\frac{1}{2}(n^2 + n)$ variables t_i ($i \in [n]$) and $t_{i,j}$ ($i < j \in [n]$) be empty.
repeat
  Choose an observation $(v, w) \in O$, $v \notin B \cup \{\vec{0}\}$, and compute the unique subset $I \subseteq [n]$ with $v = \bigoplus_{i\in I} v^i$.
  Enlarge the system LES by the linear equation $T_B(I, w)$.
until the system LES has $\frac{1}{2}(n^2 + n)$ linearly independent equations.
Compute by Gaussian elimination the unique solution θ of the system LES.
Compute from θ the unique correct assignments to x_i, y_i for all $i \in [n]$.

The correct assignments to the x_i and y_i variables (the last step of Algorithm 14) can be computed from $\theta = (\theta_i)_{i\in[n]}\,(\theta_{i,j})_{i<j\in[n]}$ as follows. For $b = 0, 1$ let K_b denote the set $K_b = \{i \in [n], \theta_i = b\}$. We know that for all $i \in K_0$, $x_i = y_i = w_i$ is satisfied, and for all $i \in K_1$ it holds that $y_i = x_i \oplus 1$. This implies that for all $i < j$ in K_1, $\theta_{i,j}$ satisfies

$$\theta_{i,j} = x_i(x_j \oplus 1) \oplus x_j(x_i \oplus 1) = x_i \oplus x_j \ .$$

This yields a system LES^* of $1/2|K_1|(|K_1|-1)$ linear equations in the variables x_i, $i \in K_1$, of rank $|K_1| - 1$. Since it does not matter which of the two secret linear subspaces we denote by V_1 and which by V_2, we have the freedom to set $x_k = 0$ for some fixed $k \in K_1$. The system LES^* together with $x_k = 0$ yields a system of full rank and allows to compute the correct assigment to the x_i-variables by Gaussian elimination.

Analysis and Experimental Results

The reason for the fact that the repeat cycle of the algorithm is left after a finite number of rounds is that the following $(2^n - (n + 1)) \times (n(n + 1)/2)$-matrix $M(n)$ over GF(2) has full row rank (which is not hard to show). The

row indices of $M(n)$ are all subsets $I \subseteq [n]$ with $|I| \geq 2$, the column indices are $[n] \cup \{(i,j), 1 \leq i < j \leq n\}$. We have $M(n)_{I,i} = 1$ iff $i \in I$ and $M(n)_{I,(i,j)} = 1$ iff $\{i,j\} \subseteq [n]$.

We do not give here a theoretical analysis of the expected number of rounds of the repeat cycle. Our experiments show that the algorithm needs only slightly more than $\frac{1}{2}(n^2+n)+n$ observations to compute the secret functions f_1 and f_2. Particularly for $n = 128$, we need approx. 8390 examples and 4 minutes on a 3.4 GHz Intel Pentium IV with 4 GB RAM and Magma V2.15-9.

How severe is the restriction that the secret subspaces have the special form $V = \{(v, f(v)), v \in \mathrm{GF}(2)^n\}$ for some surjective linear mapping $f : \mathrm{GF}(2)^n \longrightarrow \mathrm{GF}(2)^k$? Let us consider the general case $V = \{A \circ v, v \in \mathrm{GF}(2)^n\}$ for an $((n+k) \times n)$ matrix A. V can be written in the special form iff the first n rows of A are linearly independent. For randomly chosen A this is true with probability $p(n) \approx 0.2887$ (Lemma 9.1).

We have seen that we can solve the LULS problem with parameters $(n,k,2)$ by solving k LULSproblems with parameters $(n,1,2)$. For the special LULS problem with parameters $(n,1,L)$, $L > 2$, we can define a similar system LES consisting of degree-L equations in the variables x_i^l, $i \in [n]$, $l \in [L]$, induced as above by equations of the form

$$\left(w \oplus \bigoplus_{i \in I} x_i^1\right) \ldots \left(w \oplus \bigoplus_{i \in I} x_i^L\right) = 0 \ . \tag{9.4}$$

The problem is that for $L > 2$ the equations have several symmetries such that the system can not be solved uniquely. A possible way out is to

- choose an appropriate parameter $s < k$ which divides k, let $k = s \cdot p$,
- write vectors $w \in \mathrm{GF}(2)^k$ as vectors $w \in \mathrm{GF}(2^s)^p$, and
- solve the corresponding p LULS problem with parameters $(n,1,L)$ over $\mathrm{GF}(2^s)$.

Hamann (2010) has described a learning algorihm based on this idea that solves the described special case of the LULS problem in average running time in the order of $kn^{\mathcal{O}(L)}$. His analysis supports the conjecture that there is no faster way to solve an (n,k,L) LULS problem, which suggests parameter choices like $(n,L) \in \{(128,8),(256,6)\}$ for practical applications.

9.7 Discussion

We have seen that the secret key of CKK^2 protocols can be computed very quickly from a sufficiently large set of messages sent by the prover. This kind of protocol should not be used in practice.

The parameters of $(n,k,L)^{++}$ protocols have to be chosen in such that solving the LULS problem with parameters $(\frac{n}{2},k,L)$ is infeasible. We recommend to use $n = 256$, $k = 64$ and $L = 5$.

Another interesting question is to search for simpler nonlinear connection functions f for which a security proof can be found. In our proposal, the prover has to perform three multiplications in the finite field of order $2^{n/2}$ in order to compute $f(a,b)$.

Yet another open question is whether the very symmetrically structured systems of degree-L equations arising in our LULS algorithm in Section 9.6.3 can be solved more efficiently by more advanced techniques like the F4- or F5-algorithm or cube attacks (Dinur and Shamir, 2008, 2009, Faugère, 1999, 2002). If one could generate convincing evidence that such algorithms cannot beat our linearization attack, then $(n, k, L)^{++}$ protocols with the above parameters could be seriously considered for practical use.

