Design and Analysis of Cryptographic Pseudorandom Number/Sequence Generators with Applications in RFID

by

Kalikinkar Mandal

A thesis
presented to the University of Waterloo
in fulfilment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2013

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

This thesis is concerned with the design and analysis of strong de Bruijn sequences and span n sequences, and nonlinear feedback shift register (NLFSR) based pseudorandom number generators for radio frequency identification (RFID) tags. We study the generation of span n sequences using structured searching in which an NLFSR with a class of feedback functions is employed to find span n sequences. Some properties of the recurrence relation for the structured search are discovered. We use five classes of functions in this structured search, and present the number of span n sequences for $6 \leq n \leq 20$. The linear span of a new span n sequence lies between near-optimal and optimal. According to our empirical studies, a span n sequence can be found in the structured search with a better probability of success. Newly found span n sequences can be used in the composited construction and in designing lightweight pseudorandom number generators.

We first refine the composited construction based on a span n sequence for generating long de Bruijn sequences. A de Bruijn sequence produced by the composited construction is referred to as a composited de Bruijn sequence. The linear complexity of a composited de Bruijn sequence is determined. We analyze the feedback function of the composited construction from an approximation point of view for producing strong de Bruijn sequences. The cycle structure of an approximated feedback function and the linear complexity of a sequence produced by an approximated feedback function are determined. A few examples of strong de Bruijn sequences with the implementation issues of the feedback functions of an $(n + 16)$-stage NLFSR are presented.

We propose a new lightweight pseudorandom number generator family, named Warbler family based on NLFSRs for smart devices. Warbler family is comprised of a combination of modified de Bruijn blocks (CMDB) and a nonlinear feedback Welch-Gong (WG) generator. We derive the randomness properties such as period and linear complexity of an output sequence produced by the Warbler family. Two instances, Warbler-I and Warbler-II, of the Warbler family are proposed for passive RFID tags. The CMDBs of both Warbler-I and Warbler-II contain span n sequences that are produced by the structured search. We analyze the security properties of

Warbler-I and Warbler-II by considering the statistical tests and several cryptanalytic attacks. Hardware implementations of both instances in VHDL show that Warbler-I and Warbler-II require 46 slices and 58 slices, respectively. Warbler-I can be used to generate 16-bit random numbers in the tag identification protocol of the EPC Class 1 Generation 2 standard, and Warbler-II can be employed as a random number generator in the tag identification as well as an authentication protocol for RFID systems.

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my supervisor, Prof. Guang Gong, for accepting and supporting me as a Ph.D. student and allowing to work in the field of my research interests. Prof. Gong's invaluable supervision, guidance, constructive criticism, and encouragement have been crucial in helping me to develop as a researcher. Her counsel and expertise in the field resolved many difficulties that I encountered during my doctoral studies. I thank Prof. Gong for her advice and valued suggestions to all aspects of my research and beyond research.

I sincerely thank, Prof. Radha Poovendran at the University of Washington, Seattle, for serving as my external examiner and providing me many valuable suggestions and comments. I am deeply grateful to my thesis committee, Prof. Mark Aagaard, Prof. Anwar Hasan, and Prof. David Jao at the University of Waterloo, for giving me insightful comments and suggestions, and helping me to improve my thesis. It is a privilege to obtain such a great committee.

I am obliged to Dr. Xinxin Fan for his assistance from the first day at the University of Waterloo, his friendship, being a co-author, exchanging valuable ideas and giving me many helpful suggestions. I am thankful to my former colleagues Dr. Honggang Hu, Dr. Zhijun Li, Dr. Anuchart Tassanaviboon, Dr. Qi Chai for their support, friendship and sharing valued ideas. I am also thankful to my colleagues and friends Fei Huo, Bo Zhu, Yang Yang, Teng Wu, Shasha Zhu, Khizer Kaleem, Roy Feng, Yin Tan, Yao Chen, Gangqiang Yang, and Nusha Zidaric, for their support, their friendship, and having fun with them during my PhD studies. I would like to thank all the members of the Communication Security Lab (ComSec) at the University of Waterloo for making and maintaining a wonderful research environment. I would like to thank all my friends at the University of Waterloo and Beacon Tower-702 for their cooperation and help, and having fun with them, and I would like to single out Arindam, Harshwardhan, Pradeep, Rudra, Manu, and Saurabh.

I thank my M. Tech. friends, Pulak, Sanjay, Swarup, Sandeep, Mrinmoy, Nargis, Aritra, Subhabrata, Somindu, Santanu, and Chiranjit for their supportive, lovable,

humorous, and entertaining friendship, and keeping in touch.

Last but not least, I would like to thank my parents, my brothers Uday and Ashok, and my family for their endless love, support and sacrifices. I am indebted to my parents and my brother Uday for their encouragements and sacrifices. I dedicate this thesis to my parents. I would also like to thank my beloved Arpita Sinha for her endless love and support. I thank my lovable grandmothers for their unconditional love.

Thank you very much to all of you!

Dedication

To my parents

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Randomness plays an essential role in cryptography. A good random number or sequence generator is a crucial component used in designing a secure system. Security of cryptographic algorithms and protocols extensively relies on random numbers and keys. Random number generators can be classified into two categories, namely true random number generators and pseudorandom number generators. The role of a pseudorandom sequence or number generator becomes crucial in a practical scenario when a true random bit generator cannot produce enough truly random bits or a true random bit generator cannot be employed to produce random bits. Pseudorandom sequence/number generators are broadly used in stream ciphers, generating random numbers, cryptographic protocols, digital signature generation algorithms, RFID systems, and sensor networks. This chapter provides an introduction to cryptographic pseudorandom sequences, and the motivation and outline of this thesis.

1.1 Pseudorandom Sequence Generators

A pseudorandom sequence generator (PRSG) is a deterministic algorithm, which takes a small length truly random sequence, called the *seed*, as input, and generates an output sequence of any desired length. The output sequence is called a *pseudorandom sequence*, which is *random-looking* to an outside observer, and will be

indistinguishable from a truly random sequence. Obviously, the output sequence length is much greater than the input sequence length. In other words, a PRSG is used to expand a truly random seed to a long length pseudorandom sequence, which should have the following statistical properties: long period, balance, equal distribution of runs, uniform tuple distribution, ideal 2-level autocorrelation, low crosscorrelation, and high linear span [19, 43, 45, 102].

Numerous algorithms and techniques can be found in the literature for generating pseudorandom sequences including feedback shift registers (FSR), feedback with carry shift registers, linear congruential generators, lagged fibonacci generators, many designs based on block ciphers and hash functions. The feedback shift register has a rich theory, and that can be classified into two categories, namely linear feedback shift registers (LFSRs) and nonlinear feedback shift registers (NLFSRs) [43]. Feedback shift register sequences such as de Bruijn sequences and span n sequences have good randomness properties, and FSRs have an efficient implementation in hardware [43]. The designs of the eSTREAM profile 2 finalists Grain and Trivium are based on feedback shift registers [30]. Several architectures of FSR-based PRNGs and stream ciphers can be found in [19]. A wide range of applications of FSRs can be found in cryptography, spread spectrum communication, test vector generation in hardware design, etc. [43, 45].

1.2 Cryptographic Pseudorandom Sequences and Their Applications

In cryptography, strong pseudorandom sequences are significant for providing security in various applications. A PRSG is called a cryptographically secure PRSG if no *polynomial-time algorithm* with the first l bits of a sequence as input can predict the $(l+1)$-th bit of the sequence with probability significantly greater than $\frac{1}{2}$ [87]. Known cryptographically secure PRNGs are Shamir's generators and Blum-Blum-Shub generators [87]. For an FSR-generated sequence, the value of l should be approximately equal to the period of the sequence. A cryptographically secure sequence should have an indistinguishability property, an unpredictability prop-

erty, and good statistical properties such as long period, high linear span, balance, equal distribution of runs, uniform tuple distribution, ideal 2-level autocorrelation, and low crosscorrelation. Moreover, the pseudorandom sequence generator will be resistant to the cryptanalytic attacks. Many cryptographic statistical tests have been proposed to measure the randomness in a pseudorandom sequence, for example the NIST statistical test suite [103]. In practice, the randomness in a sequence is measured by applying the statistical tests to an arbitrary but small segment of an output sequence. Due to computational limitations, it is not possible to take the entire sequence as input to the statistical testing algorithms. A cryptographic sequence should pass all the statistical tests. A cryptographic pseudorandom sequence generator needs to be characterized by the statistical properties of its output sequences, and by its attack resistance properties.

It is desirable that the period or the lower bound of period of a pseudorandom sequence be known for cryptographic applications. For instance, in a stream cipher, the length of a sequence needs to be the same as the length of a message, and the sequence will never be repeated. Furthermore, the linear span or linear complexity of a sequence must be high, so that an adversary is not able to generate the entire sequence from a partially known segment by the Berlekamp-Massey algorithm [79]. For a cryptographic pseudorandom sequence, the period and linear span or complexity of a sequence must be large because the linear complexity is the measure of unpredictability of a sequence.

There are several applications of (pseudo)random sequences and numbers in both symmetric-key and public-key cryptography. In any cryptosystem, the keys of an encryption and decryption scheme are chosen in a random fashion from a keyspace; pseudorandom sequences can be used for those purposes. For example, prime numbers and the private key in an RSA cryptosystem can be chosen using a PRNG. A stream cipher uses a pseudorandom sequence generator for generating keystreams to encrypt and decrypt messages. A variety of applications of NLFSR-based PRNGs can be found in resource constrained-environments such as RFID tags and sensor networks. Note that most of the authentication protocols use nonce as a challenge and the security of the protocol depends on nonces. In those application

scenarios, the pseudorandom sequences could be a better choice to use as nonces. Last but not least, a segment of a pseudorandom sequence can be used as a random number in a digital signature generation algorithm to protect the private key.

1.3 Motivation and This Thesis

A pseudorandom sequence generator constructed by feedback shift registers can be observed as a system of multivariate algebraic equations. Over the past few decades, a number of cryptanalytic attacks on FSR-based PRNGs/PRSGs have been developed, for instance algebraic attacks, correlation attacks, cube attacks, time-memory-data (TMD) tradeoff attacks, and discrete fourier transformation (DFT) attacks [6,20,21,24,46,84,100,105]. These attacks are very powerful against LFSR-based PRNGs or stream ciphers. However, it is believed that an NLFSR-based PRNG or stream cipher can resist the existing attacks due to the hardness of solving a system of multivariate nonlinear equations over the binary field. Replacing the LFSR building blocks by well-chosen NLFSRs in the existing architectures of PRSGs, such as filtering generator and combinatorial generator, either the aforementioned attacks can be prevented or the attacks' complexities are increased so that launching the attacks become infeasible. Feedback shift registers can also be implemented efficiently in hardware. Motivated by the attack-resistance properties and efficient hardware implementations of NLFSR-based PRSGs/PRNGs, we study nonlinear feedback shift registers, and NLFSR-based pseudorandom sequence and number generators in this thesis.

This thesis concentrates on the generation of de Bruijn sequences and span n sequences using NLFSRs due to their good randomness properties. A binary de Bruijn sequence generated by an NLFSR has period 2^n in which all binary n-tuples occur exactly once, and linear complexity at least $2^{n-1}+n$ [14]. On the other hand, a binary span n sequence or modified de Bruijn sequence generated by an NLFSR has period 2^n-1 where every nonzero n-tuple occurs exactly once in a period. A span n sequence may also have high linear complexity. An NLFSR that generates a de Bruijn sequence or a span n sequence needs to be used as a building block in

designing a PRNG or a stream cipher, because the randomness properties of output sequences can be promised for a suitable design. Unfortunately, there is no known general construction of an NLFSR that can generate a span n sequence.

In Chapter 4, we use the structured search methods employing a class of nonlinear feedback functions to study the generation of span n sequences using NLFSRs. Chapter 5 refines and examines the composited construction and its sequence properties for producing long and strong de Bruijn sequences. Chapter 6 presents Warbler family, a new pseudorandom number generator family based on NLFSRs with desirable randomness properties for smart devices such as radio frequency identification tags. Using the span n sequences newly found by the structured search, we design two instances, Warbler-I and Warbler-II, of the Warbler family in Chapters 7 and 8, respectively, for radio frequency identification systems.

1.4 Radio Frequency Identification Systems

Radio Frequency Identification (RFID) is a promising technology for automatic identification of remote objects. A typical RFID system consists of three main components, namely a *reader*, *tags*, and a *backend database*. A general overview of an RFID system is provided in Figure 1.1.

- *Readers:* A reader is a transceiver, which queries to the tags through radiowaves. Readers are as powerful as computers, and have enough capabilities to perform cryptographic operations. A reader is connected to a backend database by a secure wired/wireless channel.

- *Tags or transponders:* A tag is composed of a tiny integrated circuit for storing and processing identification information, and a radio antenna for wireless data transmission. There are three basic types of RFID tags, and the computation capability of a tag depends on the type of tag.

  - *Active tags:* An active tag contains internal batteries so that it can initialize communications with the reader and perform heavy computations.

– *Semi-passive tags:* Semi-passive tags use batteries only to power up their circuit and harvest power from the reader for communication.

– *Passive tags:* A passive tag does not contain any battery, it solely obtains power from the reader for both computation and communication. Passive tags usually have constrained capabilities in every aspect of computation, communication, and storage, due to the extremely low production cost. The reading range of a passive tag is up to several meters.

- *Backend database:* A backend database is connected to a reader, and it efficiently stores information about all the tags in the system, for example IDs, secrecy keys of tags. The connection between the backend server and the reader can be a wired or wireless connection.

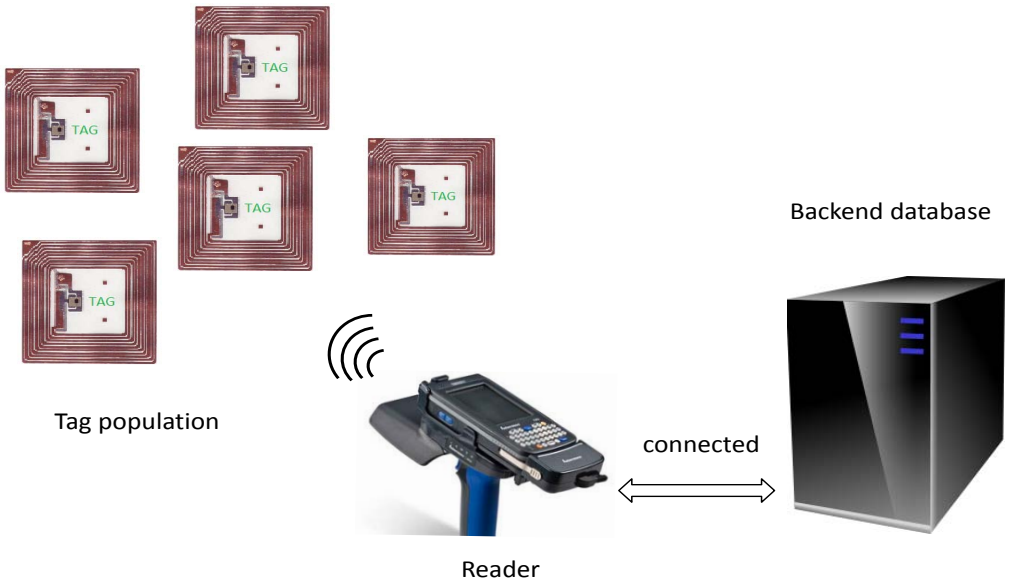

Figure 1.1: An RFID system

The EPC C1 Gen2 Tag Identification Protocol

The EPCglobal Class 1 Generation 2 (EPC C1 Gen2 in brief) standard has been approved as an ISO 18000-6C standard in 2006 [29]. Figure 1.2 shows an overview

of the tag identification protocol. In the EPC C1 Gen2 tag identification protocol, two main operations, namely *inventory* and *access*, are performed for managing the tag population. In the inventory operation (Steps 1-4 in Figure 1.2), after receiving a request from the reader, a tag generates a 16-bit random number, denoted by $RN16$, and temporarily stores the number in a slot counter. When the slot counter is zero, the tag backscatters $RN16$ to the reader. Thereafter, the reader copies $RN16$ to an acknowledgement packet to be sent to the tag. When the tag receives the acknowledgement packet, it first compares the random number in the acknowledgement packet with $RN16$. If these two numbers are the same, then the tag backscatters the acknowledgement packet.

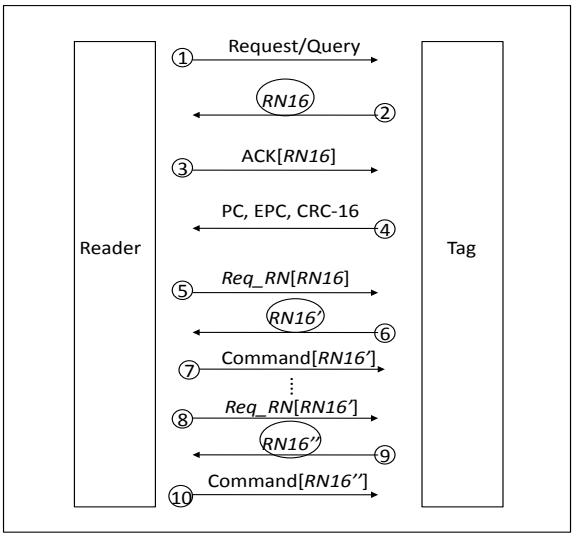

Figure 1.2: The EPC C1 Gen2 tag identification protocol

In the access operation (Steps 5-7 in Figure 1.2), after receiving a request, denoted by $ReqRN$, from the reader, the tag compares the random number in the request $ReqRN$ with the stored $RN16$. If two random numbers match, the tag generates another random number $RN16'$, which is called *handle* and backscatters it to the reader. The reader then issues the commands such as Read, Write, and BlockWrite. Steps 8-10 in Figure 1.2 demonstrate a further access operation. Note that for each access operation the tag generates a new random number.

1.5 Overview and Main Contributions

We provide an overview and the main contributions of this thesis. The thesis is divided into three parts. The first part includes Chapters 1,2 and 3 that provide an introduction, a literature survey on the de Bruijn and span n sequences and random number generators, and preliminaries to pseudorandom sequence and number generators. The second part includes Chapters 4 and 5 that concentrate on the generation of strong span n sequences and de Bruijn sequences, and the third part contains Chapters 6, 7 and 8 that present the application of span n sequences in designing lightweight pseudorandom number generators for the RFID tags.

- **Chapter 1** starts by providing an introduction to cryptographic pseudorandom sequence and number generators. This thesis concentrates on nonlinear feedback shift register based pseudorandom sequence and number generators. We give a brief overview of radio frequency identification systems. We also provide the motivation and an outline of this thesis in this chapter.