A problem of (n, k, L) protocols is the large key length of $L \cdot n \cdot n + k$ in the case that random mappings $F_1, \ldots, F_L$ are used. It is an important task to look for secure and efficient ways to generate pseudorandom keys. In this context, the (still unbroken) $\mathsf{CKK}^{\sigma,L}$ protocols look appealing, but we conjecture that $\mathsf{CKK}^{\sigma,L}$ protocols can be efficently broken. However, promising suggestions for key length reductions have been made by Gilbert et al. (2008a) and Bringer and Chabanne (2008) in the context of Trusted-HB (see Section 8.3.5). Adapting these ideas to (n, k, L) protocols would mean

- to consider special forms of secret subspaces $V_l = \{(A_l \circ v), v \in \mathrm{GF}(2)^n\}$, where A_l denotes a secret $(n + k) \times n$ Toeplitz matrix (Gilbert et al., 2008a), and
- to define the Toeplitz matrix A_l to be generated by a secret Linear Feedback Shift Register (Bringer and Chabanne, 2008).

Checking the feasibility and security of these constructions should be a matter of further research.

Chapter 10

Conclusion

In this thesis, we have analyzed two of the most prominent security requirements in electronic communication, confidentiality of messages and authenticity of entities.

Concerning confidentiality of messages, we have defined and analyzed hardware-oriented stream ciphers and their most important building blocks. We have described three generic attacks on stream ciphers, BDD-Attacks, correlation attacks and algebraic attacks, and analyzed their impact on practically used stream ciphers as well as newly proposed designs. In the case of the E_0 keystream generator from the Bluetooth standard, we have indicated ways to improve its security with respect to the considered attacks by careful local modifications of the design.

In order to provide entity authentication for environments in which only little computational resources are available, e.g. on RFID-tags or mobile telephones, we defined and investigated lightweight authentication protocols that are based on randomly choosing elements from a secret set of vector spaces. We related the security of these protocols to the hardness of a certain learning problem and provided a first complexity analysis of this problem as a starting point for further research.

Bibliography

Carlisle Adams. Identification. In Henk Tilborg, editor, *Encyclopedia of Cryptography and Security*, pages 272–273. Springer US, 2005.

Miklós Ajtai, László Babai, Péter Hajnal, János Komlós, Pavel Pudlák, Vojtech Rödl, Endre Szemerédi, and György Turán. Two lower bounds for branching programs. In *Proc. of STOC '86*, pages 30–38. ACM, 1986.

Ross Anderson, Francesco Bergadano, Bruno Crispo, Jong-Hyeon Lee, Charalampos Manifavas, and Roger Needham. A new family of authentication protocols. *SIGOPS Oper. Syst. Rev.*, 32:9–20, 1998.

Frederik Armknecht. Improving fast algebraic attacks. In *Proc. of FSE 2004*, volume 3017 of *LNCS*, pages 65–82. Springer, 2004a.

Frederik Armknecht. On the existence of low-degree equations for algebraic attacks. Technical report, Cryptology ePrint Archive, Report 2004/185, 2004b.

Frederik Armknecht. *Algebraic Attacks on Certain Stream Ciphers*. PhD thesis, University of Mannheim, Mannheim, Germany, 2006.

Frederik Armknecht and Gwenolé Ars. Algebraic attacks on stream ciphers with Gröbner bases. In *Gröbner Bases, Coding, and Cryptography*, pages 329–348. Springer, 2009.

Frederik Armknecht and Matthias Krause. Algebraic attacks on combiners with memory. In *Proc. of CYPTO 2003*, volume 2729 of *LNCS*, pages 162–176. Springer, 2003.

Frederik Armknecht and Matthias Krause. Constructing single- and multi-output boolean functions with maximal algebraic immunity. In *Proc. of ICALP 2006*, volume 4052 of *LNCS*, pages 180–191. Springer, 2006.

Frederik Armknecht, Joseph Lano, and Bart Preneel. Extending the resynchronization attack. In *Proc. of SAC 2004*, volume 3357 of *LNCS*, pages 19–38. Springer, 2004.

Frederik Armknecht, Claude Carlet, Philippe Gaborit, Simon Künzli, Willi Meier, and Olivier Ruatta. Efficient computation of algebraic immunity for algebraic and fast algebraic attacks. In *Proc. of EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 147–164. Springer, 2006.

François Arnault and Thierry P. Berger. Design and properties of a new pseudo-random generator based on a filtered FCSR automaton. *IEEE Trans. Comp.*, 54(11):1374–1383, 2005a.

François Arnault, Thierry P. Berger, and Abdelkader Necer. Feedback with carry shift register synthesis with the euclidean algorithm. *IEEE Trans. Inform. Theory*, 50(5):910–917, 2004.

François Arnault, Thierry P. Berger, and Cédric Lauradoux. Update on F-FCSR stream cipher. eSTREAM, ECRYPT Stream Cipher Project, Report 2006/025, 2006. `http://www.ecrypt.eu.org/stream`.

François Arnault, Thierry P. Berger, and Marine Minier. Some results on FCSR automata with applications to the security of FCSR-based pseudorandom generators. *IEEE Trans. Inform. Theory*, 54(2):836–840, 2008.

François Arnault and Thierry P. Berger. F-FCSR: Design of a new class of stream ciphers. In *Proc. of FSE 2004*, volume 3557 of *LNCS*, pages 83–97. Springer, 2005b.

François Arnault, Thierry P. Berger, Cédric Lauradoux, and Marine Minier. X-FCSR – a new software oriented stream cipher based upon FCSRs. In *Proc. of INDOCRYPT 2007*, volume 4859 of *LNCS*, pages 341–350. Springer, 2007.

Jean-Philippe Aumasson, Itai Dinur, Willi Meier, and Adi Shamir. Cube testers and key recovery attacks on reduced-round MD6 and Trivium. In *Proc. of FSE 2008*, volume 5665 of *LNCS*, pages 1–22. Springer, 2009.

Steve Babbage, Christophe de Cannière, Anne Canteaut, Calos Cid, Henri Gilbert, Thomas Johansson, Matthew Parker, Bart Preneel, Vincent Rijmen, and Matthew J.B. Robshaw. The eSTREAM portfolio (rev. 1). eSTREAM, ECRYPT Stream Cipher Project, 2008. `http://www.ecrypt.eu.org/stream`.

Alex Biryukov and Adi Shamir. Cryptanalytic time/memory/data tradeoffs for stream ciphers. In *Proc. of ASIACRYPT 2000*, volume 1976 of *LNCS*, pages 1–13. Springer, 2000.

Alex Biryukov, Adi Shamir, and David Wagner. Real time cryptanalysis of A5/1 on a PC. In *Proc. of FSE 2000*, volume 1978 of *LNCS*, pages 1–13. Springer, 2000.

John Black, Phillip Rogaway, and Thomas Shrimpton. Black-box analysis of the block-cipher-based hash-function constructions from PGV. In *Proc of CRYPTO 2002*, volume 2442 of *LNCS*, pages 103–118. Springer, 2002.

Erik-Oliver Blass, Anil Kurmus, Refik Molva, Guevara Noubir, and Abdullatif Shikfa. The F_f-family of protocols for RFID-privacy and authentication. `http://eprint.iacr.org/2008/476`, 2008.

Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *J. ACM*, 50(4):506–519, 2003.

Lenore Blum, Manuel Blum, and Michael Shub. A simple unpredictable pseudo-random number generator. *SIAM J. Comput.*, 15(1):364–383, 1986.

Andrey Bogdanov, Lars Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J.B. Robshaw, Yannick Seurin, and Charlotte Vikkelsoe. PRESENT: An ultra-lightweight block cipher. In *Proc. of CHES 2007*, volume 4727 of *LNCS*, pages 450–466. Springer, 2007.

Wieb Bosma, John Cannon, and Catherine Playoust. The MAGMA algebra system. i. the user language. *J. Symbolic Comput.*, 24(3-4):235–265, 1997.

An Braeken and Joseph Lano. On the (im)possibiliy of practical and secure nonlinear filters and combiners. In *Proc. of SAC 2005*, volume 3897 of *LNCS*, pages 159–174. Springer, 2005.

Jörg Brandeis. Implementierung eines algebraischen Angriffs auf den E_0-Generator und verwandte Chiffren. Master's thesis, University of Mannheim, Mannheim, Germany, 2004. (in german).

Marc Briceno, Ian Goldberg, and David Wagner. *A pedagogical implementation of A5/1*, May 1999. `http://jya.com/a51-pi.htm`.

Julien Bringer and Hervé Chabanne. Trusted-HB: A low cost version of HB^+ secure against a man-in-the-middle attack. *IEEE Trans. Inform. Theor.*, 54: 4339–4342, 2008.

Julien Bringer, Hervé Chabanne, and Emmanuelle Dottax. HB^{++}: A lightweight authentication protocol secure against some attacks. In *Proc. of SecPerU*, pages 28–33. IEEE Computer Society Press, 2006.

Randal E. Bryant. Graph-based algorithms for Boolean function manipulation. *IEEE Trans. Comp.*, 35(8):677–691, 1986.

Claude Carlet. A method of construction of balanced functions with optimum algebraic immunity. In *Proc. of International Workshop on Coding and Cryptology*, Coding and Cryptology, pages 25–43. World Scientific, 2008.

Jacek Cichoń, Marek Klonowski, and Mirosław Kutyłowski. Privacy protection for RFID with hidden subset identifiers. In *Proc. of Pervasive 2008*, volume 5013 of *LNCS*, pages 298–314. Springer, 2008.

Don Coppersmith, Hugo Krawczyk, and Yishay Mansour. The shrinking generator. In *Proc. of CRYPTO 1993*, volume 773 of *LNCS*, pages 22–39. Springer, 1994.

Nicolas Courtois. Fast algebraic attacks on stream ciphers with linear feedback. In *Proc. of CRYPTO 2003*, volume 2729 of *LNCS*, pages 177–194. Springer, 2003.

Nicolas Courtois and Willi Meier. Algebraic attacks on stream ciphers with linear feedback. In *Proc. of EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 345–359. Springer, 2003.

Nicolas Courtois, Alexander Klimov, Jaques Patarin, and Adi Shamir. Efficient algoprithms for solving overdefined systems of multivariate polynomial equations. In *Proc. of EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 392–407. Springer, 2000.

Raymond Couture and Pierre L'Ecuyer. On the lattice structure of certain linear congruential sequences related to AWC/SWB generators. *Math. Comput.*, 62 (206):799–808, 1994.

Deepak Kumar Dalai, Kishan Chand Gupta, and Subhamoy Maitra. Cryptographically significant boolean functions: Construction and analysis in terms of algebraic immunity. In *Proc. of FSE 2005*, volume 3557 of *LNCS*, pages 98–111. Springer, 2005.

Ivan Damgård. A design principle for hash functions. In *Proc of CRYPTO '89*, volume 435 of *LNCS*, pages 416–427. Springer, 1990.

Donald Davies and Wyn L. Price. Digital signatures, an update. In *Proc. 5th International Conference on Computer Communication*, pages 845–849, 1984.