- **Chapter 2** presents a survey on the existing techniques for generating de Bruijn sequences and span n sequences. The random number generator proposals for the EPC C1 Gen2 RFID tags are presented as well.

- **Chapter 3** describes and defines the mathematical concepts related to finite fields, sequences and Boolean functions, which are fundamental mathematical concepts behind designing and analyzing a pseudorandom sequence generator. We present the basic concepts related to feedback shift registers and their sequence properties such as period, linear complexity. We recall the cryptographic properties such as nonlinearity, algebraic immunity of Boolean functions, which are also important in designing secure pseudorandom sequence/number generators.

- **Chapter 4** focuses on the generation of span n sequences using nonlinear feedback shift registers. The span n sequences are generated by the structured search where in a nonlinear recurrence relation is composed of a permutation and a trace function over a finite field, a decimation number and a t-tap

position when $5 \leq t < n$. We use several classes of feedback functions such as WG transformations, 5-term functions, 3-term functions, monomial functions with a Kasami exponent, and MCM functions in the structured search. We present the number of span n sequences produced by the aforementioned functions in an NLFSR for $6 \leq n \leq 20$. We study the linear complexity of new span n sequences, and our analysis shows that the linear complexity of a span n sequence lies between $(2^n - 2 - 3n)$ (near-optimal) and optimal $(2^n - 2)$. Moreover, we summarize the upper and lower bounds of span n sequences for each class of feedback functions. Our empirical comparison shows that a span n sequence with optimal or near-optimal linear complexity can be found by the structured search with a better probability of success than that of a random span n sequence generation method.

- **Chapter 5** investigates how to generate a strong de Bruijn sequence from a span n sequence by the composited construction. We first refine the composited construction by Lempel and Mykkeltveit *et al.* for generating long de Bruijn sequences by nonlinear feedback shift registers. In the composited construction, an $(n + k)$-stage NLFSR is constructed from an n-stage NLFSR by repeatedly applying the composition operation. A de Bruijn sequence produced by the composited construction is called a composited de Bruijn sequence. The linear complexity of a composited de Bruijn sequence is determined. We perform a cryptanalysis of the composited construction for generating strong de Bruijn sequences. In the analysis, we first consider an approximation of the feedback function of an $(n + k)$-stage composited NLFSR. Then, we determine the cycle structure of an approximated feedback function and the linear complexity of a sequence produced by an approximated feedback function. Our analysis shows that a de Bruijn sequence produced by the composited construction can be strong if the span n sequence produced by the n-stage NLFSR is strong. We present a few instances of cryptographically strong de Bruijn sequences with periods in the range of 2^{35} and 2^{40}, whose feedback functions are known. Finally, we consider the implementation issues of a feedback function of an $(n + 16)$-stage NLFSR.

- **Chapter 6** proposes Warbler family – a new lightweight pseudorandom number generator family with desirable randomness properties. Warbler family is composed of two building blocks, namely a combination of modified de Bruijn blocks and a nonlinear feedback WG generator. The combination of modified de Bruijn blocks is built by a number of primitive NLFSRs, and the nonlinear feedback WG generator is composed of an NLFSR and a WG transformation module over a finite field. We derive the randomness properties of sequences generated by the combination of modified de Bruijn blocks, followed by a description of the initialization and running phases of the Warbler family. Randomness properties such as period and linear complexity of output sequences produced by the Warbler family are inherited from the combination of modified de Bruijn blocks. Some parameter selection criteria for the Warbler family are proposed. Two instances of the Warbler family are presented in the next two chapters for RFID tags.

- **Chapter 7** presents a new instance, Warbler-I, of the Warbler family for the EPC C1 Gen2 RFID tags. Warbler-I is a nonlinear feedback shift register based pseudorandom number generator, and can generate 16-bit random numbers for the tag identification protocol of the EPC C1 Gen2 standard. Warbler-I, uses WG-5 transformations, is composed of two building blocks. The first one contains two primitive NLFSRs of lengths 17 and 18, and the second building block contains an NLFSR of length 6 over $\mathbb{F}_{2^5}$. We conduct a security analysis of Warbler-I in two steps. First, we perform the cryptographic statistical tests proposed by the EPC C1 Gen2 standard and the NIST standard. Then, we perform a cryptanalysis against Warbler-I by considering algebraic attacks, cube attacks, time-memory-data tradeoff attacks in great detail. Furthermore, a hardware implementation on a Xilinx Spartan-3 FPGA device shows that the new PRNG can be implemented using 46 slices.

- **Chapter 8** proposes another instance, Warbler-II, of the Warbler family, which is designed to offer a better security level. We give a detailed mathematical description of the design of Warbler-II, which also uses WG-5 transformations

in both building blocks. The combination of modified de Bruijn blocks of Warbler-II contains three primitive NLFSRs of lengths 19, 21 and 22. The security analysis of Warbler-II is conducted in a similar way to that of Warbler-I, which is twofold. In the first step, we performed the statistical tests proposed by the EPC C1 Gen2 standard and the NIST standard. In the second step, we investigate the attack resistance properties of Warbler-II by considering algebraic attacks, cube attacks, time-memory-data tradeoff attacks, Mihaljević *et al.*'s attacks, and weak internal state and fault injection attacks. A hardware implementation in VHDL for the low-cost Spartan-3 XC3S50 FPGA device shows that Warbler-II can be implemented using about 58 slices. Warbler-II can be used as a random number generator in the EPC C1 Gen2 standard tag identification protocol and RFID authentication protocols.

- **Chapter 9** summarizes the key contributions of this thesis, and presents future research directions related to the research.

Chapter 2

Literature Review

This chapter presents the existing methods for the generation of span n sequences and de Bruijn sequences, and the random number generators for the EPC C1 Gen2 passive RFID tags. A one-to-one correspondence exists between a de Bruijn sequence and a span n sequence or modified de Bruijn sequence: A span n sequence can be obtained from a de Bruijn sequence by removing one zero from the run of zeros of length n, and likewise, a de Bruijn sequence can be obtained from a span n sequence by adding one zero to the run of zeros of length $n-1$. We refer the reader to Chapter 3 for the explanation of the technical terms used in this chapter.

2.1 Existing de Bruijn Sequence Generation Methods

Construction of a de Bruijn sequence is a mathematical problem and it is widely investigated in the literature. Several authors studied the problem from graph theoretic, algorithmic, and feedback shift register points of view. The FSR-based de Bruijn sequence generation technique is the simplest one among them from an implementation point of view. Plenty of publications in the literature have been discussed various techniques about producing de Bruijn sequences, e.g., [2, 15, 17, 32–39, 53, 59, 67, 81, 83, 91, 92, 107, 111]. There are two main challenges for the

production of de Bruijn sequences: One challenge is to produce plenty of de Bruijn sequences of the same period using different recurrence relations, another one aims to produce long de Bruijn sequences that can have many practical applications in cryptography to design stream ciphers and PRSGs. Most of the existing techniques are concerned about the production of different de Bruijn sequences of the same period, and that are not efficient for generating long period de Bruijn sequences [2, 15, 32, 36, 53, 59, 111]. A few recent publications consider the production of long period de Bruijn sequences using NLFSRs by the exhaustive search [27, 99]. The problem of generating a de Bruijn sequence efficiently for large values of n still needs to be investigated for the practical applications of de Bruijn sequences. In this thesis, we concentrate on the generation of long de Bruijn sequences.

2.1.1 D-homomorphism Based de Bruijn Sequence Construction

In 1970, Lempel [67] introduced the idea of D-morphic image and preimages of a binary sequence. As an application of D-morphic preimages, Lempel presented a construction of producing a de Bruijn sequence of period 2^{n+1} from a de Bruijn sequence of period 2^n by first computing D-morphic preimages of the de Bruijn sequence of period 2^n and then concatenating two preimages at a conjugate pair.

The conception of the composition of two feedback functions f and g was suggested by Green and Dimond [50] in 1970 and independently by Mykkeltveit in 1976 [90]. Later on, in 1979, Mykkeltveit *et al.* [92] studied the cycle structures of f composed with g and g composed with f, and presented the construction of Lempel [67] in the form of a composited recurrence relation of a de Bruijn sequence.

Annexstein [2] proposed a recursive algorithm based on the above ideas of Lempel for generating a long de Bruijn sequence. In the algorithm, a long de Bruijn sequence is obtained by repeatedly computing preimages of (lower order) de Bruijn sequences and concatenating at the conjugate pair. This algorithm is not efficient for producing a long de Bruijn sequence of period 2^n, say $n = 40$. Chang *et al.* [17] proposed another algorithm based on Lempel's D-homomorphism for producing a

long de Bruijn sequence from a short de Bruijn sequence.

Games [39] proposed a generalized construction of de Bruijn sequences in which a de Bruijn sequence of period 2^{n+1} is constructed from two different de Bruijn sequences of period 2^n. In the construction, two D-morphic preimages p_0 and p_1 of a de Bruijn sequence $\mathbf{r}$ of period 2^n and D-morphic preimages q_0 and q_1 of another de Bruijn sequence $\mathbf{s}$ of period 2^n are obtained. Then a de Bruijn sequence of period 2^{n+1} is constructed from either p_0 and q_1 or p_1 and q_0 by concatenating preimages p_0 and q_1 or p_1 and q_0 at a conjugate pair. This can be regarded as Lempel's idea of concatenating at the conjugate pair. Games also introduced the notion of reverse-complementary de Bruijn sequences, and a reverse-complementary de Bruijn sequence can be constructed using $\mathbf{r}$ and $\mathbf{s}$ where $\mathbf{s}$ is a reverse de Bruijn sequence of de Bruijn sequence $\mathbf{r}$ with the images of D-morphic preimages of $\mathbf{s}$ and $\mathbf{r}$ are the same.

Mandal and Gong [71] recently analyzed the composited de Bruijn sequences produced by the composited construction by exploiting the higher order D-morphic preimages of binary sequences. Moreover, a new iterative technique with its parallel extension for evaluating the feedback function of the composited construction is proposed. The D-morphic analysis and the efficient iterative technique are not included in this thesis.

2.1.2 Cycle Joining Algorithms for de Bruijn Sequences

Jansen *et al.* [59] presented an algorithm based on the principle of joining cycles for generating de Bruijn sequences using feedback shift registers. The feedback function $g = (f + q)$ of a de Bruijn sequence is composed of two function modules, one is a feedback function f, and another function q is constructed using the feedback function f. They showed that $O(2^{\frac{2n}{\log(2n)}})$ de Bruijn sequences of period 2^n can be produced by considering the feedback functions for all irreducible polynomials in a feedback shift register. The storage requirement for the implementation of the cycle joining method is $3n$ bits, and $4n$ shifts are required to generate one bit of a de Bruijn sequence.

Yang and Dai [111] proposed a construction of an m-ary de Bruijn sequence

based on joining the cycles using modification sets of a feedback function f. In the construction, a nonlinear feedback function F of a de Bruijn sequence is constructed from the feedback function f using the modification sets of f. This method is not efficient for large values of n, since the method requires the cycles decomposition of f to construct the function F, and for a large n, it is very hard to obtain the cycle decomposition of f. Moreover, the feedback function would contain many product terms for joining of the cycles. The aim of the authors is to construct a number of feedback functions that can generate de Bruijn sequences. They showed that at least $2^{(\frac{m^n}{n}-mn)}$ feedback functions that generate de Bruijn sequences can be constructed by choosing $f = x_0$.

Hauge and Helleseth [53] proposed a technique based on an irreducible polynomial and its adjacency graph to generate de Bruijn sequences. In this technique, a de Bruijn sequence is obtained as maximum spanning trees from the adjacency graph of a feedback function corresponding to an irreducible polynomial. The lower bound for the number of de Bruijn sequences is determined in terms of the cyclotomic numbers.

2.1.3 Algorithmic Approach for the de Bruijn Sequence Generation

Fredricksen and Kessler [37] developed a technique based on lexicographic compositions for constructing de Bruijn sequences. In [38], Fredricksen and Maiorana presented an algorithm for generating necklaces of length n in k colors, and a k-ary de Bruijn sequence of period k^n is produced by juxtaposing in order the periodic reductions of the necklaces.

Fredricksen [35] proposed an algorithm for generating nonlinear de Bruijn sequences, and the algorithm requires $3n$ units of storage and outputs one bit in around n units of time. Fredricksen also showed that a new de Bruijn sequence can be obtained from a de Bruijn sequence by the method of cross-joining. Moreover, Fredricksen demonstrated that a set of new 2^{2n-5} de Bruijn sequences can be obtained from a de Bruijn sequence by the cross-joining pairs. The storage re-

quirement for the implementation of the cross-join method is about $6n$ units. A detailed summary of many other de Bruijn sequence generation techniques can be found in [36].

Etzion and Lempel [32] presented a construction of de Bruijn sequences, where the linear complexity of a de Bruijn sequence can attain the lower bound $(2^{n-1}+n)$ for all $n \geq 3$.

A nonsingular feedback shift register $f(x_0, x_1, ..., x_{n-1}) = x_0 + g(x_1, ..., x_{n-1})$ can be used to generate de Bruijn sequences. Fredricksen [36] first characterized the feedback functions of de Bruijn sequences by the Hamming weight of function g. In [81], Mayhew presented the distribution of the feedback functions of de Bruijn sequences for the odd weights of g for $4 \leq n \leq 6$ and $n = 7$ (partial results).

2.1.4 Linear Span Based de Bruijn Sequence Construction

Chan *et al.* [14] first proved the linear complexity of a de Bruijn sequence of period 2^n lies in the range of $(2^{n-1} + n)$ and $(2^n - 1)$. Etzion and Lempel [32] showed a construction of de Bruijn sequences that can attain the minimal linear complexity value $(2^{n-1}+n)$. Games [39] presented a special construction of a de Bruijn sequence of period 2^{n+1} with maximum linear complexity $2^{n+1} - 1$ from a de Bruijn sequence of period 2^n with maximum linear complexity. A summary on the linear complexity of de Bruijn sequences can be found in [31].

2.2 Span n Sequence Generation by the Exhaustive Search Method

We provide a review on the existing techniques for the generation of span n sequences using nonlinear feedback shift registers. Golomb [44] introduced the term *span n sequence* for a sequence of period $2^n - 1$ in which every nonzero binary n-tuple occurs exactly once in a period. A span n sequence is also known as a *modified de Bruijn sequence*. Mayhew and Golomb first studied the characteristic of span n sequences produced by NLFSRs and their feedback functions [80, 82].

2.2.1 Exhaustive Search for Small Span n Sequences

Mayhew and Golomb [80] presented the upper and lower bounds on the linear span of a span n sequence, and showed the (LFSR) characteristic polynomial of a span n sequence is a product of irreducible polynomials of degree between 1 and n. They categorized the number of span n sequences for different values of the linear span for $4 \leq n \leq 6$, where the span n sequences are found by computer simulations. The experimental result shows that the linear span of a span n sequence generated by an NLFSR lies in the range of $3n$ and $2^n - 2$. In [82], they characterized the nonlinear feedback functions, and presented the recurrence relation for a reverse span n sequence when the recurrence relation of the span n sequence is known. Mayhew and Golomb also classified the number of span n sequences according to the number of monomials in a feedback function for the same values of n.

2.2.2 Span n Sequence Generation Using Quadratic Feedback Function

Chan *et al.* [16] considered the quadratic feedback functions to generate span n sequences. The quadratic functions are of two types: 1) a function in 4-variable has only three terms, two linear terms and one term of degree 2, 2) a function in 2-variable has only two terms, one linear term and one quadratic term. They reported the number of span n sequences for $4 \leq n \leq 7$.

Dubrova [27] presented a few span n sequences of period $2^n - 1$, for $4 \leq n \leq 24$. All span n sequences are generated by nonlinear feedback functions with few linear terms and one or two quadratic terms. Note that all the techniques use an exhaustive search to verify whether the feedback function generates a span n sequence.

2.2.3 Span n Sequence Generation Using Cubic and Quartic Feedback Functions

Gammel *et al.* [41] proposed stream cipher Achterban:128/80 based on nonlinear feedback shift registers where each NLFSR generates a span n sequence. They

presented thirteen NLFSRs for span n sequences of periods in the range of $2^{21}-1$ and $2^{33}-1$, and the feedback functions for the NLFSRs contain only few monomials. Another two variants of Achterban contains eight and ten NLFSRs that also generate span n sequences of periods in the range of $(2^{21}-1)$ and $(2^{32}-1)$ [40,42].

Some recent studies on the generation of span n sequences can be found in [27, 99]. Rachwalik *et al.* presented seven span n sequences of periods $(2^{25}-1)$ and $(2^{27}-1)$ in [99]. The feedback functions of the NLFSRs contains few terms, and that are of degree three and four. These NLFSRs were found by an exhaustive search using an FPGA implementation.

2.2.4 General Studies on Span n Sequences

The period of a sequence produced by an NLFSR trace generator is investigated by Ng in [95] where a nonlinear feedback function of the NLFSR trace generator is the sum of a linear term and a trace function in $(n-1)$ variables with different decimations. The NLFSR trace generator can generate a span n sequence for a proper combination of a decimation number and a basis of the finite field where the trace function is defined. The number of span n sequences produced by the NLFSR trace generator is reported for $n = 7, 8, 9$ and 10 in [95].

Gong [48] studied the randomness properties of span n sequences where a span n sequence is viewed as an output of a filtering generator composed of an LFSR and a filtering function. It is well-know that m-sequences are a class of span n sequences generated by LFSRs. A de Bruijn sequence can also be constructed from an m-sequence using their one-to-one correspondence. When a de Bruijn sequence is constructed from an m-sequence, the linear complexity of the de Bruijn sequence is at least $(2^{n-1}+n)$ [14]. An attacker can remove one zero from the run of zeros of length n of the de Bruijn sequence, then it again becomes an m-sequence with linear complexity n. The linear complexity of the sequences fluctuates drastically. Gong suggested to study the randomness properties of span n sequences instead of de Bruijn sequences for cryptographic applications.

2.3 RNG for the EPC C1 Gen2 Standard

Over the last few years, a number of random number generators have been proposed for the EPC C1 Gen2 passive RFID tags [4,18,57,64,85,97,109]. A random number generator (RNG) is the only component in an RFID tag for providing security functionalities. Random number generators can be classified into two categories, namely true random number generators (TRNGs) and pseudorandom number generators (PRNGs). True random number generators are implemented by relying on a physical process/phenomenon, and some instances of TRNG can be found in [4, 57, 109]. Pseudorandom number generators, on the other hand, are designed using complex nonlinear mathematical relations [64, 97]. Other than these two types of proposals, another type of proposal can be found in the literature, and that is composed of a true random number generator and a pseudorandom number generator [18, 85].

2.3.1 TRNG Based RNG Proposals

Che *et al.* [18] designed a PRNG based on a combination of an oscillator-based TRNG and a linear feedback shift register with 16 stages. Randomness in a 16-bit number is introduced by adding one true random bit to each bit of the 16-bit random number. In 16 clock cycles, a 16-bit random number is generated by the PRNG. Due to the linear structure, Che *et al.*'s scheme has been attacked by Melia-Segui *et al.* in [85] with high success probability.

Melia-Segui *et al.* [85] proposed a PRNG based on multiple primitive polynomials in an LFSR in order to avoid the linear structure. The PRNG module is comprised of an LFSR of 16 stages with eight primitive polynomials and a TRNG where a primitive polynomial is chosen according the TRNG in a clock cycle. A hardware implementation of the PRNG requires 761 gate equivalents (GE) where the cost of the TRNG is not included, and a 16-bit random number is produced within 16 clock cycles. Melia-Segui *et al.* [86] recently proposed J3Gen, which contains four PRNGs based on the same design principle. The lengths of the internal states of four LFSRs in the PRNGs are 16, 24, 32, 64, and each LFSR contains either 8, 16 or 32 primitive polynomials.

2.3.2 Pseudorandom Number Generator Proposals

Peris-Lopez *et al.* [97] proposed a PRNG named LAMED for RFID tags, which can provide 32-bit random numbers as well as 16-bit random numbers. The internal state of LAMED is of 64 bits, and the operations involved to update the internal states are bitwise XOR operations, modular algebra, and bit rotations. A compact hardware implementation of LAMED requires $1,585$ GE and LAMED can produce random numbers each 1.8 ms.

Martín *et al.* [78] proposed two pseudorandom number generators, named AKARI-1/2 based on T-function, introduced by Klimov and Shamir in [64]. The internal state of AKARI-1/2 is of m bits, $m = 2^i, 3 \leq i \leq 9$ and the operations used to update the internal state are modular addition, multiplication, shift operation, bitwise AND and OR. For AKARI-1 and AKARI-2, an $m/2$-bit random number is obtained by taking lower-half from the m-bit internal state after applying 64-round and 50-round of a state update schedule, respectively. They presented the hardware implementation of both AKARI-1/2 in [78].

2.4 Summary of Chapter 2

In this chapter, we reviewed the existing methods for generating de Bruijn sequences and span n sequences. Current method for verifying the span n property of a binary sequence of period (2^n-1) is the exhaustive search method whose time complexity is exponential in n. Most proposed techniques for generating de Bruijn sequences are not efficient for a large value of n. The pseudorandom number generator proposals for the EPC Class 1 Gen2 RFID tags are presented.

Chapter 3

Preliminaries

In this chapter, we recall the mathematical concepts related to finite fields, sequences, and Boolean functions that we use to design and describe pseudorandom sequence and number generators. The theory of finite fields and feedback shift registers presented here can be found in [43, 45, 68].

3.1 Finite Fields

We denote by $\mathbb{F}_2 = \{0, 1\}$ the Galois field with two elements. Let $\mathbb{F}_{2^t}$ $(t \geq 2)$ be an extension field or a finite finite with 2^t elements, which is defined by a defining element α where α is a root of an irreducible polynomial.

A polynomial $f(x)$ of degree t over $\mathbb{F}_2$ is called a *primitive polynomial* if $f(x)|(x^{2^t-1}+1)$ but $f(x) \nmid (x^r+1)$ when $r < 2^t - 1$, and the *order* of $f(x)$ is $2^t - 1$. For a positive t, the number of primitive polynomials of degree t is equal to $\frac{\phi(2^t-1)}{t}$, where $\phi(\cdot)$ is Euler's phi function. We always define the finite field $\mathbb{F}_{2^t}$ by a primitive polynomial over $\mathbb{F}_2$. It is well-known that $\mathbb{F}_{2^t}$ is isomorphic with $\mathbb{F}_2^t = \{(x_0, x_1, \cdots, x_{t-1}) : x_i \in \mathbb{F}_2\}$, a vector space with 2^t elements over $\mathbb{F}_2$.

Definition 1 *Let α be a defining element of $\mathbb{F}_{2^t}$, which is a root of an irreducible polynomial $p(x)$, i.e., $p(\alpha) = 0$, where $p(x)$ is irreducible of degree t over $\mathbb{F}_2$. Then, the polynomial basis of $\mathbb{F}_{2^t}$ is given by $\underline{\alpha} = \{1, \alpha, \alpha^2, ..., \alpha^{t-1}\}$.*

According to the polynomial basis $\underline{\alpha}$, the field $\mathbb{F}_{2^t}$ is represented by

$$\mathbb{F}_{2^t} = \{c_0 + c_1\alpha + \cdots + c_{t-1}\alpha^{t-1} : \ c_i \in \mathbb{F}_2\}.$$

In this thesis, we always take α to be a root of a primitive polynomial $p(x)$ over $\mathbb{F}_2$, and the polynomial basis $\underline{\alpha}$ is used to represent the elements of $\mathbb{F}_{2^t}$.

Definition 2 *The cyclotomic coset of s modulo (2^t-1) is defined as $C_s = \{s, 2s, ..., 2^{j-1}s\}$ where j is the smallest number such that $s \equiv 2^j s \ mod \ 2^t - 1$. The smallest element in C_s is called the coset leader of C_s.*

We define a set D_t that contains the coset leaders as

$$D_t = \{d \ : \gcd(d, 2^t - 1) = 1, \ d \text{ is a coset leader}\}$$

where $\gcd(a, b)$ is the greatest common divisor of a and b. Then the size of D_t, denoted by $|D_t|$, is equal to $= \frac{\phi(2^t-1)}{t}$.

Definition 3 *The trace function $\mathrm{Tr} : \mathbb{F}_{2^t} \to \mathbb{F}_2$ is defined by*

$$\mathrm{Tr}(x) = x + x^2 + x^{2^2} + \cdots + x^{2^{t-1}}, \ x \in \mathbb{F}_{2^t}.$$

3.2 Feedback Shift Register Sequences

This section presents the fundamental concepts related to feedback shift register sequences.

3.2.1 Basic Definitions and Properties of NLFSRs

A binary sequence $\{a_i\}_{i\geq 0}$ can be generated by an n-stage feedback shift register whose recurrence relation is defined as [43]

$$a_{n+k} = f(a_k, a_{k+1}, \cdots, a_{n+k-1}), a_i \in \mathbb{F}_2, \ k \geq 0 \tag{3.1}$$

where $(a_0, a_1, \cdots, a_{n-1})$ is called an *initial state* and $S_k = (a_k, a_{k+1}, \cdots, a_{n+k-1})$ is called the *k-th state* of the shift register. An n-stage feedback shift register is also called a *feedback shift register of length n*. The sequence $\{a_i\}_{i\geq 0}$ is called a *linear feedback shift register (LFSR) sequence* if the function f is linear and is of the form

$$f(x_0, x_1, \cdots, x_{n-1}) = c + c_0x_0 + c_1x_1 + \cdots + c_{n-1}x_{n-1},\ c, c_i \in \mathbb{F}_2.$$

Otherwise, it is called a *nonlinear feedback shift register (NLFSR) sequence.*

Definition 4 *The sequence* $\{a_0, a_1, \cdots, a_{T-1}, \cdots\}$ *is called* periodic *with period* T *if* $a_i = a_{i+T}, i \geq 0$.

For generating a periodic sequence, a feedback function must have the form given in the following theorem.

Theorem 3.2.1 *[43] Let* $\boldsymbol{a} = \{a_i\}$ *be a binary sequence generated by the recurrence relation* $a_{n+i} = f(a_i, a_{i+1}, \cdots, a_{i+n-1})$. *Then the sequence* $\boldsymbol{a}$ *is periodic if and only if f is written as*

$$f(a_i, a_{i+1}, \cdots, a_{i+n-1}) = a_i + g_1(a_{i+1}, \cdots, a_{i+n-1}) \tag{3.2}$$

where g_1 *is a Boolean function in* $(n-1)$ *variables.*

A recurrence relation of the above form is called a *nonsingular recurrence relation.* Denoting the left shift operator by L. For a periodic sequence $\mathbf{a} = \{a_i\}_{i\geq 0}$ with period T, the *k-th shift* of sequence $\mathbf{a}$ is defined as $L^k(\mathbf{a}) = \{a_k, a_{k+1}, ..., a_{T-1}, a_0, ..., a_{k-1}\}$ and the sequence $L^k(\mathbf{a})$ is called the k-th shifted sequence of $\mathbf{a}$.

Definition 5 *A binary sequence with period* $2^n - 1$ *generated by an n-stage linear feedback shift register is called an* m-sequence.

Assume that α is a primitive element of $\mathbb{F}_{2^n}$, then an m-sequence $\{a_i\}$ of period $(2^n - 1)$ can be written as $a_i = \mathrm{Tr}(\alpha^i),\ i = 0, 1, ..., 2^n - 2$.

Let $\mathbf{a} = \{a_i\}$ and $\mathbf{b} = \{b_i\}$ be two periodic binary sequences with period T_1 and T_2, respectively. Then the *crosscorrelation* between $\mathbf{a}$ and $\mathbf{b}$ over $\mathbb{F}_2$ is defined

as [45]

$$C_{\mathbf{a},\mathbf{b}}(\tau) = \sum_{i=0}^{T} (-1)^{a_{i+\tau}+b_i}, \tau = 0, 1, ...,$$

where T is the least common multiple of T_1 and T_2. If $\mathbf{b} = \mathbf{a}$, then $C_{\mathbf{a},\mathbf{a}}(\tau)$ is called *autocorrelation* of $\mathbf{a}$ [45].

Definition 6 *The* linear span or linear complexity *of a sequence is the length of the shortest LFSR that generates the entire sequence.*

Linear complexity is an important property of a sequence and that measures the unpredictability of a sequence. For a sequence to be useful in cryptographic applications, the sequence must have a large linear complexity. Note that an m-sequence has good randomness properties: period $(2^n - 1)$, balance, ideal run distribution, k-tuple distribution and ideal 2-level autocorrelation. But, it has the least linear complexity n, i.e., if only any $2n$ consecutive bits are known from the entire sequence, then it is possible to determine the characteristic polynomial uniquely by the Berlekamp-Massey algorithm [79], thereby the entire sequence can be reconstructed. On the other hand, a random sequence has linear complexity approximately half the length of the sequence [102].

3.2.2 Golomb's Randomness Postulates

For a periodic binary sequence $\{a_i\}$ with period T, the randomness of the binary sequence can be measured by the following properties [43, 45]:

1. In a period, the difference between the number of zeros and the number of ones is almost equal, i.e., $|\sum_{i=0}^{T-1}(-1)^{a_i}| \leq 1$.

2. In a period, half the runs have length 1, one fourth have length 2, $\frac{1}{2^k}$ runs have length k, and so on.

3. The autocorrelation of the sequence is two valued and defined by

$$C(\lambda) = \begin{cases} N & \text{if} \quad \lambda \equiv 0 \bmod T \\ K & \text{if} \quad \lambda \not\equiv 0 \bmod T \end{cases}$$

for odd T, $K = -1$ and even T, $K = 0$.

These three properties are known as *Golomb's randomness postulates.*

3.2.3 Relationship Between de Bruijn Sequences and Span n Sequences

Definition 7 *A binary sequence of period* 2^n *is called a* de Bruijn sequence *if all binary n-tuples occur exactly once in a period.*

Example 1 For $n = 4$, the feedback function $f(x_0, x_1, x_2, x_3) = 1+x_0+x_1+x_1x_2+x_1x_3+x_1x_2x_3$ in recurrence relation 3.1 generates the following de Bruijn sequence of period $2^4 = 16$. The de Bruijn sequence is $\{1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0\}$.

Definition 8 *A binary sequence of period* $2^n - 1$ *is called a* span n sequence *or* modified de Bruijn sequence *if every nonzero n-tuple occurs exactly once in a period.*

Example 2 For $n = 4$, the feedback function $f(x_0, x_1, x_2, x_3) = x_0+x_2+x_3+x_2x_3$ in recurrence relation 3.1 generates a span n sequence of period $(2^4 - 1) = 15$. The span n sequence is $\{1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0\}$.

Definition 9 *A feedback function of an n-stage FSR that generates a span n sequence of period* $2^n - 1$ *is called a* primitive feedback function.

Definition 10 *A nonlinear feedback function of an n-stage NLFSR that generates a span n sequence of period* $2^n - 1$ *is called a* primitive feedback function*, and the NLFSR is called a* primitive NLFSR.

We remember that the "span n" property and "linear span" property are two different properties of a sequence. The *span n* property of a sequence is referred to as when all nonzero n-tuples occur exactly once in a period of the sequence, and the *linear span* or *linear complexity* of a sequence is the measure of unpredictability.

A one-to-one correspondence between a de Bruijn sequence and a span n sequence is provided in the following theorem.

Proposition 3.2.2 *[43] Let f be a feedback function in n variables that generates a span n sequence, then the function $h = f + \prod_{i=1}^{n-1}(x_i + 1)$ generates a de Bruijn sequence.*

An m-sequence generated by an LFSR is a span n sequence, but a span n sequence generated by an NLFSR is not an m-sequence. It has been conjectured that only m-sequences have 2-level autocorrelation [44]. The truth of the conjecture signifies that a span n sequence generated by an NLFSR cannot have 2-level autocorrelation. Nonlinearly generated de Bruijn and span n sequences have excellent randomness properties. A binary span n sequence generated by an NLFSR has the randomness properties: long period $2^n - 1$, balance, and n-tuple distribution. A span n sequence may also have high linear span [44, 80].

Definition 11 *The* minimal polynomial *of a sequence $\boldsymbol{a}$ is defined by the characteristic polynomial of the LFSR of shortest length that can generate the sequence, and the* degree *of the minimal polynomial determines the linear complexity of the sequence $\boldsymbol{a}$.*

Property 1 *The linear span of a de Bruijn sequence, denoted as LS_{db}, is bounded by [14]*

$$2^{n-1} + n \leq LS_{db} \leq 2^n - 1. \tag{3.3}$$

On the other hand, the linear span of a span n sequence, denoted as LS_s, is bounded by [80]

$$2n < LS_s \leq 2^n - 2. \tag{3.4}$$

From this property, we say that a span n sequence has the *optimal or near-optimal linear span* if its linear span is equal to $2^n - 2$ or close to $2^n - 2$. Similarly, we call a de Bruijn sequence has the *optimal or near-optimal linear span* if its linear span is equal to $2^n - 1$ or close to $2^n - 1$.

Definition 12 *Two sequences are called* shift distinct *of each other if one sequence can not be obtained from the shifted version of another.*

For an LFSR sequence produced by a feedback function $f(x_0, x_1, ..., x_{n-1}) = \sum_{i=0}^{n-1} c_i x_i$, the periodicity of the sequence can be determined by calculating the period of the polynomial $p(x) = c_0 + c_1 x + \cdots + c_{n-1}x^{n-1} + x^n$ over the field $\mathbb{F}_2$. When the feedback function f is nonlinear, determining the periodicity of a sequence produced by f is an unsolved problem.

3.2.4 Unsolved Problems on Synthesis of NLFSRs

Due to the existence of polynomial time algorithms, determining the period of a univariate polynomial is possible in polynomial time. As a result, the period of an LFSR sequence can be determined certainly. For a nonlinear feedback function, there is no polynomial time algorithm for checking the primitivity of a nonlinear feedback function. The exhaustive search is the only method for checking the primitivity of an NLFSR, and its time complexity is exponential in the length of the NLFSR. Therefore, determining the period of an NLFSR sequence is infeasible for a long length of the NLFSR in real time. Most of the known results on the synthesis of NLFSRs are collected in Golomb's book [43]. Following problems are still open since last five decades for a nonlinear feedback function in the theory of NLFSRs.

1. There is no known general construction of an NLFSR that can generate a span n sequence or a de Bruijn sequence.

2. For a given positive integer P, no construction of a feedback function that can generate sequences with periods bounded below by P is known.

3. There is no algorithm other than the exhaustive search for checking the primitivity of a nonlinear feedback function of an NLFSR.

3.2.5 D-homomorphisms and Compositions of NLFSRs

This section presents the notion of D-homomorphism of binary sequences and the composition of feedback functions of NLFSRs, which will be used in Chapter 5.

The D-morphisms of Binary Sequences

In 1970, the idea of the D-homomorphism (D-morphism in brief) and its inverse of a binary sequence was first introduced by Lempel in [67]. As an application, Lempel showed that the preimages of a de Bruijn sequence of period 2^n can be used to construct another de Bruijn sequence of period 2^{n+1}. Let $\mathbf{a} = (a_0, a_1, a_2, ..., a_{N-1})$ be a binary sequence of length $N(\geq 1)$. The *first order* D-morphic image of $\mathbf{a}$ is defined as [67]

$$D(\mathbf{a}) = (a_0 + a_1, a_1 + a_2, a_2 + a_3, ..., a_{N-2} + a_{N-1}).$$

The D-morphic preimages of a binary sequence $\mathbf{a}$ are given by [67]

$$z = (z_i) = (0, a_0, a_0 + a_1, a_0 + a_1 + a_2, ..., \sum_{i=0}^{N-1} a_i) \text{ and } \bar{z} = (\bar{z}_i), \bar{z}_i = z_i + 1.$$

Cycle Decomposition of NLFSRs

A nonsingular recurrence relation with an initial state generates a sequence of states and ends with the initial state. The sequence of states is called a *cycle* [43]. A nonsingular feedback shift register with a feedback function f partitions the space of 2^n binary n-tuples into a finite number of cycles, which is known as the *cycle decomposition* or *cycle structure* of f, and we denote by $\Omega(f)$ the cycle decomposition of f [43]. Each cycle in $\Omega(f)$ is nothing but a periodic sequence. The nonsingular recurrence relation generates the same cycle for any n-tuple of the cycle, and it generates a different cycle when an n-tuple initial state is not in that cycle. For the details of the cycle decomposition, see [43]. For an arbitrary feedback function, the method other than the exhaustive search of determining the number of cycles produced by the feedback function is unknown in general.

In particular, the cycle decomposition of a feedback function that generates a span n sequence contains only two sequences, namely a span n sequence and the zero sequence. Similarly, the cycle decomposition of a feedback function that generates a de Bruijn sequence contains only one sequence, the de Bruijn sequence.