Christophe de Cannière and Bart Preneel. TRIVIUM specifications. eSTREAM, ECRYPT Stream Cipher Project, 2005. `http://www.ecrypt.eu.org/stream`.

Christophe De Cannière, Orr Dunkelman, and Miroslav Knežević. KATAN and KTANTAN — a family of small and efficient hardware-oriented block ciphers. In *Proc. of CHES 2009*, volume 5747 of *LNCS*, pages 272–288. Springer, 2009.

Blandine Debraize and Louis Goubin. Guess-and-determine algebraic attack on the self-shrinking generator. In *Proc. of FSE 2008*, volume 5086 of *LNCS*, pages 235–252. Springer, 2008.

Frédéric Didier and Jean-Pierre Tillich. Computing the algebraic immunity efficiently. In *Proc. of FSE 2006*, volume 4047 of *LNCS*, pages 359–374. Springer, 2006.

Tim Dierks and Eric Rescorla. The transport layer security (TLS) protocol version 1.2. RFC 5246, 2008. `http://tools.ietf.org/html/rfc5246`.

Itai Dinur and Adi Shamir. Cube attacks on tweakable black box polynomials. Cryptology ePrint Archive, Report 2008/385, 2008. `http://eprint.iacr.org`.

Itai Dinur and Adi Shamir. Cube attacks on tweakable black box polynomials. In *Proc. of EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 278–299. Springer, 2009.

Danny Dolev and Andrew Yao. On the security of public key protocols. *IEEE Trans. Inform. Theory*, 29:198–208, 1983.

Dang Nguyen Duc and Kwangjo Kim. Securing HB^+ against GRS man-in-the-middle attack. In *Proc. of SCIS 2007*, 2007. `http://koasas.kaist.ac.kr/bitstream/10203/23125/1/SCIS2007_Duc.pdf`.

Orr Dunkelman, Nathan Keller, and Adi Shamir. A practical-time related-key attack on the kasumi cryptosystem used in GSM and 3G telephony. In *Proc. of CRYPTO 2010*, volume 6223 of *LNCS*, pages 393–410. Springer, 2010.

William F. Ehrsam, Carl H. W. Meyer, John L. Smith, and Walter L. Tuchman. Message verification and transmission error detection by block chaining. US Patent 4074066, 1976.

Tobias Eibach. *Generic Attacks on Stream Ciphers*. PhD thesis, Universität Ulm, Ulm, Germany, 2008.

Patrik Ekdahl. *On LFSR Based Stream Ciphers (Analysis and Design)*. PhD thesis, Lund University, Sweden, 2003.

Patrik Ekdahl and Thomas Johansson. Another attack on A5/1. In *Proc. of International Symposium on Information Theory*, page 160. IEEE, 2001.

eStream. eSTREAM, ECRYPT stream cipher project, 2008. `http://www.ecrypt.eu.org/stream`.

eSTREAM Discussion Forum. A reformulation of TRIVIUM. eSTREAM, ECRYPT Stream Cipher Project, Discussion Forum, 2005. `http://www.ecrypt.eu.org/stream/phorum/read.php?1,448`.

Jean-Charles Faugère. A new efficient algorithm for computing Gröbner bases (F4). *Journal of pure and applied algebra*, 139(1-3):61–68, 1999.

Jean-Charles Faugère. A new efficient algorithm for computing Gröbner basis without reduction to zero (F5). In *Proc. of ISSAC 2002*, pages 75–83. ACM Press, 2002.

Jean-Charles Faugère and Gwenole Ars. An algebraic cryptanalysis of nonlinear filter generators using Gröbner bases, 2003. `http://www.inria.fr/rrrt/rr-4739.html`.

Simon Fischer, Willi Meier, and Dirk Stegemann. Equivalent representations of the F-FCSR keystream generator. In *Workshop Record of The State of the Art of Stream Ciphers (SASC 2008)*, 2008.

Ewan Fleischmann, Christian Forler, and Michael Gorski. Classification of the SHA-3 candidates. Cryptology ePrint Archive, Report 2008/511, 2008. `http://eprint.iacr.org/`.

Scott R. Fluhrer and Stefan Lucks. Analysis of the E_0 encryption system. In *Proc. of SAC 2001*, volume 2259 of *LNCS*, pages 38–48. Springer, 2001.

Marc Fossorier, Miodrag Mihaljević, Hideki Imai, Yang Cui, and Kanta Matsuura. An algorithm for solving the LPN problem and its application to security evaluation of the HB protocols for RFID authentication. In *Proc. of INDOCRYPT 2006*, volume 4329 of *LNCS*, pages 48–62. Springer, 2006.

Dimitry Frumkin and Adi Shamir. Untrusted-HB: Security vulnerabilities of Trusted-HB. Cryptology ePrint Archive, Report 2009/044, 2009. `http://eprint.iacr.org`.

Heni Gilbert, Matthew J.B. Robshaw, and Yannick Seurin. HB$^{\#}$: Increasing the security and efficiency of HB^{+}. In *Proc. of EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 361–378. Springer, 2008a.

Henri Gilbert, Matthew J.B. Robshaw, and Hervé Sibert. Active attack against HB^{+}: A provable secure lightweight authentication protocol. *Electronic Letters*, 41:1169–1170, 2005.

Henri Gilbert, Matthew Robshaw, and Yannick Seurin. Good variants of HB^{+} are hard to find. In *Proc. of Financial Cryptography and Data Security*, volume 5143 of *LNCS*, pages 156–170. Springer, 2008b.