Composition of Recurrence Relations

Let $g(x_0, x_1, ..., x_{n-1}, x_n) = x_0 + G(x_1, x_2, ..., x_{n-1}) + x_n = 0$ and $f(x_0, x_1, ..., x_{m-1}, x_m) = x_0 + F(x_1, x_2, ..., x_{m-1}) + x_m = 0$ be two recurrence relations of n and m stages, respectively that generate periodic sequences, where G and F are Boolean functions in $(n-1)$ and $(m-1)$ variables, respectively. Then, a *composite recurrence relation*, denoted as $g \circ f$, is defined by [92]

$$g \circ f = g(f(x_0, ..., x_m), f(x_1, ..., x_{m+1}), ..., f(x_n, ..., x_{m+n-1})) = 0,$$

which is a recurrence relation of $(n+m)$ stages. The operation "$\circ$" is regarded as the composition operation of recurrence relations. Note that $g \circ f$ and $f \circ g$ are not the same in general for nonlinear feedback functions. For any feedback function f, the cycle decomposition of g is a subset of the cycle decomposition of $g \circ f$. For more detailed treatments on the cycle decomposition of a composite recurrence relation, see [92].

In 1979, Mykkeltveit *et al.* [92] presented the construction of Lempel [67] for producing a de Bruijn sequence of period 2^{n+1} from a de Bruijn sequence of period 2^n in terms of the composition of recurrence relation.

Lemma 3.2.3 *[92] Let p be a characteristic polynomial, and $q(x_0, ..., x_n) = x_0 + x_n + w(x_1, ..., x_{n-1})$ where w is a Boolean function in $(n-1)$ variables and let $a \in \Omega(q)$ and $x \in \Omega(q \circ p)$. If the minimal polynomial of a is coprime with p, then $x = b + c$ where b's minimal polynomial is the same as the minimal polynomial of a and c's minimal polynomial is p.*

Theorem 3.2.4 *[92] Let $g = x_0 + x_n + f(x_1, ..., x_{n-1})$, which generates a de Bruijn sequence with period 2^n and let $\psi(x_0, x_1) = x_0 + x_1$. Then both $h_1 = g \circ \psi + \prod_{i \in Z_o^n} x_i \prod_{i \in Z_e^n} (x_i + 1)$ and $h_2 = g \circ \psi + \prod_{i \in Z_o^n} (x_i + 1) \prod_{i \in Z_e^n} x_i$ generate de Bruijn sequences with period 2^{n+1}.*

We denote by $e_i = (1, 1, 0, 1, 0, ..., 1, 0) \in \mathbb{F}_{2^i}$ and $\hat{e}_i = (0, 1, 0, 1, 0, ..., 1, 0) \in \mathbb{F}_{2^i}$ the conjugate pair for an i-stage NLFSR.

3.3 Boolean Functions

In this section, we define some definitions which are used to characterize a Boolean function in cryptography. The concepts related to Boolean functions presented here can be found in [23].

3.3.1 Nonlinearity of Boolean Functions and Vector Boolean Functions

Let $f(x_0, ..., x_{n-1})$ be a Boolean function in n variables. A Boolean function f is called balanced if the truth table of f contains equal number of 0's and 1's.

Definition 13 *The* algebraic normal form *of a Boolean function f in n variables is defined as*

$$f(x_0, x_1, ..., x_{n-1}) = a_0 + \sum_{i=0}^{n-1} a_i x_i + \sum_{0 \le i < j \le n-1} a_{i,j} x_i x_j + \cdots + a_{i_1,i_2,...,i_{n-1}} x_{i_1} x_{i_2} ... x_{i_{n-1}}$$

where $a_0, a_1, ..., a_{i_1,i_2,...,i_{n-1}} \in \mathbb{F}_2$ are called coefficients. The algebraic degree *of the Boolean function is defined as the number of variables in the highest nonzero coefficient. A Boolean function of the form $f(x_0, x_1, ..., x_{n-1}) = a + \sum_{i=0}^{n-1} a_i x_i$ is called an affine or linear function.*

Definition 14 *The* support *of a Boolean function f, denoted as $Supp(f)$, is defined as the set of all inputs for which $f(x) = 1, x \in \mathbb{F}_{2^n}$.*

Definition 15 *The* Hamming weight *of a Boolean function f, denoted as $H(f)$, is defined as the number of ones in the truth table of the function f. In other words, the Hamming weight of f is the cardinality of $Supp(f)$.*

The *Hadamard* (or Walsh or Fourier) transform of f is defined by

$$\hat{f}(\mathbf{w}) = \sum_{\mathbf{x} \in \mathbb{F}_2^n} (-1)^{f(\mathbf{x}) + \mathbf{w} \cdot \mathbf{x}} = \sum_{x \in \mathbb{F}_{2^n}} (-1)^{f(x) + \mathrm{Tr}(\mathbf{w} x)}$$

where $\mathbf{w} = (w_0, ..., w_{n-1}) \in \mathbb{F}_2^n$ and $\mathbf{w} \cdot \mathbf{x} = \sum_{i=0}^{n-1} w_i x_i$, the inner product of $\mathbf{w}$ and $\mathbf{x}$.

The *distance* between two binary vectors $\mathbf{a} = (a_0, ..., a_{n-1})$ and $\mathbf{b} = (b_0, ..., b_{n-1})$, denoted by $d(\mathbf{a}, \mathbf{b})$, is defined as the number of disagreements of terms of $\mathbf{a}$ and $\mathbf{b}$, i.e.,

$$d(\mathbf{a}, \mathbf{b}) = |\{i : \ a_i \neq b_i, 1 \leq i < n \ \}| \text{ or equivalently}$$

$$d(\mathbf{a}, \mathbf{b}) = H(\mathbf{a} + \mathbf{b})$$

where $H(\mathbf{x})$ is the Hamming weight of $\mathbf{x}$.

The *nonlinearity* of f, denoted as N_f, is defined by the *minimum distance* between f and all affine functions. In other words,

$$N_f = \min_{\mathbf{w} \in \mathbb{F}_2^n, c \in \mathbb{F}_2} d(f, \mathbf{w} \cdot \mathbf{x} + c)$$

or equivalently

$$N_f = 2^{n-1} - \frac{1}{2} \hat{f}_{\max}$$

where

$$\hat{f}_{\max} = \max_{\mathbf{w} \in \mathbb{F}_2^n} |\widehat{f}(\mathbf{w})|.$$

The nonlinearity of a Boolean function is an important cryptographic property and it should be high enough to prevent having a linear approximation of the Boolean function.

We say that F is an (n, m)-vectorial Boolean function or simply an (n, m)-function if it is a function mapping from $\mathbb{F}_2^n$ to $\mathbb{F}_2^m$. An (n, m)-function F can be written as [13]

$$F(x_0, \cdots, x_{n-1}) = (f_0(x_0, ..., x_{n-1}), ..., f_{m-1}(x_0, ..., x_{n-1}))$$

where f_i's are Boolean functions in n variables and known as component functions.

The *nonlinearity* of F, denoted as N_F, is defined by [13]

$$N_F = \min_{\mathbf{b} \in \mathbb{F}_{2^m}} N_{\mathbf{b} \cdot F}$$

where $\mathbf{b} \cdot F$ is the inner product. Or equivalently,

$$N_F = 2^{n-1} - \frac{1}{2}\hat{F}_{\max}$$

where

$$\hat{F}_{\max} = \max_{\mathbf{w}\in\mathbb{F}_2^n, \mathbf{b}\in\mathbb{F}_2^m} |\widehat{\mathbf{b} \cdot F}(\mathbf{w})|.$$

Let F be an (n, m)-vectorial boolean function. For any $\mathbf{a}$ $(\neq \mathbf{0}) \in \mathbb{F}_2^n, \mathbf{b} \in \mathbb{F}_2^m$, we call that F is *differentially k-uniform* [13] if the following equation has at most k solutions in $\mathbb{F}_2^n$

$$F(\mathbf{x}) + F(\mathbf{x} + \mathbf{a}) = \mathbf{b}.$$

3.3.2 Resiliency and Propagation of Boolean Functions

Let f be a Boolean function in n variables. The *additive autocorrelation* of f is defined as [45]

$$A_f(a) = \sum_{x\in\mathbb{F}_{2^n}} (-1)^{f(x)+f(x+a)},\ a \in \mathbb{F}_{2^n}.$$

We say that f has *k-order propagation* if $A_f(a) = 0$ for $1 \leq H(a) \leq k$. A Boolean function f is said to be *k-order correlation immune* if $\hat{f}(\lambda) = 0$ for $1 \leq H(\lambda) \leq k$ [105]. A balanced k-th correlation immune Boolean function is called *k-resilient* Boolean function.

3.3.3 Algebraic Immunity of Boolean Functions

Let B_n be the set consisting of all Boolean functions in n variables. The *algebraic immunity* of f, denoted by $AI(f)$, is defined as

$$AI(f) = \min_{g\in B_m} \{\deg(g) \,|\, f \cdot g = 0 \ \text{ or } (f + 1) \cdot g = 0\}$$

where $\deg(g)$ is the algebraic degree of g and $f \cdot g$ is the product of two Boolean functions f and g. For a Boolean function f in n variables, the maximum value of the algebraic immunity is equal to $\lceil \frac{n}{2} \rceil$ [21].

3.4 Some Permutations and Functions over $\mathbb{F}_{2^t}$

In this section, we review the definitions of WG transformations, five-term functions, three-term functions, monomial functions with Kasami exponents, and MCM functions over finite fields.

3.4.1 The Welch-Gong (WG) Transformation

Let $t \not\equiv 0 \bmod 3$ and k be a positive integer such that $3k \equiv 1 \bmod t$. We define the function $h : \mathbb{F}_{2^t} \to \mathbb{F}_{2^t}$ as $h(x) = x + x^{q_1} + x^{q_2} + x^{q_3} + x^{q_4}$ where q_i's are given by $q_1 = 2^k + 1, q_2 = 2^{2k} + 2^k + 1, q_3 = 2^{2k} - 2^k + 1, q_4 = 2^{2k} + 2^k - 1$. Then, the function from $\mathbb{F}_{2^t}$ to $\mathbb{F}_{2^t}$ defined by

$$\mathrm{WGP}(x) = h(x+1) + 1$$

is called the *WG permutation* and $h(x)$ is called the *five-term permutation*. We define functions from $\mathbb{F}_{2^t}$ to $\mathbb{F}_2$ as

$$WG(x) = f(x) = \mathrm{Tr}(\mathrm{WGP}(x)) = \sum_{i \in I} \mathrm{Tr}(x^i), x \in \mathbb{F}_{2^t}$$
$$g(x) = \mathrm{Tr}(h(x)),\ x \in \mathbb{F}_{2^t}$$

where $I = I_1 \cup I_2 \cup I_3 \cup I_4$, $I_1 = \{2^{\frac{k-1}{2}} + 2 + i : \ 0 \le i \le 2^{\frac{k-1}{2}} - 2\}$, $I_2 = \{2^{\frac{k+1}{2}} + 1 + 2(i + 2^{\frac{k-1}{2}}(2^{j+1} - 1) + 2^j - 1) : 0 \le j \le \frac{k-7}{2}, 1 \le i \le 2^j\}$, $I_3 = \{2^{\frac{k+1}{2}} + 1 + 2(i + 2^{\frac{k-1}{2}}(2^{\frac{k-3}{2}} - 1) + 2^{\frac{k-5}{2}} - 1) : 1 \le i \le 2^{\frac{k-5}{2}}\}$ and $I_4 = \{2^{\frac{k+1}{2}} + 1 + 2(i + 2^{\frac{k-1}{2}}(2^{\frac{k-1}{2}} - 1) + 2^{\frac{k-3}{2}} - 1) : 2 \le i \le 2^{\frac{k-3}{2}}\}$ when $m \bmod 3 = 1$, and $I = I_5 \cup I_6$, $I_5 = \{2^{2k-1} + 2^{k-1} + 2 + j : 0 \le j \le 2^{k-1} - 3\}$ and $I_6 = \{2^{2k} + 2 \cdot j + 1 : 1 \le j \le 2^{k-1} - 1\}$ when $m \bmod 3 = 2$ [76]. Then $f(x)$ is known as the *WG transformation* and $g(x)$ is the *five-term (or 5-term) function* [25, 47]. The WG transformation has good cryptographic properties such as high algebraic degree, nonlinearity, linear span, and at least 1-order resiliency. We widely use the Welch-Gong transformations in this thesis in designing pseudorandom sequence and number generators.

3.4.2 Three-Term Function

Let $t = 2k+1$ and $t \geq 5$. We denote the permutation by $h(x)$ over the field $\mathbb{F}_{2^t}$ and given by $h(x) = x + x^{2^k+1} + x^{2^k-1}$, which is known as *three-term permutation* [45]. Then the *three-term (or 3-term)* function from $\mathbb{F}_{2^t}$ to $\mathbb{F}_2$ is defined by

$$f(x) = \mathrm{Tr}(h(x)),\ x \in \mathbb{F}_{2^t}.$$

3.4.3 Monomial Function with Kasami Exponent

Let t be an odd positive integer. The *Welch-Kasami* exponent is defined as $d = 2^{2k} - 2^k + 1$, where $\gcd(k, t) = 1$. Then the function

$$h(x) = x^d, x \in \mathbb{F}_{2^t}$$

is a monomial permutation over $\mathbb{F}_{2^t}$ [25]. A *monomial function with Kasami exponent*, from $\mathbb{F}_{2^t}$ to $\mathbb{F}_2$, is defined by

$$f(x) = \mathrm{Tr}(h(x)),\ x \in \mathbb{F}_{2^t}.$$

3.4.4 MCM Polynomial

Let $m > 5$ be an odd integer and $k < m$ be odd with $\gcd(k, m) = 1$. Then the MCM polynomial, from $\mathbb{F}_{2^m}$ to $\mathbb{F}_{2^m}$, is defined as [45]

$$f_k(x) = \sum_{i=0}^{k-1} x^{(2^k+1)2^i - 2^k}$$

which is a permutation over $\mathbb{F}_{2^m}$ (Cohen and Matthews 1994). For a particular k, $\mathrm{Tr}(f_k(x^d))$ is a class of functions mapping from $\mathbb{F}_{2^m}$ to $\mathbb{F}_2$ for different values of d and different bases of the finite field $\mathbb{F}_{2^m}$.

3.5 Summary of Chapter 3

This chapter presented some concepts related to finite fields, sequences, and compositions of feedback functions which will be used in the later chapters. We reviewed some mathematical functions over finite fields that we use as feedback functions in nonlinear feedback shift registers.

Chapter 4

Span n Sequence Generation by the Structured Search

A binary span n sequence generated by an n-stage NLFSR is a sequence with randomness properties: period $2^n - 1$, balance, and ideal n-tuple distribution. A span n sequence may have a high linear span thereto. A span n sequence can be converted to a de Bruijn sequence using their one-to-one correspondence, and vice-versa. From a standpoint of linear complexity, a span n sequence with linear complexity L ($\ll 2^{n-1}$) can be converted to a de Bruijn sequence with linear complexity varies between $(2^{n-1} + n)$ and $(2^n - 1)$ by adding one zero to the run of zeros of length $n - 1$. Likewise, one can remove any zero from the run of zeros of length n from the de Bruijn sequence, then the sequence becomes the original span n sequence with linear complexity L. For an m-sequence, the lower bound of the linear complexity drops drastically from the linear complexity at least $(2^{n-1} + n)$ to $L = n$. This suggests to study the linear complexity of a span n sequence instead of the linear complexity of a de Bruijn sequence for cryptographic applications.

This chapter studies the problem of generating span n sequences using nonlinear feedback shift registers. We present the theoretical results on span n sequences in Section 4.2 and computational results on finding the number of span n sequences in Sections 4.3 - 4.4. The nonlinear recurrence relation for an NLFSR is composed of three parameters, namely a decimation number, a primitive polynomial and a t-tap

position. Finding span n sequences using this recurrence relation is called a *structured search*. In the theoretical results, we show that a feedback shift register (FSR) generates a maximum number of span n sequences when about half the length of the FSR tap positions participate in the feedback function. We also determine an approximate number of feedback functions used in the structured search. In the computational results, we use Welch-Gong (WG) transformations, three-term functions, five-term functions, monomial functions with Kasami exponent, and MCM functions in an NLFSR, and present the number of span n sequences produced by the structured search using the aforementioned functions for $6 \leq n \leq 20$. The success probability of obtaining a span n sequence in the structured search is empirically compared with the success probability of obtaining a span n sequence in a random generation method. In Section 4.5, we analyze the linear span or complexity for each class of span n sequences, and the analysis shows the linear complexity of a span n sequence lies in the range of $(2^n - 2 - 3n)$ (near-optimal) and $(2^n - 2)$ (optimal). Partial contents of this chapter can be found in [69].

4.1 Related Work and Motivation

Most of the research efforts devoted on the study of span n sequences have been concerned about the number of span n sequences and the characteristics of nonlinear feedback functions [44, 80, 83] including the number of terms in the feedback functions [82, 83] and the weight of truth tables of the feedback functions [81, 83]. A survey on the generation of span n sequences can be found in Chapter 2. Note that all the methods use an exhaustive search for verifying the primitivity of a feedback function or whether the feedback function generates a span n sequence.

For $n \geq 8$, it is difficult to employ all feedback functions in n variables in an NLFSR, and produce all span n sequences due to the huge number of functions. Our goal is to use a class of feedback functions with t ($< n$) variables in an n-stage NLFSR and produce a number of span n sequences, where the class of feedback function is composed of a permutation polynomial and a trace function, and a decimation number. Finding span n sequences in this technique is called a structured

search, since all the feedback functions of the NLFSR have a special representation. Another aim of the structure search is to obtain long span n sequences that can be used to design lightweight PRNGs and stream ciphers. For a feedback function of an NLFSR, the primitivity of the feedback function is verified by the exhaustive search method. In the structured search, we prefer to use small values of t for an efficient implementation of an NLFSR.

4.2 Theoretical Results on Span n Sequences

This section presents some theoretical results on the structured search for producing span n sequences. We first describe the recurrence relation of nonlinear feedback shift registers whose feedback functions are composed of a permutation and a trace function over a finite field. In an n-stage NLFSR, the feedback function is a Boolean function in t variables where $5 < t \leq n-1$. All the feedback functions in t variables are balanced as a function composed by a permutation and trace function is balanced and have even Hamming weight 2^{t-1}. Thus, the new span n sequences generated by a class of feedback functions belong to the weight class 2^{n-2}. Then, we calculate the approximate number of feedback functions used in the structured search.

4.2.1 Description of a Span n Sequence Generation Procedure

Let $\mathbf{a} = \{a_i\}$ be a binary sequence generated by an n-stage nonlinear recurrence relation, which is defined as

$$a_{n+k} = a_k \oplus f_d(x_k) = a_k \oplus \mathrm{Tr}(P(x_k^d)),\ x_k = (a_{r_1+k}, a_{r_2+k}, \ldots, a_{r_t+k}) \in \mathbb{F}_{2^t},$$
$$d \in D_t^*,\ 0 < t < n,\ k \geq 0 \tag{4.1}$$

where $(r_1, r_2, ..., r_t)$ with $0 < r_1 < r_2 < \ldots < r_t \leq n-1$ is called a t-tap position of the NLFSR, $f_d(x) = \mathrm{Tr}(P(x^d))$, $P(x)$ is a nonlinear permutation over $\mathbb{F}_{2^t}$, and $\oplus$ is

the addition over $\mathbb{F}_2$. The recurrence relation is depicted in Figure 4.1. For a proper selection of a t-tap position and a feedback function $f_d(x)$, the binary sequence $\mathbf{a}$ can be a span n sequence. We note that for any choice of a t-tap position and a feedback function $f_d(x)$, the binary sequence may not be a span n sequence. The reason for choosing $t \leq (n-1)$ is to employ a small number of internal state variables in the feedback functions for an efficient implementation of an NLFSR as well as the production of more feedback functions.

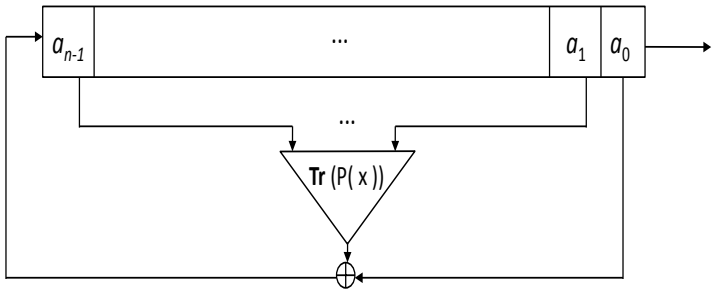

Figure 4.1: Span n sequence generation by the structured search

Let $\mathbf{b} = \{b_i\}$ be a binary sequence generated by the following recurrence relation

$$b_{n+k} = 1 \oplus b_k \oplus f_d(x_k) = 1 \oplus b_k \oplus \mathrm{Tr}(P(x_k^d)),\ x_k = (b_{r_1+k}, \ldots, b_{r_t+k}) \in \mathbb{F}_{2^t},$$
$$d \in D_t^*,\ k \geq 0. \tag{4.2}$$

Similarly, for a proper selection of a t-tap position and a feedback function $f_d(x)$, the complementary binary sequence $\bar{\mathbf{b}}$ of $\mathbf{b}$ can be a span n sequence, but the sequence $\mathbf{b}$ is not a span n sequence since it contains the all-zero state.

If the number of terms in the algebraic normal form representation of the function f_d is even, then the recurrence relation (4.1) cannot generate a span n sequence for any choice of a t-tap position, since for the all-one state the recurrence relation generates the all-one sequence. When the number of terms in f_d is even, the recurrence relation (4.2) cannot generate a span n sequence, as the complement of sequence $\{b_i\}$ will contain the all-zero tuple.

Proposition 4.2.1 *If* $\mathrm{Tr}(P(x^d)) = 0$ *for* $x = (1, 1, ..., 1) \in \mathbb{F}_{2^t}$*, then recurrence relations (4.1) and (4.2) cannot generate span n sequences.*

Varying three parameters, namely the primitive polynomial $p(x)$, the decimation number d, and the t-tap position $(r_1, r_2, ..., r_t)$ in recurrence relations (4.1) and (4.2), a number of new span n sequences can be produced and that number mainly depends on the length n of the NLFSR and the number t of inputs to the function f_d. We call this searching technique a *structured search*, where an NLFSR has a compact representation in terms of feedback functions and tap positions. Note that we may not always obtain a span n sequence for a fixed value of t and for any length n of the NLFSR. A special case of recurrence relation (4.1) with the trace function in $(n-1)$ variables as the feedback function is defined in [95].

A periodic reverse binary sequence is defined as follows [81, 82]: For a binary sequence $\{a_0, a_1, ..., a_{2^n-2}\}$ with period $2^n - 1$, the reverse sequence of the binary sequence is defined by $\{a_{2^n-2}, a_{2^n-3}, ..., a_1, a_0\}$. A reverse sequence of a span n sequence is also a span n sequence, which is not shift equivalent to the original one and the reverse span n sequence can be generated by the same function but with a different t-tap position.

Proposition 4.2.2 *[81] Let* $g(x_0, x_1, \ldots, x_{n-1}) = x_0 \oplus f(x_1, \ldots, x_{n-1})$ *generates a span n sequence with period* $2^n - 1$*. Then the function* $h(x_0, x_{n-1}, \ldots, x_1) = x_0 \oplus f(x_{n-1}, \ldots, x_1)$ *generates a reverse span n sequence.*

Our span n sequences generated by recurrence relations (4.1) and (4.2) with a permutation are uniquely determined by the following three parameters:

1. the decimation number d,
2. the primitive polynomial $p(x)$, and
3. the t-tap position $(r_1, r_2, ..., r_t)$.

Similarly, the reverse span n sequence of a span n sequence with parameters d, $p(x)$, and $(r_1, r_2, ..., r_t)$ is represented by the same decimation number d and the same primitive polynomial $p(x)$, but with a different t-tap position $(n-r_1, n-r_2, ..., n-r_t)$.

For a fixed function $f_d(x)$, a span n sequence generated by $f_d(x)$ is different if the t-tap position is different.

Using Proposition 3.2.2 and recurrence relations (4.1) and (4.2), we form the following recurrence relations that can generate de Bruijn sequences.

$$s_{i+n} = s_0 \oplus \mathrm{Tr}(P(x_i^d)) \oplus \prod_{i=1}^{n-1}(s_i \oplus 1), x_i = (s_{r_1+i}, s_{r_2+i}, ..., s_{r_t+i}), d \in D \tag{4.3}$$

$$z_{i+n} = 1 \oplus z_0 \oplus \mathrm{Tr}(P(y_i^d)) \oplus \prod_{i=1}^{n-1} z_i, y_i = (z_{r_1+i}, z_{r_2+i}, ..., z_{r_t+i}), d \in D \tag{4.4}$$

We note that sequence $\{s_i\}$ is a de Bruijn sequence when recurrence relation (4.1) generates a span n sequence and the complementary sequence of $\{z_i\}$ is a de Bruijn sequence when recurrence relation (4.2) generates a complementary span n sequence.

Example 3 The following example describes our span n sequence generation procedure for $t = 5$ when the permutation P is the WG permutation. The WG transformation over $\mathbb{F}_{2^5}$ is given by

$$\begin{aligned} f(x) &= \mathrm{Tr}(x + (x+1)^5 + (x+1)^{13} + (x+1)^{19} + (x+1)^{21}),\ x \in \mathbb{F}_{2^5} \\ &= \mathrm{Tr}(x^{19}),\ \text{after simplification.} \end{aligned}$$

For $t = 5$, the set of coset leaders for which $f_d(x)$ is nonlinear is given by $D_t^* = \{1, 3, 7, 11, 15\}$. The d-th decimation of $f(x)$ is given by

$$f_d(x) = f(x^d) = \mathrm{Tr}(x^{d'}),\ d' = (19 \cdot d) \bmod 2^t - 1,\ d \in D_t^*.$$

The n-stage nonlinear recurrence relation with a t-tap position is given by

$$a_{n+k} = a_k \oplus f_d(x_k), x_k = (a_{r_1+k}, \ldots, a_{r_5+k}) \in \mathbb{F}_{2^5},\ k \geq 0.$$

For $n = 7$, the span n sequences produced by recurrence relations (4.1) and (4.2) are presented in Table 4.1.

Table 4.1: Span n sequences generated using WG5 for $n = 7$

By recurrence relation (4.1)		
Decimation d	Polynomial $(c_0, c_1, c_2, c_3, c_4)$	t-tap position $(r_1, r_2, r_3, r_4, r_5)$
1	1 1 1 0 1	1 2 3 4 5
1	1 1 0 1 1	1 3 4 5 6
7	1 0 0 1 0	1 2 3 4 6
7	1 0 1 0 0	1 2 4 5 6
7	1 0 1 1 1	2 3 4 5 6
11	1 0 0 1 0	1 2 4 5 6
11	1 1 1 1 0	1 2 4 5 6
11	1 1 1 0 1	1 2 4 5 6
15	1 1 1 1 0	1 2 4 5 6
By recurrence relation (4.2)		
Decimation d	Polynomial $(c_0, c_1, c_2, c_3, c_4)$	t-tap position $(r_1, r_2, r_3, r_4, r_5)$
1	1 1 1 1 0	1 2 3 4 5
1	1 1 1 0 1	1 3 4 5 6
1	1 0 1 0 0	1 3 4 5 6
7	1 0 1 1 1	1 2 3 4 5
7	1 0 1 0 0	1 2 3 4 5
7	1 1 0 1 1	1 2 3 5 6
15	1 1 1 1 0	1 2 3 4 5

4.2.2 Approximate Number of Functions in the Search Space

Note that three parameters, namely a decimation number d, a primitive polynomial $p(x)$, and a t-tap position determine a nonlinear recurrence relation or a feedback function that may generate a span n sequence. In other words, each feedback function can be considered as a candidate span n sequence. For a fixed value of n and t, we form a search space by including all possible combinations of these three parameters. In order to find span n sequences, an exhaustive search is performed over this search space. We now determine the size of the search space or the number of candidate span n sequences in terms of n and t in the following proposition.

Proposition 4.2.3 *For any $n > t \geq 6$, the number of feedback functions in the search space of recurrence relations (4.1) and (4.2) is given by $C = \left(\frac{\phi(2^t-1)}{t}\right)^2 \binom{n-1}{t}$.*

Proof As in the recurrence relations the first position is fixed for the sequence to be periodic and any t tap positions is chosen from $n-1$ positions ($n \geq 6$) to form a t-tap position, the number of distinct t-tap positions is given by $T = \binom{n-1}{t}$. Again, the total number of nonlinear feedback functions is given by $n_p \cdot |D_t^*|$, where $n_p = \frac{\phi(2^t-1)}{t}$ is the number of t degree primitive polynomials over $\mathbb{F}_2$ and $|D_t^*|$ is the number of decimation numbers for which the feedback function is nonlinear. Hence, for fixed n and t, the number of feedback functions in the search space is

$$C = n_p \cdot |D_t^*| \cdot T = \left(\frac{\phi(2^t-1)}{t}\right)^2 \binom{n-1}{t} \text{ for } |D_t^*| = \frac{\phi(2^t-1)}{t}.$$

□

Proposition 4.2.4 *A feedback shift register defined by recurrence relations (4.1) and (4.2) produces the maximum number of span n sequences when about half the length of the shift register tap positions participate in the feedback functions.*

Proof Without loss of generality, we assume that the number of terms in a feedback function is even in order to produce a span n sequence. In the FSR, for different t-tap positions, the feedback functions are different. Thus, for a particular value of n and t and for a feedback function in t variables, the number of different feedback functions in n variables is equal to $N_{n,t} = \binom{n-1}{t}$ and $N_{n,t}$ is maximum when $t = \lceil \frac{n}{2} \rceil$ (For linear feedback functions, t is always odd and $t \approx \lceil \frac{n}{2} \rceil$). If the feedback functions in n variables that generate span n sequences are uniformly distributed over the set of all Boolean functions, then the FSR generates the maximum number of span n sequences when $t \approx \lceil \frac{n}{2} \rceil$. Hence, the assertion is established. □

We note that an LFSR also produces the maximum number of span n sequences when $t \approx \lceil \frac{n}{2} \rceil$ (see Table 4.2). This property is also satisfied by the nonlinearly generated span n sequences using recurrence relations (4.1) and (4.2) (see Tables 4.3, 4.5, 4.6 and 4.8).

We now estimate the number of feedback functions in the search space for finding the maximum number of span n sequences. Assume that we use NLFSRs defined by recurrence relations (4.1) and (4.2) for $t = \lceil \frac{n}{2} \rceil$. Let N denote the number

Table 4.2: Tap position distribution for an LFSR of length ≤ 20

# of taps	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
2	2	2	4	–	2	2	2	–	–	–	6	–	6	2	–	2
4	4	4	10	12	16	20	44	18	66	42	82	52	152	72	158	100
6	–	–	4	4	28	28	80	86	236	226	470	368	1050	718	1774	1104
8	–	–	–	–	2	10	50	36	264	338	720	812	2674	2296	6696	4522
10	–	–	–	–	–	–	–	4	60	140	450	648	2696	2910	10238	8436
12	–	–	–	–	–	–	–	–	4	12	66	156	1006	1470	6766	7000
14	–	–	–	–	–	–	–	–	–	–	6	12	122	284	1772	2460
16	–	–	–	–	–	–	–	–	–	–	–	–	–	24	190	354
18	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	22

of span n sequences (including reverse span n sequences) obtained by recurrence relations (4.1) and (4.2). Then we have the following theorem.

Theorem 4.2.5 *An approximate number of candidate span n sequences or feedback functions in recurrence relations (4.1) and (4.2) is given by* C_0*, where* $C_0 \approx \left(\frac{\phi(2^{\lceil \frac{n}{2} \rceil}-1)}{\lceil \frac{n}{2} \rceil}\right)^2 \cdot \frac{2^{n-1}}{\sqrt{\pi \cdot \frac{n-1}{2}}}$ *and* $C_0 \approx \frac{2^{2n-1}-2^{\frac{3n}{2}+1}}{\sqrt{\pi}\cdot(\lceil \frac{n}{2} \rceil)^{5/2}}$*, if* 2^t-1 *is a Mersenne prime, and the success probability of obtaining such a span n sequence is given by* $\frac{N}{C_0}$*.*

Proof We recall that an approximated number of functions in the search space is approximately

$$C = \left(\frac{\phi(2^t-1)}{t}\right)^2 \binom{n-1}{t}, \quad \text{for } |D_t^*| = \frac{\phi(2^t-1)}{t}.$$

Putting $t = \lceil \frac{n}{2} \rceil$ in the above formula, then we get

$$\begin{aligned} C_0 &= \left(\frac{\phi(2^{\lceil \frac{n}{2} \rceil}-1)}{\lceil \frac{n}{2} \rceil}\right)^2 \cdot \binom{n-1}{\lceil \frac{n}{2} \rceil} \\ &= \left(\frac{\phi(2^{\lceil \frac{n}{2} \rceil}-1)}{\lceil \frac{n}{2} \rceil}\right)^2 \cdot \binom{n-1}{\lfloor \frac{n-1}{2} \rfloor + 1}, \text{ for positive } n \\ &= \left(\frac{\phi(2^{\lceil \frac{n}{2} \rceil}-1)}{\lceil \frac{n}{2} \rceil}\right)^2 \cdot \frac{(n-\lfloor \frac{n-1}{2} \rfloor - 1)\cdot \binom{n-1}{\lfloor \frac{n-1}{2} \rfloor}}{(\lfloor \frac{n-1}{2} \rfloor + 1)}. \end{aligned}$$

By Stirling's formula

$$\binom{m}{\lfloor \frac{m}{2} \rfloor} \sim \frac{2^m}{\sqrt{\pi m/2}},$$

the above equation can be written as

$$\begin{aligned} C_0 &\sim \left(\frac{\phi(2^{\lceil \frac{n}{2} \rceil} - 1)}{\lceil \frac{n}{2} \rceil} \right)^2 \cdot \frac{\lfloor \frac{n-1}{2} \rfloor \cdot 2^{n-1}}{(\lfloor \frac{n-1}{2} \rfloor + 1) \cdot \sqrt{\pi \cdot \frac{n-1}{2}}} \\ &\sim \left(\frac{\phi(2^{\lceil \frac{n}{2} \rceil} - 1)}{\lceil \frac{n}{2} \rceil} \right)^2 \cdot \frac{2^{n-1}}{\sqrt{\pi \cdot \frac{n-1}{2}}}. \\ &\approx \frac{2^{2n-1} - 2^{\frac{3n}{2}+1}}{\sqrt{\pi} \cdot (\lceil \frac{n}{2} \rceil)^{5/2}}, \text{ if } 2^t - 1 \text{ is a Mersenne prime.} \end{aligned}$$

Thus the success probability of obtaining a span n sequence is equal to $\frac{N}{C_0}$. Hence, the result is proved. □

Note that recurrence relations (4.1) and (4.2) use only a class of Boolean functions in n variables where the total number of Boolean functions in n variables is $2^{2^{n-1}}$.

4.3 Span n Sequence Generation Using WG transformations

In this section, we report the number of new span n sequences generated using WG transformations, and show an empirical comparison of the success probability of obtaining a span n sequence using WG transformations and a random span n sequence generation method. We also present a heuristic method for searching long WG span n sequences.

4.3.1 WG Span n Sequences

WG span n sequences are obtained by putting the WG permutation in recurrence relations (4.1) and (4.2) for different t and n. The span n sequences are generated

by computer simulations. We use the WG transformations over the field $\mathbb{F}_{2^t}$ for $t = 5, 7, 8, 10$, and 11 (see, Section 3.4.1). We denote by WG-t the WG transformations over the field $\mathbb{F}_{2^t}$. Table 4.3 presents the number of new span n sequences produced by recurrence relations (4.1) and (4.2), respectively for $6 \leq n \leq 20$ (new reverse span n sequences are not taken into account). However, this method can be applied to generate long span n sequences. In Table 4.3, "$\times$" represents the recurrence relations are not defined for such values of n and t and $\sim$ represents those cases the number of span n sequences is not yet determined. We present some instances of new span n sequences in Appendix A.

Table 4.3: Number of WG span n sequences

By recurrence relation (4.1)																
		n														
t	WG-t	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
5	WG-5	0	9	7	14	8	11	17	11	13	10	3	7	7	0	1
7	WG-7	$\times$	$\times$	3	25	42	63	108	138	138	125	126	111	83	86	63
8	WG-8	$\times$	$\times$	$\times$	3	9	18	34	76	96	104	106	108	110	90	79
10	WG-10	$\times$	$\times$	$\times$	$\times$	$\times$	5	40	107	246	373	627	819	999	$\sim$	$\sim$
11	WG-11	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	31	204	574	1313	2539	4079	$\sim$	$\sim$	$\sim$
Total		0	9	10	42	59	97	230	536	1067	1925	3401	5124	–	–	–
By recurrence relation (4.2)																
		n														
t	WG-t	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
5	WG-5	1	7	7	10	16	18	10	8	4	10	2	1	3	1	0
7	WG-7	$\times$	$\times$	4	25	47	59	121	122	137	125	123	98	74	84	54
8	WG-8	$\times$	$\times$	$\times$	1	6	35	33	75	73	91	123	115	106	99	77
10	WG-10	$\times$	$\times$	$\times$	$\times$	$\times$	4	47	118	270	401	680	863	$\sim$	$\sim$	$\sim$
11	WG-11	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	33	186	576	1350	2522	4010	$\sim$	$\sim$	$\sim$
Total		1	7	11	36	69	116	244	509	1060	1977	3450	5087	–	–	–

A graphical representation of the number of new span n sequences for different WG-t is provided in Figure 4.2. According to the figure for different t, the number of span n sequences increases as n increases and it reaches the maximum for some value of n, and thereafter the number of span n sequences decreases as n increases. At a quick glance, we can observe that the number of span n sequences is maximal when n close to $2t$, which follows from the fact that the size of the search space is a multiple of a binomial coefficient (see Proposition 4.2.4). This fact reveals that

there exists a tradeoff between n and t for obtaining the maximum number of span n sequences.