Zbigniew Gołębięwski, Krzysztof Majcher, and Filip Zagórski. Attacks on CKK family of RFID authentication protocols. In *Proc. Adhoc-now 2008*, volume 5198 of *LNCS*, pages 241–250. Springer, 2008.

Jovan Dj. Golić. Correlation via linear sequential circuit approximation of combiners with memory. In *Proc. of EUROCRYPT 1993*, volume 658 of *LNCS*, pages 113–123. Springer, 1993.

Jovan Dj. Golić. Cryptanalysis of alleged A5 stream cipher. In *Proc. of EUROCRYPT 1997*, volume 1233 of *LNCS*, pages 239–255. Springer, 1997.

Jovan Dj. Golić. Correlation properties of general binary combiners with memory. *Journal of Cryptology*, 9(2):111–126, 1996.

Jovan Dj. Golić, Vittorio Bagini, and Guglielmo Morgari. Linear cryptanalysis of Bluetooth stream cipher. In *Proc. of EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 238–255. Springer, 2002.

Solomon W. Golomb. *Shift Register Sequences*. Aegean Park Press, 1981.

Marc Goresky and Andrew Klapper. Fibonacci and Galois representations of feedback-with-carry shift registers. *IEEE Trans. Inform. Theory*, 48(11): 2826–2836, 2002.

Marc Goresky and Andrew Klapper. Periodicity and distribution properties of combined FCSR sequences. In *Proc. of SETA 2006*, volume 4086 of *LNCS*, pages 334–341. Springer, 2006.

Mark Goresky and Andrew Klapper. Arithmetic crosscorrelations of feedback with carry shift registers. *IEEE Trans. Inform. Theory*, 43:1342–1345, 1997.

Jian Guo, Thomas Peyrin, and Axel Poschmann. The PHOTON family of lightweight hash functions. In *Proc. of CRYPTO 2011*, volume 6841 of *LNCS*, pages 222–239. Springer, 2011.

Matthias Hamann. On the complexity of a learning problem induced by a lightweight cryptographic construction. Master's thesis, University of Mannheim, 2010.

Jonathan Hammell, André Weimerskirch, Joao Girao, and Dirk Westhoff. Recognition in a low-power environment. *Proc. of International Conference on Distributed Computing Systems*, 9:933–938, 2005.

Philip Hawkes and Gregory G. Rose. Rewriting variables: the complexity of fast algebraic attacks on stream ciphers. In *Proc. of CRYPTO 2004*, volume 3152 of *LNCS*, pages 390–406. Springer, 2004.

Martin Hell. *On the Design and Analysis of Stream Ciphers*. PhD thesis, Lund University, Sweden, 2007.

Martin Hell and Thomas Johansson. Breaking the F-FCSR-H stream cipher in real time. In *Proc. of ASIACRYPT 2008*, volume 5350 of *LNCS*, pages 557–569. Springer, 2008.

Martin Hell and Thomas Johansson. Breaking the stream ciphers F-FCSR-H and F-FCSR-16 in real time. *Journal of Cryptology*, pages 1–19, 2009.

Martin Hell and Thomas Johansson. Two new attacks on the self-shrinking generator. *IEEE Trans. Inform. Theory*, 52(8):3837–3843, 2006.

Martin Hell, Thomas Johansson, Alexander Maximov, and Willi Meier. A stream cipher proposal: Grain-128. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/010, 2005. `http://www.ecrypt.eu.org/stream`.

Miia Hermelin and Kaisa Nyberg. Correlation properties of the Bluetooth combiner. In *Proc. of ICISC 1999*, volume 1787 of *LNCS*, pages 17–29. Springer, 1999.

Christopher Hooley. On artin's conjecture. *J. Reine Angew. Math.*, 22:209–220, 1967.

Nicholas Hopper and Manuel Blum. Secure human identification protocols. In *Proc. of ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 52–66. Springer, 2001.

ISO/IEC. *ISO/IEC 9797-1: Information technology – Security techniques – Message Authentication Codes (MACs) – Part 1: Mechanisms using a block cipher*, 1999.

ISO/IEC. *ISO/IEC 9798-2: Information Technology - Security techniques — Entitiy Authentication Mechanisms Part 2: Entity Authentication with symmetric techniques*, 1993.

ISO/IEC. *ISO/IEC 13616-2: Financial services - International bank account number (IBAN) – Part 2: Role and responsibilities of the Registration Authority*, 2007.

Markus Jakobsson and Susanne Wetzel. Security weakness in Bluetooth. In *Proc. of CT-RSA 2001*, volume 2020 of *LNCS*, pages 176–191. Springer, 2001.

Éliane Jaulmes and Frédéric Muller. Cryptanalysis of the F-FCSR stream cipher family. In *Proc. of SAC 2006*, volume 3897 of *LNCS*, pages 20–35. Springer, 2006.

Éliane Jaulmes and Frédéric Muller. Cryptanalysis of ECRYPT candidates F-FCSR-8 and F-FCSR-H. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/046, 2005. `http://www.ecrypt.eu.org/stream`.

Ari Juels and Stephen A. Weis. Authenticating pervasive devices with human protocols. In *Proc. of CRYPTO 2005*, volume 3621 of *LNCS*, pages 293–308. Springer, 2005.

David Kahn. *The Codebreakers*. Scribner, 1996.

Auguste Kerckhoffs. La cryptographie militaire. *Journal des Sciences Militaires*, pages 161–191, 1883.

Andrew Klapper. A survey of feedback with carry shift registers. In *Proc. of SETA 2004*, volume 3486 of *LNCS*, pages 56–71. Springer, 2004.

Andrew Klapper and Marc Goresky. Feedback shift registers, 2-adic span, and combiners with memory. *Journal of Cryptology*, 10:111–147, 1997.