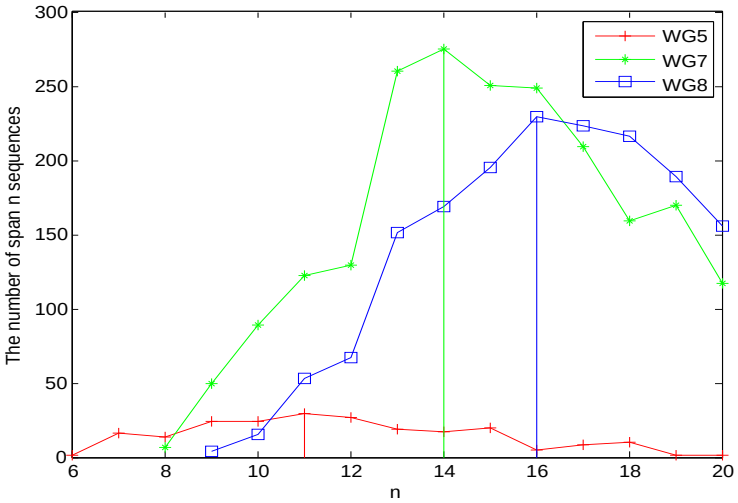

Figure 4.2: Distribution for the number of span n sequences

Remark 4.3.1 We observe that there exist many span n sequences whose t-tap positions and the bases of the finite fields are the same but their decimation numbers are different.

4.3.2 The Success Probability Comparison

Usually, in a random generation method, a span n sequence can be produced by choosing a nonlinear feedback function in n variables randomly and verifying the primitivity of the feedback function. The success probability of obtaining a randomly generated span n sequence is $\frac{1}{2^{n-3}}$ [83]. We compared the success probability of obtaining a span n sequence (including reverse sequences) in our approach with a random span n sequence generation method for $t = 5, 7, 8$ (for $t \approx \lceil \frac{n}{2} \rceil$), 10 and 11 (for $13 \leq n \leq 17$) and the comparison shows that in our approach one can produce a span n sequence with a better success probability than that of a

random span n sequence generation method. A comparison of success probability for $t = 5, 7,$ and 8 is provided in Table 4.4.

Table 4.4: The success probability comparison for WG span n sequences

	$n = 2t$	Our approach	Randomly chosen
WG5	10	$\frac{1}{2^{6.56}}$	$\frac{1}{2^7}$
WG7	14	$\frac{1}{2^{9.98}}$	$\frac{1}{2^{11}}$
WG8	16	$\frac{1}{2^{11.81}}$	$\frac{1}{2^{13}}$

4.3.3 The Search Complexity Reduction for WG Span n Sequences

It is worth noticing that the number of feedback functions in the search space increases exponentially as t increases. For large t, it will be hard to find span n sequences by considering all functions in the search space. Thus, for large n and t, a search in the search space can be performed for finding span n sequence by restricting the search over a particular type of decimation numbers and over the selections of t-tap positions. Below we list a type of decimation numbers and t-tap positions observed for WG span n sequences. In some cases, we may not find any span n sequence. However, according to our observations based on the above idea, it is possible to obtain many span n sequences.

Observations on Decimation Numbers

We have performed a search on the following type of decimation numbers for different n

$$D_{dec} = \{d : \ d \in D_t^* \text{ and } d = 2^i - 1, i = 1, 2, \ldots, t-1\}$$

for $t = 7$, 8, and 10 and the result shows that there exist many span n sequences whose decimation numbers in the recurrence relation (4.1) and (4.2) are of the above type. For this type of decimation numbers, an approximate number of feedback

functions in the search space is given by

$$C_{dec} = \frac{\phi(2^t - 1)}{t}(t-1)\binom{n-1}{t} \approx \phi(2^t - 1)\binom{n-1}{t}.$$

Obviously, the reduced complexity C_{dec} is less than the original complexity C.

Observations on t-tap Positions

Likewise, a search in the search space can be performed according to some pattern of t-tap positions for finding long period span n sequences. Assume that it is possible to fix, say, k tap positions ($1 \leq k \leq t$). Then, the total number of fixed tap positions in the recurrence relations is $(k+1)$ and we only need to choose $(t-k)$ positions out of $(n-1-k)$ positions. So, for k fixed choices of tap positions, the search complexity is

$$C_{tap} = \left(\frac{\phi(2^t - 1)}{t}\right)^2 \binom{n-1-k}{t-k}.$$

Based on our observations on the t-tap positions for $t = 7, 8,$ and 10, the following types of t-tap positions are effective when the slope of the curves in Figure 4.2 increases gradually. For example, when $t = 7$, $n = 11, 12, 13$ and 14 and $t = 8$, $n = 13, 14, 15, 16, 17$ and 18, the t-tap positions are given by: $\{1, 2, 3, 4, \cdots\}$, $\{1, 2, 3, \cdots, n-1\}$, $\{1, 2, \cdots, n-2, n-1\}$, $\{1, \cdots, n-3, n-2, n-1\}$, where the numbers in the tap positions represent fixed positions in the t-tap positions (i.e., $k = 4$ fixed positions) and "..." represents a combination of $(n-k-1)$ tap positions. We performed a search according to the first pattern of t-tap position, the following span n sequences generated by two WG transformations have been found for $t = 13$ and $n = 24$.

Decimation d	Polynomial $(c_0, c_1, c_2, ..., c_{11}, c_{12})$	t-tap position $(r_1, r_2, ..., r_{12}, r_{13})$
1207	$(1,0,1,0,1,0,1,0,1,0,1,0,0)$	$(1,2,3,4,5,6,7,10,11,12,13,15,22)$
55	$(1,1,1,0,1,1,0,0,1,1,0,1,0)$	$(1,2,3,4,5,6,9,10,11,12,13,15,17)$

4.4 Span n Sequence Generation by 3-term, 5-term, and Monomial Functions and MCM Functions

This section reports the number of span n sequences produced using three-term functions, five-term functions, monomial functions with Kasami exponents and MCM functions.

4.4.1 3-term and 5-term Span n Sequences

Considering three-term permutation and five-term permutation in recurrence relations (4.1) and (4.2), a number of span n sequences can be obtained by the structured search. Tables 4.5 - 4.6 present the number of span n sequences obtained using the recurrence relations (4.1) and (4.2) for three-term functions, five-term functions. When $t = 5$, three-term functions and five-term functions degenerate to the same functions. Furthermore, we compare the success probability of obtaining a span n sequences using 3-term and 5-term functions in Table 4.7 for $t = 5, 7, 8, 9$. Table 4.7 illustrates that a span n sequence can be produced using one of 3-term, 5-term and monomial functions with Kasami exponent with a better success probability.

4.4.2 Monomial and MCM Functions Span n Sequences

We take the monomial permutation and the MCM polynomials $f_k(x)$ (defined in Section 3.4.4) in recurrence relations (4.1) and (4.2), and produce span n sequences. We call the span n sequences produced using MCM functions *MCM span n sequences*. The MCM polynomials for different values of k over the fields $\mathbb{F}_{2^t}$, $t = 7, 9$ and 11 are considered. Tables 4.8 and 4.9 present the number of span n sequences produced using monomial functions with Kasami exponents and MCM functions, respectively for $8 \leq n \leq 20$. Some examples of span n sequences produced by monomial functions and MCM span n sequences are provided in Tables A.6 and A.7,

Table 4.5: Number of three-term span n sequences

By recurrence relation (4.1)													
		n											
t	T3-t	6	7	8	9	10	11	12	13	14	15	16	17
5	T3-5*	1	3	9	8	9	8	4	3	5	2	3	1
7	T3-7	×	×	6	25	51	89	103	150	131	128	127	123
9	T3-7	×	×	×	×	8	52	104	223	391	549	710	770
11	T3-11	×	×	×	×	×	×	35	190	624	1323	2580	4056
Total	–	1	3	15	33	68	149	246	566	1151	2002	3420	4950
By recurrence relation (4.2)													
		n											
t	T3-t	6	7	8	9	10	11	12	13	14	15	16	17
5	T3-5*	1	2	2	5	10	5	6	5	3	1	3	5
7	T3-7	×	×	4	24	44	84	98	122	133	146	128	111
9	T3-7	×	×	×	×	12	47	109	237	361	553	694	823
11	T3-11	×	×	×	×	×	×	34	186	578	1416	2554	4007
Total	–	1	3	6	29	66	136	247	550	1075	2116	3379	4946

respectively. In tables, × denotes that the recurrence relation is not defined the parameters t and n and $\sim$ denotes that the cases are incomplete due to a huge number of functions in the search space. When $t = 5$, the WG transformations and monomial functions with Kasami exponent degenerate to the same functions. Table 4.7 contains an empirical success probability comparison between a span n sequence generation using monomial functions and a random span n sequence generation method. Our empirical comparisons also show that the success probability of obtaining a span n sequence by the structured search using MCM functions is greater than that of a random span n sequence generation method. We don't provide the success probability values due to the large number of cases.

Remark 4.4.1 For 3-term, 5-term and MCM functions, the complexity of the search space is the same as the complexity of the search space for WG transformations for $t = \left\lceil \frac{n}{2} \right\rceil$. For monomial functions with Kasami exponent, the complexity of the search space can be obtained in the similar manner.

Table 4.6: Number of five-term span n sequences

By recurrence relation (4.1)															
		n													
t	FT-t	6	7	8	9	10	11	12	13	14	15	16	17	18	19
5	FT-5*	1	3	9	8	9	8	4	4	5	2	3	1	0	1
7	FT-7	×	×	5	22	44	66	118	131	115	135	124	118	99	90
8	FT-8	×	×	×	1	9	18	37	56	88	101	104	86	92	90
10	FT-10	×	×	×	×	×	9	37	116	246	411	621	797	943	∼
11	FT-11	×	×	×	×	×	×	25	171	590	1443	2618	4194	∼	∼
Total		1	3	14	31	62	101	221	478	1044	2092	3470	5196	–	–
By recurrence relation (4.2)															
		n													
t	FT-t	6	7	8	9	10	11	12	13	14	15	16	17	18	19
5	FT-5*	1	2	2	5	10	5	6	5	3	1	3	5	0	1
7	FT-7	×	×	8	19	43	74	108	138	138	127	117	102	84	91
8	FT-8	×	×	×	0	6	22	38	54	66	116	89	106	83	93
10	FT-10	×	×	×	×	×	7	47	119	223	443	627	861	∼	∼
11	FT-11	×	×	×	×	×	×	20	172	609	1397	2558	4062	∼	∼
Total		1	2	10	24	59	108	219	488	1039	2084	3394	5136	–	–

4.5 Linear Span Analysis of New Span n Sequences

In this section, we study the linear span of new span n sequences generated using WG transformations, five-term functions, three-term functions, monomial functions with Kasami exponent, and MCM functions. We note that all the nonlinear feedback functions have a trace representation. The linear span of a sequence is an important randomness property, which is considered as an upper bound on sequence unpredictability because, using only twice-linear span consecutive bits, one can certainly predict the remaining bits of the sequence by the Berlekamp-Massey algorithm [5, 79]. Sequences with optimal linear complexity are of practical interests in cryptography, since an attacker requires the whole sequence to decrypt the message in a stream cipher. There is no theoretical result on the linear span of span n sequences generated by a nonlinear feedback shift register. Other than the Berlekamp-Massey algorithm, the linear complexity of a span n sequence can be determined by computing the spectral sequence of a span n sequence. What we know about the linear complexity of span n/de Bruijn sequences is the bounds

Table 4.7: The success probability comparison for 3-term, 5-term and monomial functions span n sequences

3-term Span n sequences			
	$n \approx 2t$	Our approach	Randomly chosen
TT5	10	$\frac{1}{2^{6.89}}$	$\frac{1}{2^7}$
TT7	14	$\frac{1}{2^{10.04}}$	$\frac{1}{2^{11}}$
TT9	17	$\frac{1}{2^{13.04}}$	$\frac{1}{2^{14}}$
5-term Span n sequences			
	$n = 2t$	Our approach	Randomly chosen
FT5	10	$\frac{1}{2^{6.89}}$	$\frac{1}{2^7}$
FT7	14	$\frac{1}{2^{10.10}}$	$\frac{1}{2^{11}}$
FT8	16	$\frac{1}{2^{12.02}}$	$\frac{1}{2^{13}}$
Monomial functions with Kasami exponent Span n sequences			
	$n \approx 2t$	Our approach	Randomly chosen
MF5	10	$\frac{1}{2^{6.88}}$	$\frac{1}{2^7}$
MF7	14	$\frac{1}{2^{10.29}}$	$\frac{1}{2^{11}}$
MF9	17	$\frac{1}{2^{12.96}}$	$\frac{1}{2^{14}}$

presented in Property 1 in Chapter 3.

We compute the linear span of new span n sequences by the Berlekamp-Massey algorithm and our computational results show that the linear spans attained by new sequences are the optimal $(2^n - 2)$, the near-optimal $(2^n - 2 - 3n)$ and between the near-optimal and optimal. Table A.8 presents a summary of the linear spans of WG span n sequences generated by the recurrence relations (4.1) and (4.2). Moreover, Tables A.9, A.10, A.11, and A.12 exhibit a summary of the linear spans of the span n sequences generated by five-term functions, three-term functions, monomial functions with Kasami exponent, and MCM functions, respectively for different values of t. Our computational results also show that most of new sequences obtain the optimal linear span $(2^n - 2)$, only very few span n sequences obtain the linear span $(2^n - 2 - 3n)$ and in some cases all the linear spans are greater than $(2^n - 2 - 3n)$. We summarize the above discussions in the following two properties.

Property 2 *For all newly found span n sequences (including reverse span n se-*

Table 4.8: Number of span n sequences generated by monomial functions

By recurrence relation (4.1)															
		n													
t	MF-t	6	7	8	9	10	11	12	13	14	15	16	17	18	19
5	MF-5	0	9	7	14	8	11	17	11	13	10	3	7	7	0
7	MF-7	×	×	6	17	41	76	79	118	108	99	125	78	88	72
9	MF-9	×	×	×	×	10	43	120	258	410	519	662	788	∼	∼
11	MF-11	×	×	×	×	×	×	26	188	604	1423	2491	4056	∼	∼
Total	–	0	9	13	31	59	130	242	575	1135	2051	3281	4929	–	–
By recurrence relation (4.2)															
		n													
t	MF-t	6	7	8	9	10	11	12	13	14	15	16	17	18	19
5	MF-5	1	7	7	10	16	18	10	8	4	10	2	1	3	1
7	MF-7	×	×	4	25	45	60	98	117	114	104	116	96	86	77
9	MF-9	×	×	×	×	6	37	131	239	367	558	740	860	∼	∼
11	MF-11	×	×	×	×	×	×	32	184	596	1403	2547	4074	∼	∼
Total	–	1	7	11	35	67	115	271	548	1081	2075	3405	5031	–	–

quences), $7 \leq n \leq 20$ *and* n *is a prime number, the linear span or linear complexity of the WG, 5-term, 3-term, monomial functions with Kasami exponent, MCM span* n *sequences takes the following three values* $\{2^n - 2 - 2n, 2^n - 2 - n, 2^n - 2\}$.

Property 3 *For* $7 \leq n \leq 20$ *and all the other cases, except for those in Property 2, the linear span, denoted as* LS*, is bounded by*

$$2^n - 2 - 3n \leq LS \leq 2^n - 2$$

for all WG, 5-term, 3-term, monomial function, and MCM span n *sequences when* n *is a composite number and their respective reverse span* n *sequences for any* n*.*

4.6 Summary of Chapter 4

This chapter presented the theoretical results on span n sequences and computational results about the number of span n sequences. In the theoretical results, we described the nonlinear recurrence relations used in the structured search where

Table 4.9: Number of MCM span n sequences

By recurrence relation (4.1)														
		n												
t	k	8	9	10	11	12	13	14	15	16	17	18	19	20
7	3	3	26	53	86	113	141	147	142	126	121	110	77	57
	5	5	22	44	72	112	124	128	148	112	122	92	80	52
9	5	×	×	13	47	106	247	418	553	674	799	846	∼	∼
	7	×	×	9	32	118	238	383	577	632	779	∼	∼	∼
11	3	×	×	×	×	23	196	614	1392	2595	4200	∼	∼	∼
	5	×	×	×	×	39	188	610	1384	2560	3981	∼	∼	∼
	7	×	×	×	×	30	187	565	1374	2587	4106	∼	∼	∼
	9	×	×	×	×	48	196	615	1380	2606	4093	∼	∼	∼
By recurrence relation (4.2)														
		n												
t	k	8	9	10	11	12	13	14	15	16	17	18	19	20
7	3	5	23	51	84	106	125	115	136	122	103	107	81	59
	5	6	25	44	80	113	151	141	132	141	112	111	87	59
9	5	×	×	9	59	131	250	372	577	679	854	830	∼	∼
	7	×	×	11	45	139	245	425	543	714	786	809	∼	∼
11	3	×	×	×	×	35	174	615	1368	2493	4204	∼	∼	∼
	5	×	×	×	×	26	179	559	1357	2596	3989	∼	∼	∼
	7	×	×	×	×	22	172	585	1376	2551	4046	∼	∼	∼
	9	×	×	×	×	24	192	566	1359	2520	4212	∼	∼	∼

a feedback function of an NLFSR is composed of a decimation number, a primitive polynomial and a t-tap position. We then showed that the structured search produces the maximum number of span n sequences when half the length of FSR tap positions participate in the feedback function, and estimated the approximate number of feedback functions involved in the structured search for the above parameters.

In the computational results, we reported the number of span n sequences produced by the structured search using WG transformations, three-term functions, five-term functions, monomial functions with Kasami exponent, and MCM functions as nonlinear feedback functions. We calculated the probability of success for

obtaining a span n sequences for each case. Our empirical comparison shows that the success probability of obtaining a span n sequence in the structured search is greater than that of a random span n sequence generation method when n close to $2t$. An analysis on the linear span of new span n sequences produced by the aforementioned functions is conducted, and a summary of the bounds of the linear span for different values of t is presented. The linear span of a new span n sequence lies between the near-optimal and optimal. We observed that the majority of span n sequences have an optimal linear span. Our computational results show that the structure search can be used to find span n sequences with a moderate n.