Andrew Klapper and Mark Goresky. 2-adic shift registers. In *Proc. of FSE 1994*, volume 809 of *LNCS*, pages 174–178. Springer, 1994.

Andrew Klapper and Jinzhong Xu. Register synthesis for algebraic feedback shift registers based on non-primes. *Des. Codes Cryptography*, 31(3):227–250, 2004.

Matthias Krause. BDD-based cryptanalysis of keystream generators. In *Proc. of EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 222–237. Springer, 2002.

Matthias Krause. OBDD-based cryptanalysis of oblivious keystream generators. *Theor. Comp. Sys.*, 40(1):101–121, 2007.

Matthias Krause, Christoph Meinel, and Stephan Waack. Separating the eraser turing machine classes L_e, NL_e, $co-NL_e$ and P_e. In *Mathematical Foundations of Computer Science 1988*, volume 324 of *LNCS*, pages 405–413. Springer, 1988.

Hugo Krawczyk. LFSR-based hashing and authentication. In *Proc. of CRYPTO 1994*, volume 839 of *LNCS*, pages 129–139. Springer, 1994.

Hugo Krawczyk, Mihir Bellare, and Ran Canetti. HMAC: Keyed-hashing for message authentication. RFC 2104, 1997. `http://tools.ietf.org/html/rfc2104`.

Leslie Lamport. Password authentication with insecure communication. *Commun. ACM*, 24:770–772, November 1981.

Yuseop Lee, Kitae Jeong, Jaechul Sung, and Seokhie Hong. Related-key chosen IV attacks on Grain-v1 and Grain-128. In *Information Security and Privacy*, volume 5107 of *LNCS*, pages 321–335. Springer, 2008.

Éric Levieil and Pierre-Alain Fouque. An improved LPN algorithm. In *Security and Cryptography for Networks*, volume 4116 of *LNCS*, pages 348–359. Springer, 2006.

Yi Lu and Serge Vaudenay. Faster correlation attack on the Bluetooth keystream generator. In *Proc. of CRYPTO 2004*, volume 3152 of *LNCS*, pages 407–425. Springer, 2005.

Yi Lu and Serge Vaudenay. Cryptanalysis of an E0-like combiner with memory. *Journal of Cryptology*, 21:430–457, 2008.

Yi Lu and Serge Vaudenay. Cryptanalysis of the Bluetooth keystream generator two-level E0. In *Proc. of ASIACRYPT 2004*, volume 3329 of *LNCS*, pages 483–499. Springer, 2004.

Yi Lu, Willi Meier, and Serge Vaudenay. The conditional correlation attack: A practical attack on Bluetooth encryption. In *Proc. of CRYPTO 2005*, volume 3621 of *LNCS*, pages 97–117. Springer, 2005.

Stefan Lucks, Erik Zenner, André Weimerskirch, and Dirk Westhoff. Concrete security for entity recognition: The Jane Doe protocol. In *Proc. of INDOCRYPT 2008*, volume 5365 of *LNCS*, pages 158–171. Springer, 2008.

Yishay Mansour, Noam Nisan, and Prasoon Tiwari. The computational complexity of universal hashing. In *Proc. of STOC '90*, pages 235–243, New York, NY, USA, 1990. ACM.

George Marsaglia and Arif Zaman. A new class of random number generators. *Annals of Appl. Prob.*, 1(3):462–480, 1992.

Alexander Maximov and Alex Biryukov. Two trivial attacks on TRIVIUM. eSTREAM, ECRYPT Stream Cipher Project, Report 2007/006, 2007. `http://www.ecrypt.eu.org/stream`.

Alexander Maximov, Thomas Johansson, and Steve Babbage. An improved correlation attack on A5/1. In *Proc. SAC 2005*, volume 3357 of *LNCS*, pages 1–18. Springer, 2005.

Willi Meier and Othmar Staffelbach. Nonlinearity criteria for cryptographic functions. In *Proc. of EUROCRYPT 1989*, volume 434 of *LNCS*, pages 549–562. Springer, 1989.

Willi Meier and Othmar Staffelbach. Fast correlation attacks on stream ciphers. In *Proc. of EUROCRYPT 1988*, volume 330 of *LNCS*, pages 301–314. Springer, 1988.

Willi Meier and Othmar Staffelbach. The self-shrinking generator. In *Proc. of EUROCRYPT 1994*, volume 950 of *LNCS*, pages 205–214. Springer, 1994.

Willi Meier, Enes Pasalic, and Claude Carlet. Algebraic attacks and decomposition of boolean functions. In *Proc. of EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 474–491. Springer, 2004.

Christoph Meinel. *Modified Branching Programs and Their Computational Power*, volume 370 of *LNCS*. Springer, 1989.

Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Applied Cryptography*. CRC Press, 2001.

Ralph C. Merkle. *Secrecy, Authentication, and Public Key Systems*. PhD thesis, Stanford University, 1979.

Ralph C. Merkle. A certified digital signature. In *Proc. of CRYPTO '89*, volume 435 of *LNCS*, pages 218–238. Springer, 1990.

Miodrag J. Mihaljević. A faster cryptanalysis of the self-shrinking generator. In *Proc. of ACISP 1996*, volume 1172 of *LNCS*, pages 192–189. Springer, 1996.

Amir Moradi, Axel Poschmann, San Ling, Christof Paar, and Huaxiong Wang. Pushing the limits: A very compact and a threshold implementation of AES. In *Proc. of EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 69–88. Springer, 2011.

Jorge Munilla and Alberto Peinado. HB-MP: A further step in the HB-family of lightweight authentication protocols. *Computer Networks*, (51):2262–2267, 2007.

National Institute of Standards and Technology NIST. *Secure Hash Standard (SHS)*, October 2008. `http://csrc.nist.gov/publications/fips/fips180-3/`.