Another aspect of studying the structured search is to find a general construction of a nonlinear feedback function that can generate a span n sequence. Unfortunately, we could not establish any such general construction of a span n sequence. The new span n sequences or span n sequences produced by the structured search can be used as building blocks in designing lightweight PRNGs and stream ciphers for securing communication systems.

Chapter 5

Strong de Bruijn Sequences with Large Periods by the Composited Construction

The concept of the composition of two feedback functions f and g, defined in Section 3.2.5, was suggested by Green and Dimond [50] in 1970 and independently by Mykkeltveit in 1976 [90]. In 1970, Lempel [67] introduced the idea of D-morphic image and preimages of a binary sequence, and presented a construction of producing de Bruijn sequences using D-morphic preimages. Later on, in 1979, Mykkeltveit *et al.* [92] widely studied the cycle structures of f composed with g and g composed with f, and presented the construction of Lempel in the form of a composited feedback function.

This chapter investigates how to generate a strong de Bruijn sequence from a span n sequence through the composition method by Lempel [67] and Mykkeltveit *et al.* [92] where the span n sequence has an optimal or near-optimal linear complexity. In Section 5.1, we refine the composited construction in which a feedback function of a long de Bruijn sequence is constructed from a feedback function of a span n sequence. Section 5.2 determines the linear complexity of a composited de Bruijn sequence, and Section 5.3 conducts an analysis of a composited nonlinear recurrence relation from a cryptographic point of view. In the analysis, we investigate an

approximation of the feedback function by setting some product terms as constant functions. We also determine the cycle structure of an approximated feedback function and the linear complexity of a sequence generated by an approximated feedback function. The analysis shows that a de Bruijn sequence generated by the composited construction is strong if the starting span n sequence is strong. In Section 5.4, we derive an algebraic representation of an $(n+16)$-stage NLFSR, and present a few instances of cryptographically strong de Bruijn sequences with periods in the range of 2^{35} and 2^{40}. We use the span n sequences with optimal or near-optimal linear complexity discovered in Chapter 4 in the composited construction. We discuss the implementation issues of a composited NLFSR for a de Bruijn sequence in Section 5.5. Finally, we summarize this chapter in Section 5.6. Partial contents of the chapter have been published in [70] and some results can be found in [71].

Table 5.1: Notations used in Chapter 5

Z_o^n:	Set of odd integers between 1 and n
Z_e^n:	Set of even integers between 1 and n
$Supp(f)$:	The support of Boolean function f
$H(f)$:	The Hamming weight of the Boolean function f
$\psi(x_0, x_1) = x_0 + x_1$:	A Boolean function to be used for composition
ψ^k:	The k-th order composition of ψ
$\Omega(g)$:	Cycle decomposition of feedback function g

5.1 Feedback Functions of Composited de Bruijn Sequences

In [92], Mykkeltveit *et al.* mentioned the idea of constructing a long stage NLFSR from a short stage NLFSR by repeatedly applying Theorem 3.2.4 when g is a linear function in two variables that generates a de Bruijn sequence. In this section, we first refine Mykkeltveit *et al.*'s idea so that we can generate long de Bruijn sequences, and then show an analytic formulation of a recursive feedback function of an $(n+k)$-

stage NLFSR, which is constructed from a feedback function of an n-stage NLFSR by repeatedly applying Theorem 3.2.4 and the composition operation.

5.1.1 The k-th Order Composition of a Boolean Function

Let $g(x_0, x_1, ..., x_n) = x_0 + x_n + G(x_1, x_2, ..., x_{n-1})$ be a Boolean function in $(n+1)$ variables, where G is a Boolean function in $(n-1)$ variables, and $g(x_0, ..., x_n) = 0$ is a nonsingular recurrence relation of n stages. The *first order* composition of g and ψ, denoted as $g \circ \psi$, is given by [92]

$$\begin{aligned} g(y_0, ..., y_n) \circ \psi(x_0, x_1) &= g(x_0 + x_1, x_1 + x_2, ..., x_n + x_{n+1}) \\ &= x_0 + x_1 + x_{n+1} + x_n + G(x_1 + x_2, ..., x_{n-1} + x_n). \end{aligned}$$

Similarly, the *k-th order* composition of g with respect to ψ is defined by

$$g \circ \psi^k = \left(g \circ \psi^{k-1}\right) \circ \psi, k \geq 2$$

where $g \circ \psi^{k-1}$ is $(k-1)$-th order composition of g with respect to ψ.

Proposition 5.1.1 *For a positive integer k, the number of distinct variables in $x_i \circ \psi^k$ is equal to 2^l where l is the Hamming weight of k.*

We can simply expand $x_i \circ \psi^k$ by first computing the power set of $\{k_1, k_2, ..., k_l\}$ with the empty is set to zero, and then summing up all the elements of each set and adding i to each sum, where $k = \sum_{j=1}^{l} k_j$, $k_j = 2^q$ for some q. For an efficient evaluation of $(g \circ \psi^k)$, the value of k must be chosen such that the Hamming weight of k is low.

Example 4 The k-th order composition $(1 \leq k \leq 16)$ of $f = x_i$ with respect to ψ is given in Table 5.2.

Table 5.2: The k-th order composition of x_i w.r.t ψ

k	$f \circ \psi^k$	k	$f \circ \psi^k$
1	$x_i + x_{i+1}$	9	$x_i + x_{i+1} + x_{i+8} + x_{i+9}$
2	$x_i + x_{i+2}$	10	$x_i + x_{i+2} + x_{i+8} + x_{i+10}$
3	$x_i + x_{i+1} + x_{i+2} + x_{i+3}$	11	$x_i + x_{i+1} + x_{i+2} + x_{i+3} + x_{i+8} + x_{i+9} + x_{i+10} + x_{i+11}$
4	$x_i + x_{i+4}$	12	$x_i + x_{i+4} + x_{i+8} + x_{i+12}$
5	$x_i + x_{i+1} + x_{i+4} + x_{i+5}$	13	$x_i + x_{i+1} + x_{i+4} + x_{i+5} + x_{i+8} + x_{i+9} + x_{i+12} + x_{i+13}$
6	$x_i + x_{i+2} + x_{i+4} + x_{i+6}$	14	$x_i + x_{i+2} + x_{i+4} + x_{i+6} + x_{i+8} + x_{i+10} + x_{i+12} + x_{i+14}$
7	$\sum_{l=0}^{7} x_{i+l}$	15	$\sum_{l=0}^{15} x_{i+l}$
8	$x_i + x_{i+8}$	16	$x_i + x_{i+16}$

5.1.2 Repeated Compositions of a Product Term

Let X_0^p be a product term in p variables which is given by

$$X_0^p = \prod_{i \in Z_o^p} x_i \prod_{i \in Z_e^p} (x_i + 1).$$

The first order composition of X_0^p with respect to ψ, denoted as X_1^p, is given by

$$X_1^p = \prod_{i \in Z_o^p} (x_i + x_{i+1}) \prod_{i \in Z_e^p} (x_i + x_{i+1} + 1)$$

which is a *product-of-sum term* or *composed term* in $(p+1)$ variables. Similarly, the k-th order composition of X_0^p with respect to ψ, denoted by X_k^p, is defined as

$$X_k^p = (X_{k-1}^p) \circ \psi, k \geq 2$$

which is a product-of-sum term in $(p+k)$ variables. Note that the composition operation with respect to ψ increases the number of variables in X_0^p by one when it repeats once, but the composition operation does not increase the algebraic degree of X_0^p.

We denote by $J^{n-1} = \prod_{i=1}^{n-1}(x_i + 1)$. In a similar manner, the k-th order compo-

sition of J^{n-1} with respect to ψ, denoted as J_k^{n-1}, is defined by $J_k^{n-1} = \left(J_{k-1}^{n-1}\right) \circ \psi$, where J_{k-1}^{n-1} is the $(k-1)$-th order composition of J^{n-1}.

Let us now define a Boolean function I_k^n in $(n+k-1)$ variables as

$$I_k^n(x_1, x_2, ..., x_{n+k-1}) = J_k^{n-1} + X_{k-1}^n + X_{k-2}^{n+1} + \cdots + X_1^{n+k-2} + X_0^{n+k-1}$$

which is a sum of $(k+1)$ product-of-sum terms and the algebraic degree of I_k^n is maximum and equals $(n+k-1)$. Function I_k^n can also be written in terms of the composition operation as follows

$$I_{k+1}^n = I_k^n \circ \psi + X_0^{n+k}, \text{ for } k \geq 0 \text{ and } n \geq 2,$$

where $I_0^n = J^{n-1}$.

5.1.3 The Composited Construction of a de Bruijn Sequence

We now present the construction of an $(n+k)$-stage NLFSR that is constructed from an n-stage NLFSR.

Proposition 5.1.2 *Let $g(x_0, x_1, ..., x_n) = x_n + x_0 + G(x_1, x_2, ..., x_{n-1})$, which generates a span n sequence of period $(2^n - 1)$, where G is a Boolean function in $(n-1)$ variables. Then, for any integer $k \geq 1$, $R_k^n(x_0, x_1, ..., x_{n+k}) = (x_n + x_0) \circ \psi^k + G(x_1, x_2, ..., x_{n-1}) \circ \psi^k + I_k^n(x_1, ..., x_{n+k-1})$ generates a de Bruijn sequence of period 2^{n+k}.*

Proof The feedback function $(g + J^{n-1}) = 0$ generates a de Bruijn sequence of period 2^n. By applying Theorem 3.2.4 to the feedback function $(g + J^{n-1})$ k times, the feedback function becomes

$$\begin{aligned} R_k^n(x_0, x_1, ..., x_{n+k}) &= (x_n + x_0) \circ \psi^k + G(x_1, x_2, ..., x_{n-1}) \circ \psi^k + \\ &\quad I_k^n(x_1, ..., x_{n+k-1}), k \geq 0 \qquad (5.1) \\ &= (x_n + x_0) \circ \psi^k + G(x_1 \circ \psi^k, ..., x_{n-1} \circ \psi^k) + \\ &\quad I_k^n(x_1, x_2, ..., x_{n+k-1}). \qquad (5.2) \end{aligned}$$

The function $R_k^n = 0$ is a feedback function in $(n+k)$ variables of an NLFSR, and generates a de Bruijn sequence with period 2^{n+k}. □

Definition 16 *A de Bruijn sequence of period 2^{n+k} produced by recurrence relation (5.1) is referred to as a* composited de Bruijn sequence.

Definition 17 *The recurrence relation (5.1) is called a* composited recurrence relation*, and the NLFSR for recurrence relation (5.1) is referred to as a* composited NLFSR.

One can construct the feedback function R_{k+1}^n from R_k^n in the following recursive manner

$$R_{k+1}^n = R_k^n \circ \psi + X_0^{n+k} \text{ or } R_{k+1}^n = g \circ \psi^{k+1} + I_{k+1}^n, k \geq 0$$

where $R_0^n = (g + J^{n-1})$.

Remark 5.1.3 For $k = 1$, Proposition 5.1.2 is the same as Theorem 3.2.4 which is also found by Lempel in [67]. For $k = 1$ and g is a primitive polynomial, Proposition 5.1.2 is similar to Theorem 2 in [92].

Remark 5.1.4 According to Theorem 3.2.4, the product term X_0^p in the recurrence relation (5.1) can be replaced by the product term $\prod_{i \in Z_o^p}(x_i + 1) \prod_{i \in Z_e^p} x_i$.

5.1.4 Algebraic Form of I_{16}^n

We now present an algebraic form of I_{16}^n for a recurrence relation of $(n+16)$ stages, and the algebraic form is derived by putting $k = 16$ in the recurrence relation (5.1). Then, the nonlinear recurrence relation of $(n + 16)$ stages is given by

$$\begin{aligned} R_{16}^n(x_0, ..., x_{n+16}) = {} & x_{n+16} + x_n + x_0 + x_{16} + G(x_1 + x_{17}, ..., x_{n-1} + x_{n+15}) \\ & + J_{16}^{n-1} + X_{15}^n + \cdots + X_1^{n+14} + X_0^{n+15} = 0 \end{aligned} \tag{5.3}$$

where $J_{16}^{n-1} = \prod_{i=1}^{n-1}(x_i + x_{i+16} + 1)$ and $X_j^i = T_{o,j}^i \cdot T_{e,j}^i$, $n \leq i \leq n+15, 15 \geq j \geq 0$, $T_{o,j}^i$ and $T_{e,j}^i$ are given in Table 5.3. In the product-of-sum terms, the subscripts o and e represent the odd indices product terms and even indices product terms,

respectively. Each product-of-sum term $X_j^i,\ n \le i \le n+15, 15 \ge j \ge 0$, is a function of $(n+15)$ variables. The expansion of $(x_i \circ \psi^k)$ can be found in Table 5.2 for $1 \le k \le 16$.

Table 5.3: Product-of-sum terms in I_{16}^n of the recurrence relation (5.3)

$T_{o,15}^n = \prod_{i\in Z_o^n} (\sum_{l=0}^{15} x_{i+l})$	$T_{o,14}^{n+1} = \prod_{i\in Z_o^{n+1}} (\sum_{l=0}^{7} x_{i+2l})$
$T_{o,13}^{n+2} = \prod_{i\in Z_o^{n+2}} (x_i + x_{i+1} + \sum_{l=1}^{3} (x_{i+2^l} + x_{i+2^l+1}))$	$T_{o,12}^{n+3} = \prod_{i\in Z_o^{n+3}} (\sum_{l=0}^{3} x_{i+4l})$
$T_{o,11}^{n+4} = \prod_{i\in Z_o^{n+4}} (\sum_{l=0}^{4} x_{i+l} + \sum_{l=8}^{11} x_{i+l})$	$T_{o,10}^{n+5} = \prod_{i\in Z_o^{n+5}} (x_i + x_{i+2} + x_{i+8} + x_{i+10})$
$T_{o,9}^{n+6} = \prod_{i\in Z_o^{n+6}} (x_i + x_{i+1} + x_{i+8} + x_{i+9})$	$T_{o,8}^{n+7} = \prod_{i\in Z_o^{n+7}} (x_i + x_{i+8})$
$T_{o,7}^{n+8} = \prod_{i\in Z_o^{n+8}} (\sum_{l=0}^{7} x_{i+l})$	$T_{o,6}^{n+9} = \prod_{i\in Z_o^{n+9}} (\sum_{l=0}^{3} x_{i+2l})$
$T_{o,5}^{n+10} = \prod_{i\in Z_o^{n+10}} (x_i + x_{i+1} + x_{i+4} + x_{i+5})$	$T_{o,4}^{n+11} = \prod_{i\in Z_o^{n+11}} (x_i + x_{i+4})$
$T_{o,3}^{n+12} = \prod_{i\in Z_o^{n+12}} (\sum_{l=0}^{3} x_{i+l})$	$T_{o,2}^{n+13} = \prod_{i\in Z_o^{n+13}} (x_i + x_{i+2})$
$T_{o,1}^{n+14} = \prod_{i\in Z_o^{n+14}} (x_i + x_{i+1})$	$T_{o,0}^{n+15} = \prod_{i\in Z_o^{n+16}} x_i$
$T_{e,15}^{n} = \prod_{i\in Z_e^{n}} (\sum_{l=0}^{15} x_{i+l} + 1)$	$T_{e,14}^{n+1} = \prod_{i\in Z_e^{n+1}} (\sum_{l=0}^{7} x_{i+2l} + 1)$
$T_{e,13}^{n+2} = \prod_{i\in Z_e^{n+2}} (x_i + x_{i+1} + \sum_{l=1}^{3} (x_{i+2^l} + x_{i+2^l+1}) + 1)$	$T_{e,12}^{n+3} = \prod_{i\in Z_e^{n+3}} (\sum_{l=0}^{3} x_{i+4l} + 1)$
$T_{e,11}^{n+4} = \prod_{i\in Z_e^{n+4}} (\sum_{l=0}^{4} x_{i+l} + \sum_{l=8}^{11} x_{i+l} + 1)$	$T_{e,10}^{n+5} = \prod_{i\in Z_e^{n+5}} (x_i + x_{i+2} + x_{i+8} + x_{i+10} + 1)$
$T_{e,9}^{n+6} = \prod_{i\in Z_e^{n+6}} (x_i + x_{i+1} + x_{i+8} + x_{i+9} + 1)$	$T_{e,8}^{n+7} = \prod_{i\in Z_e^{n+7}} (x_i + x_{i+8} + 1)$
$T_{e,7}^{n+8} = \prod_{i\in Z_e^{n+8}} (\sum_{l=0}^{7} x_{i+l} + 1)$	$T_{e,6}^{n+9} = \prod_{i\in Z_e^{n+9}} (\sum_{l=0}^{3} x_{i+2l} + 1)$
$T_{e,5}^{n+10} = \prod_{i\in Z_e^{n+10}} (x_i + x_{i+1} + x_{i+4} + x_{i+5} + 1)$	$T_{e,4}^{n+11} = \prod_{i\in Z_e^{n+11}} (x_i + x_{i+4} + 1)$
$T_{e,3}^{n+12} = \prod_{i\in Z_e^{n+12}} (\sum_{l=0}^{3} x_{i+l} + 1)$	$T_{e,2}^{n+13} = \prod_{i\in Z_e^{n+13}} (x_i + x_{i+2} + 1)$
$T_{e,1}^{n+14} = \prod_{i\in Z_e^{n+14}} (x_i + x_{i+1} + 1)$	$T_{e,0}^{n+15} = \prod_{i\in Z_e^{n+16}} (x_i + 1)$

5.2 Linear Complexity of Composited de Bruijn Sequences

This section determines the linear complexity of a composited de Bruijn sequence produced by a composited nonlinear recurrence relation in which the linear complexity of the starting span n sequence is known. In this chapter, we use the de Bruijn sequence of order n and the de Bruijn sequence of period 2^n interchangeably.

5.2.1 A Closer Look at the Composited Construction

Let $\mathbf{s}$ be a de Bruijn sequence of order $(n+1)$ produced by the recurrence relation (5.1) when $k = 1$. The composited construction of a de Bruijn sequence of order $(n+k)$ or the recurrence relation (5.1) can be interpreted as follows. Let

$\mathbf{a} = \{a_i\}_{i\geq 0}$ be a de Bruijn sequence of order n that is generated by $h = g + J^{n-1}$. Then the D-morphic preimages of $\mathbf{a}$ are z and $\bar{z}$, given in Section 3.2.5. According to the recurrence relation (5.1) for $k = 1$, the sequence $\mathbf{s}$ can be written as $\mathbf{s} = z\|E^t(\bar{z})$, where $z = z'\|e$, $\bar{z} = z''\|\hat{e}$, $0 \leq t \leq 2^n - 1$, E is the left shift operator, and $\|$ denotes the concatenation operation [14].