National Institute of Standards and Technology NIST. *Cryptographic Hash Algorithm Competition*, December 2010. `http://www.nist.gov/hash-competition/`.

Karsten Nohl and Sascha Krißler. Geheimnislos - Verschlüsselung von Handy-Gesprächen knacken. *i'X*, 5:97–99, 2010. (in german).

Jim Noras. Fast pseudorandom sequence generators: Linear feedback shift registers, cellular automata, and carry feedback shift registers. Technical Report 94, Univ. Bradford Elec. Eng. Dept., Bradford, U.K., 1997.

Khaled Ouafi, Raphael Overbeck, and Serge Vaudenay. On the security of HB$^{\#}$ against a man-in-the-middle attack. In *Proc. of ASIACRYPT 2008*, volume 5350 of *LNCS*, pages 108–124. Springer, 2008.

Srinivasa R. Pappu. *Physical One-Way Functions.* PhD thesis, Massachusetts Institute of Technology, 2001.

Bart Preneel. CBC-MAC and variants. In Henk Tilborg, editor, *Encyclopedia of Cryptography and Security*, pages 63–66. Springer US, 2005.

Bart Preneel. The state of hash functions and the NIST SHA-3 competition. In *Information Security and Cryptology*, volume 5487 of *LNCS*, pages 1–11. Springer, 2009.

Bart Preneel. *Analysis and Design of Cryptographic Hash Functions.* PhD thesis, K.U. Leuven, 1993.

Bart Preneel, René Govaerts, and Joos Vandewalle. Hash functions based on block ciphers: a synthetic approach. In *Proc. of CRYPTO '93*, volume 773 of *LNCS*, pages 368–378. Springer, 1994.

Ronald Rivest. The MD5 message-digest algorithm. RFC 1321, 1992. `http://tools.ietf.org/html/rfc1321`.

Rainer A. Rueppel. *Design and Analysis of Stream Ciphers.* Springer, 1986.

Rainer A. Rueppel. Stream ciphers. In *Contemporary Cryptology – The Science of Information Integrity*, pages 65–134. IEEE Press, 1992.

Andrew Rukhin, Juan Soto, James Nechvatal, Miles Smid, Elaine Barker, Stefan Leigh, Mark Levenson, Mark Vangel, David Banks, Alan Heckert, James Dray, and San Vo. *A Statistical Test Suite for the Validation of Random Number Generators and Pseudo Random Number Generators for Cryptographic Applications*. National Institute of Standards and Technology (NIST), April 2010. `http://csrc.nist.gov/rng/`.

Markku Saarinen. Re: Bluetooth and E_0. Posted at sci.crypt.research, 02/09/00, 2000.

Mahmoud Salmasizadeh, Jovan Dj. Golić, Ed Dawson, and Leone Simpson. A systematic procedure for applying fast correlation attacks to combiners with memory. In *Proc. of SAC 1997*, 1997.

Felix Schleer. Einsatz von OBDDs zur Kryptanalyse von Flusschiffren. Master's thesis, University of Mannheim, Mannheim, Germany, 2002. (in german).

Yaniv Shaked and Avishai Wool. Cryptanalysis of the Bluetooth E_0 cipher using OBDDs. Technical report, Cryptology ePrint Archive, Report 2006/072, 2006. `http://eprint.iacr.org/`.

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

Thomas Siegenthaler. Correlation-immunity of nonlinear combining functions for cryptographic applications. *IEEE Trans. Inform. Theory*, IT-30(5):776–780, 1984.

Thomas Siegenthaler. Decrypting a class of stream ciphers using ciphertext only. *IEEE Trans. Inform. Theory*, C-34(1):81–85, 1985.

Fabio Somenzi. *CUDD: CU decision diagram package*. University of Colorado, Boulder, CO, USA, March 2001. http://vlsi.colorado.edu/~fabio/.

Paul Stankovski, Martin Hell, and Thomas Johansson. An efficient state recovery attack on X-FCSR-256. In *Proc. of FSE 2009*, volume 5665 of *LNCS*, pages 23–37. Springer, 2009.

Dirk Stegemann. BDD-basierte Kryptanalyse des A5/1 Schlüsselstromgenerators. Master's thesis, University of Mannheim, 2004. (in german).

The Bluetooth SIG. *Specification of the Bluetooth System*, February 2001.

Tian Tian and Wen-Feng Qi. Linearity properties of binary FCSR sequences. *Designs, Codes and Cryptography*, 52:249–262, 2009.

Serge Vaudenay. *A Classical Introduction to Cryptography*. Springer US, 2006.

Ingo Wegener. *Branching Programs and Binary Decision Diagrams: Theory and Applications*. SIAM Monographs on Discrete Mathematics and Applications, 2000.

Clint R. Whaley and Antoine Petitet. Minimizing development and maintenance costs in supporting persistently optimized BLAS. *Software: Practice and Experience*, 35(2):101–121, 2005.

Guo-Zhen Xiao and James L. Massey. A spectral characterization of correlation-immune combining functions. *IEEE Trans. Inform. Theory*, IT-34(3):569–571, 1988.

Hong Xu and Wen-Feng Qi. Autocorrelations of maximum-length FCSR sequences. *SIAM J. Discrete Math.*, 20(3):568–577, 2006.

Hong Xu, Wen-Feng Qi, and Yong-Hui Zheng. Autocorrelations of l-sequences with prime connection integer. *Journal of Cryptography and Communications*, 1(2):207–223, September 2009.

Erik Zenner. *On Cryptogaphic Properties of LFSR-based Pseudorandom Generators*. PhD thesis, University of Mannheim, Mannheim, Germany, 2004.

Erik Zenner, Matthias Krause, and Stefan Lucks. Improved cryptanalysis of the self-shrinking generator. In *Proc. of ACISP 2001*, volume 2119 of *LNCS*, pages 21–35. Springer, 2001.

Bin Zhang and Dengguo Feng. New guess-and-determine attack on the self-shrinking generator. In *Proc. of ASIACRYPT 2006*, volume 4284 of *LNCS*, pages 54–68. Springer, 2006.

Robert Zuccherato. Entity authentication. In Henk Tilborg, editor, *Encyclopedia of Cryptography and Security*, pages 203–203. Springer US, 2005.

List of Tables

5.1 Information rates α for the restricted A5/1 76
5.2 Simulation parameters of the BDD-based attack 77
5.3 Performance of the BDD-based attack in practice 78

6.1 The resource consumption of the fastest correlation attack on E_0 as presented by Lu and Vaudenay (2004) 90
6.2 The resource consumption of an algebraic attack on E_0 with key size n and an equation of degree d 91
6.3 Maximum absolute biases and performance of correlation attacks for β_a-generators 92
6.4 mindeg and number of Z-functions for the candidate generators . 92
6.5 Definitions of the candidate generators 92
6.6 Performance of algebraic and correlation attacks on the candidate generators 93

9.1 Performance of the passive attack on CKK^2 122

List of Figures

2.1 The Shannon communication model 8
2.2 Stream cipher communication scenario 11
2.3 Common construction of the keystream generator 11

3.1 Feedback shift register (FSR) of length n 16
3.2 LFSR in Fibonacci architecture 17
3.3 LFSR in Galois architecture 17
3.4 Mapping between periodic Galois and Fibonacci LFSR states . . 20
3.5 FCSR in Fibonacci architecture 22
3.6 FCSR in Galois architecture 23
3.7 Mapping between periodic Galois and Fibonacci FCSR states . . 27

4.1 FSR-based combination generator 34
4.2 FSR-based filter generator 34
4.3 Equivalent representations of combination and filter generators . 35
4.4 The E_0 keystream generator 37
4.5 The A5/1 keystream generator 39
4.6 Derivation of the keystream from the internal bitstream 44

5.1 An oracle graph G^0 over $\{z_0, \ldots, z_3\}$ and a G^0-FBDD 49
5.2 A π-OBDD over $\{z_0, \ldots, z_3\}$ with $\pi(0) = 0$, $\pi(1) = 2$, $\pi(2) = 1$ and $\pi(3) = 3$. 51

6.1 Comparison of the candidate generators to E_0 93

7.1 Message authentication with message authentication codes 101

8.1 Basic round of the HB protocol 107
8.2 Basic round of the HB^{+} protocol 109
8.3 Initialization of the HB^{++} protocol 110
8.4 Basic round of the HB^{++} protocol 111
8.5 Basic round of the HB^{*} protocol 112
8.6 Basic round of the HB-MP protocol 113
8.7 One round of the $\mathsf{HB}^{\#}$ protocol 114

9.1 Basic round of the (n, k, L) protocol 118
9.2 Basic round of the $(n, k, L)^{+}$ protocol 119
9.3 Basic round of the $(n, k, L)^{++}$ protocol 120

List of Algorithms

1 F-FCSR-H-KeyIVSetup(K, IV) 41
2 F-FCSR-H-KeystreamGeneration 41
3 F-FCSR-16-KeyIVSetup(K, IV) 42
4 F-FCSR-16-KeystreamGeneration 42
5 RecoverInitialState 53
6 FibonacciFCSR-$S_m(\pi_m, w)$ 58
7 SelfShrinkingGenerator-$Q_m(w, z)$ 60
8 E_0-$Q_m(q_0, w, z)$ 61
9 read-once-A5/1(w) 62
10 A5/1-$Q_m(w, z)$ 65
11 RecoverInitialState-DCS 70
12 $(n, k, L)^+$_MITM-Attack(n, k, L) 119
13 CKK^2_Attack(n, k) 122
14 LULS-solve(O) 125

Index

Zahlen
2-adic numbers 23
2-adic span 30

A
A5/1 38
Advanced Encryption Standard 10
AES . . . . . *see* Advanced Encryption Standard
autocorrelation 12

B
BDD . . . . *see* Binary Decision Diagram
Binary Decicsion Diagram 48
block cipher 9

C
CBC . . . . . *see* Cipher Block Chaining
Cipher Block Chaining 10
CKK 121
connection polynomial 18

D
Data Encryption Standard 10
DES . . . *see* Data Encryption Standard
digital signature 104

E
E_0 36
ECB . . . . . *see* Electronic Codebook
Electronic Codebook 10

F
F-FCSR 40
FBDD . . . . . *see* Free Binary Decision Diagram
Feedback Shift Register 16
Feedback with carry shift register . . 21
Fibonacci architecture 16, 22
Free Binary Decision Diagram 49

G
Galois architecture 17, 22
Grain 40

H
Hamming weight 15
hash chain 105
hash function 102
HB 107
HB* 111
HB$^+$ 108
HB^{++} 110
HB$^\#$ 113
HBMP 112
HMAC 103

L
linear complexity 21
Linear Feedback Shift Register (LFSR) 16

M
MAC *see* message authentication code
MD5 103
message authentication code 101

O
OBDD . . *see* Ordered Binary Decision Diagram
one-time pad 9
oracle graph 48
Ordered Binary Decision Diagram . 50

S
self-shrinking generator 36
SHA 103
SSG . . . . . *see* self-shrinking generator

T
Toeplitz matrix 114
Trivium 39
Trusted-HB 114

W
wt(x) . . . . . *see* Hamming weight