We denote by $\mathbf{s}_i = z_i\|E^{t_i}(\bar{z}_i)$ the de Bruijn sequence of order $(n + i)$ for $0 \leq t_i \leq 2^{n+i-1} - 1$ and $\mathbf{s}_0 = \mathbf{a}$. A de Bruijn sequence of order $(n + k)$ is constructed recursively by calculating preimages as follows:

$$\begin{aligned}
&\mathbf{s}_1 = z_1\|E^{t_1}(\bar{z}_1) \text{ for } 0 \leq t_1 \leq 2^n - 1\\
&\mathbf{s}_2 = z_2\|E^{t_2}(\bar{z}_2) \text{ for } 0 \leq t_2 \leq 2^{n+1} - 1\\
&\quad\vdots \qquad\quad \vdots\\
&\mathbf{s}_k = z_k\|E^{t_k}(\bar{z}_k) \text{ for } 0 \leq t_k \leq 2^{n+k-1} - 1
\end{aligned}$$

where z_i and $\bar{z}_i$ are D-morphic preimages of the de Bruijn sequence $\mathbf{s}_{i-1}$. This is an equivalence between the recurrence relation (5.1) and the construction of a de Bruijn sequence of order $(n + k)$ from a de Bruijn sequence of order n when the concatenation is performed at the conjugate pair $e_i = (1, 1, 0, 1, 0, ..., 1, 0) \in \mathbb{F}_{2^i}$ and $\hat{e}_i = (0, 1, 0, 1, 0, ..., 1, 0) \in \mathbb{F}_{2^i}$, $n \leq i \leq n + k - 1$.

5.2.2 Linear Complexity of a Composited de Bruijn Sequence

We now determine the linear complexity of a de Bruijn sequence produced by recurrence relation (5.1) in terms of the linear complexity of the starting span n sequence generated by g. We use the notations of Section 5.2.1 in the following theorem.

Theorem 5.2.1 *Let the linear complexity of a span n sequence generated by g be optimal, i.e,* $2^n - 2$. *Then the linear complexity of a de Bruijn sequence* $\boldsymbol{s}_k$ *of period* 2^{n+k} *generated by recurrence relation (5.1), denoted as* $LC(\boldsymbol{s}_k)$, *is bounded below by* $(2^{n+k} - 2 - \sum_{i=1}^{k} 2^{m_i})$ *where* $2^{m_i} \mid t_i$ *but* $2^{m_i+1} \nmid t_i$, $1 \leq i \leq k$.

Proof For $k = 1$, the de Bruijn sequence $\mathbf{s}_1$ can be written as $\mathbf{s}_1 = z_1 \| E^{t_1}(\bar{z}_1)$ for $0 \leq t_1 \leq 2^n - 1$, where $z_1 = z_1' \| e$, $\bar{z}_1 = z_1'' \| \hat{e}$. By Theorem 11 of [14],

$$LC(\mathbf{s}_1) \geq 2^n + 2^n - 2 - 2^{m_1} = 2^{n+1} - 2 - 2^{m_1}$$

where $2^{m_1} \mid t_1$ but $2^{m_1+1} \nmid t_1$ as $LC(\mathbf{s}_0)$ is greater than or equal to the linear complexity of the starting span n sequence generated by g. As de Bruijn sequence $\mathbf{s}_2$ is constructed from $\mathbf{s}_1$ in the same way, applying the same argument, the linear complexity of sequence $\mathbf{s}_2$ is

$$LC(\mathbf{s}_2) \geq 2^{n+1} + 2^{n+1} - 2 - 2^{m_1} - 2^{m_2} = 2^{n+2} - 2 - 2^{m_1} - 2^{m_2}$$

where $2^{m_2} \mid t_2$ but $2^{m_2+1} \nmid t_2$. In general, for $k \geq 1$, the linear complexity bound of $\mathbf{s}_k$ is

$$LC(\mathbf{s}_k) \geq 2^{n+k} - 2 - \sum_{i=1}^{k} 2^{m_i}$$

where $2^{m_i} \mid t_i$ but $2^{m_i+1} \nmid t_i$, $1 \leq i \leq k$. □

Since the exact linear complexity of a composited de Bruijn sequence depends on the values of m_i's, we computed the linear complexity of many composited de Bruijn sequences when the starting span n sequences generated by g have optimal or near-optimal linear complexity for $(n + k) = 11, 12, ...,$ and 20 and for different values of k and n. Our experimental result shows that the linear complexities of composited de Bruijn sequences are optimal or close to optimal, both of which are much greater than the lower bound $(2^{n+k-1} + n + k)$.

Remark 5.2.2 In Theorem 5.2.1, the inequality is due to no knowing the exact linear complexity of the de Bruijn sequence obtained from the span n sequence.

Remark 5.2.3 If L $(\geq 2^{n-1} + 2)$ is the linear complexity of a span n sequence generated by g, then the linear complexity of $\mathbf{s}_k$ satisfies $LC(\mathbf{s}_k) \geq L + 2^n(2^k - 1) - \sum_{i=1}^{k} 2^{m_i}$.

5.3 Cryptanalysis of a Composited NLFSR for a de Bruijn Sequence

Since the function I_k^n contains $(k+1)$ product-of-sum terms whose algebraic degrees are high and the Hamming weights of these product-of-sum terms are low, as a result, the function I_k^n can be approximated by a linear function or a constant function with high probability. In this section, we first investigate the success probability of approximating the function I_k^n by the zero function. We then study the cycle decomposition of an approximated recurrence relation after a successful approximation of the feedback function.

5.3.1 Hamming Weights of the Product-Of-Sum Terms

Before calculating the success probability of approximating the function I_k^n by the zero function, we need to derive the Hamming weight of a product-of-sum term, since I_k^n is a sum of $(k+1)$ product-of-sum terms.

Proposition 5.3.1 *For an integer $r \geq 1$, the Hamming weight of X_r^p is equal to 2^r.*

Proof For any product term X_0^p, the r-order composition is of the form

$$X_r^p = \prod_{i \in Z_o^p} U_i \cdot \prod_{i \in Z_e^p} V_i$$

where U_i is a sum of 2^c variables and V_i is a sum of 2^c variables and constant 1, c is the Hamming weight of r. For simplicity, we assume that $r = 2^l, l \geq 0$. To find the Hamming weight of X_r^p, there are two cases arise.
Case I: When $1 \leq p \leq r+1$
If $r = 2^l$, then U_i and V_j can be written as $U_i = x_i + x_{i+r},\ i \in Z_o^p,\ V_j = (x_j + x_{j+r} + 1),\ j \in Z_e^p$, respectively. $X_r^p = 1$ if and only if $U_i = 1$ and $V_j = 1$ for all

$i \in Z_o^p$ and $j \in Z_e^p$. This implies

$$\begin{aligned}
&x_1 = 1 + x_{1+r} = 1 + x_{1+2r} = \cdots = 1 + x_{l_1} = 0/1 \\
&x_2 = x_{2+r} = x_{2+2r} = \cdots = x_{l_2} = 0/1 \\
&\quad \vdots \quad \vdots \\
&x_p = 1 + x_{p+r} = 1 + x_{p+2r} = \cdots = 1 + x_{l_n} = 0/1, \text{ if } p \text{ is odd} \\
&x_p = x_{p+r} = x_{p+2r} = \cdots = x_{l_p} = 0/1, \text{ if } p \text{ is even}
\end{aligned}$$

where $l_i \leq p + r,\ i = 1, 2, ..., p$. Note that X_r^p is a function in $(p + r)$ variables. For an $(p + r)$-tuple with $X_r^p = 1$, the values at $2p$ positions are determined by the values at p positions, which follows from the above set of equations and the remaining $(p+r-2p)$ positions can take any binary values. Hence, the total number of $(p + r)$-tuples for which $X_r^p = 1$ is given by $2^p \cdot 2^{r-p} = 2^r$.

Case II: When $p \geq r + 1$

Similarly, $X_r^p = 1$ if and only if $U_i = 1$ and $V_j = 1$ for all $i \in Z_o^p$ and $j \in Z_e^p$. This implies

$$\begin{aligned}
&x_1 = 1 + x_{1+r} = 1 + x_{1+2r} = \cdots = 1 + x_{l_1} = 0/1 \\
&x_2 = x_{2+r} = x_{2+2r} = \cdots = x_{l_2} = 0/1 \\
&\quad \vdots \quad \vdots \\
&x_{r-1} = 1 + x_{2r-1} = \cdots = 1 + x_{l_{r-1}} = 0/1 \\
&x_r = x_{2r} = \cdots = x_{l_r} = 0/1
\end{aligned}$$

where $l_i \leq (p + r),\ i = 1, 2, ..., r$. According to the above system of equations, the binary values at $(p+r)$ positions are determined by the binary values at r positions and these r positions can take any values. Hence, the total number of $(p+r)$-tuples for which $X_r^p = 1$ is given by 2^r.

Considering $U_i = 1$ and $V_j = 1$ for all $i \in Z_o^p$ and $j \in Z_e^p$ as a system of linear equations with p equations and $(p + r)$ unknown variables over $\mathbb{F}_2$, it follows that

the Hamming weight of X_r^p is equal to the number of solutions of the system of linear equations, which is equal to $2^{p+r-r} = 2^r$ for any positive integer r. □

Proposition 5.3.2 *For any integer $r \geq 1$, the Hamming weight of J_r^{n-1} is equal to 2^r.*

Proof The proof is similar to the proof of Proposition 5.3.1. □

Proposition 5.3.3 *For any integer $k \geq 1$ and $n \geq 2$, the Hamming weight of function I_k^n is equal to 2^k+1. One can approximate function I_k^n by the zero function with probability $(1 - \frac{1}{2^{n-1}} - \frac{1}{2^{n+k-1}})$.*

Proof By Proposition 5.3.1, the Hamming weight of $X_j^{n+k-1-j}$, i.e, $H(X_j^{n+k-1-j})$ is equal to 2^j, for $0 \leq j \leq k-1$. Note that $X_j^{n+k-1-j} = 1$ is a system of linear equations with $(n+k-1-j)$ equations and $(n+k-1)$ unknown variables and $Supp(X_j^{n+k-1-j})$ contains the set of all solutions. It is not hard to show that the support of $X_i^{n+k-1-i}$ and $X_j^{n+k-1-j}$ are disjoint for $0 \leq i \neq j \leq n-1$. Again, $(\cup_{j=0}^{k-2} Supp(X_j^{n+k-1-j})) \subset Supp(J_k^{n-1})$, and $Supp(X_{k-1}^{n+k-1})$ and $Supp(J_k^{n-1})$ are disjoint. Then the cardinality of the support of I_k^n is equal to $(2^k+2^{k-1}-\sum_{j=0}^{k-2} 2^j) = (2^k+2^{k-1}-2^{k-1}+1) = 2^k+1$. Hence, the Hamming weight of I_k^n is 2^k+1.

Since the Hamming weight of I_k^n is 2^k+1, the number of inputs for which I_k^n takes the value zero is equal to $(2^{n+k-1} - 2^k - 1)$. Hence, one can approximate the function I_k^n by the zero function with probability $(1 - \frac{1}{2^{n-1}} - \frac{1}{2^{n+k-1}})$. □

Proposition 5.3.4 *For any $n, k \geq 1$, the nonlinearity of function I_k^n is equal to $N_{I_k^n} = H(I_k^n) = 2^k + 1$ for $n > 3$.*

Proof It is well-known that the nonlinearity of a Boolean function can be obtained by calculating the minimum distance between the function and all affine functions [23]. In Proposition 5.3.3, we calculated the cardinality of $Supp(I_k^n)$, which is equal to (2^k+1). The minimum distance between I_k^n and all affine functions is achieved only for the zero function, and that is equal to (2^k+1). For all other nonzero functions, the least distance between I_k^n and an affine function can be $(2^{n+k-2} - (2^k+1))$. For $n > 3$, $(2^{n+k-2} - (2^k+1)) > (2^k+1)$. Therefore, for $n > 3$, the nonlinearity of I_k^n equals the Hamming weight of I_k^n. □

For a small value of k, the function I_k^n can be approximated by the zero function or a linear function due to its low nonlinearity.

5.3.2 Cycle Structure of an Approximated Recurrence Relation

By Propositions 5.3.3 and 5.3.4, the function I_k^n can be approximated by the *zero function* with probability about $(1 - \frac{1}{2^{n-1}})$. As a consequence, Eq. (5.1) can be approximated as follows

$$\begin{aligned} R_{k,a}^n(x_0, x_1, ..., x_{n+k}) &= ((x_n + x_0) + G(x_1, x_2, ..., x_{n-1})) \circ \psi^k \\ &= g(x_0, x_2, ..., x_{n-1}) \circ \psi^k. \end{aligned} \tag{5.4}$$

The recurrence relation $R_{k,a}^n = 0$ is called an *approximated recurrence relation*. In the following proposition, we provide the cycle structure of an approximated recurrence relation.

Lemma 5.3.5 *For an integer $k \geq 1$, $\Omega(R_{k,a}^n) = \Omega(g) \oplus \Omega(\psi^k)$, i.e., any sequence $x \in \Omega(R_{k,a}^n)$ can be written as $x = b + c$, where b's minimal polynomial is the same as the minimal polynomial of a span n sequence that is generated by g and c's minimal polynomial is $(1 + x)^k$ and $\oplus$ denotes the direct sum operation.*

Proof Let $\mathbf{s}$ be a span n sequence generated by g and let $h(x)$ the minimal polynomial of $\mathbf{s}$. Then, $h(x) = h_1(x) \cdot h_2(x) \cdots h_r(x)$, where h_i's are distinct irreducible polynomials of degree less than or equal to n and the value of r depends on the sequence $\mathbf{s}$, see [45, 48, 82]. If $h_i(x) = (1 + x)$ for some i, then the sequence $\mathbf{s}$ is not a span n sequence. On the other hand, the minimal polynomial of ψ^k is $(1 + x)^k$. Again, the minimal polynomial of a sequence generated by ψ^k is a factor of $(1 + x)^k$. As $h(x)$ does not contain the factor $(1 + x)$, the minimal polynomial of $\mathbf{s}$ and the minimal polynomial of ψ^k are relatively prime with each other. Then, by Lemma 3.2.3, any sequence $x \in \Omega(R_{k,a}^n)$ can be represented by $x = b + c$ where $b \in \Omega(g)$ and $c \in \Omega(\psi^k)$. Hence, the cycle decomposition of $R_{k,a}^n$ is a direct sum of $\Omega(g)$ and $\Omega(\psi^k)$, i.e., $\Omega(R_{k,a}^n) = \Omega(g) \oplus \Omega(\psi^k)$. □

Proposition 5.3.6 *The cycle decomposition of $R^n_{k,a}$, i.e., $\Omega(R^n_{k,a})$ contains $2 \cdot (\Gamma_2(k)+1)$ cycles with $(\Gamma_2(k)+1)$ cycles of period at least (2^n-1) and $(\Gamma_2(k)+1)$ cycles of period at most $2^{\lceil \log_2 k \rceil}$, where $\Gamma_2(k)$ is the number of all coset leaders modulo (2^k-1).*

Proof For any positive integer $k \geq 1$, the cycle decomposition of ψ^k is the cycle decomposition of polynomial $(1+x)^k$, which contains sequences with period $2^{\lceil \log_2 i \rceil}$, $1 \leq i \leq k$, and the number of cycles is given by $(\Gamma_2(k)+1)$ including the zero cycle (see [43], Th. 3.4, page-42). Again, the cycle decomposition of g contains only two cycles, one is a cycle of length 2^n-1 and the other one is the zero cycle of length one. Therefore, by Lemma 5.3.5, $\Omega(R^n_{k,a})$ contains $2 \cdot (\Gamma_2(k)+1)$ cycles where $(\Gamma_2(k)+1)$ cycles are of length at least 2^n-1 and $(\Gamma_2(k)+1)$ cycles are of length at most $2^{\lceil \log_2 k \rceil}$. □

Remark 5.3.7 If the function R^n_k is approximated by the function $(R^n_{k,a}+J^{n-1}_k)$ with high probability, then the number of cycles in $\Omega(R^n_{k,a}+J^{n-1}_k)$ equals $(\Gamma_2(k)+1)$, and the period of a sequence in $\Omega(R^n_{k,a}+J^{n-1}_k)$ is bounded below by 2^n.

Proposition 5.3.8 *Let $\Omega(R^n_{k,a})$ be the cycle decomposition of $R^n_{k,a}$. For any sequence $x \in \Omega(R^n_{k,a})$ with period at least 2^n-1, the linear complexity of x is bounded below by the linear complexity of the sequence generated by g.*

Proof We already showed in Lemma 5.3.5 that any sequence $x \in \Omega(R^n_{k,a})$ can be written as $x = b + c$ where $b \in \Omega(g)$, $c \in \Omega(\psi^k)$, and the minimal polynomial of b is coprime with the minimal polynomial of c. Since the minimal polynomial of b is coprime with the minimal polynomial of c, the linear complexity of x is equal to the sum of the linear complexities of b and c. Therefore, the linear complexity of x is greater or equal to the linear complexity of sequence b generated by g. Hence, the assertion is established. □

Remark 5.3.9 Using recurrence relation (5.1) with G as a linear function, one can generate a de Bruijn sequence with period 2^{n+k} and linear complexity at least $(2^{n+k-1}+n+k+1)$ for an arbitrary positive integer k. Nevertheless, this de Bruijn

sequence is not suitable for using it as a building block in designing a cryptographic primitives such as PRSGs or stream ciphers, because in the entire sequence most of the bits are linearly related to the internal state bits and only at $H(I_k^n)$ positions the bits are nonlinearly related to the internal state bits due to the nonlinear term I_k^n, which is vulnerable against a cryptanalytic attack. For a more detailed analysis on both linearly and nonlinearly composed de Bruijn sequences, we refer the reader to [71]. On the other hand, if the function g is nonlinear, then the bits of the de Bruijn sequence will be nonlinearly related to the internal state bits of the NLFSR, thereby a cryptanalytic attack would be more complex.

Propositions 5.3.3, 5.3.6, and 5.3.8 suggest that in order to generate a strong de Bruijn sequence by the composited construction, the starting span n sequence generated by g should have good randomness properties, particularly, long period and an optimal or near-optimal linear complexity. If an attacker is successful in approximating the feedback function R_k^n by the feedback function $(g \circ \psi^k)$, then the security of the sequence generated by R_k^n depends on the security of the sequence generated by g. If the de Bruijn sequences are used as building blocks in PRNGs and stream ciphers, an attack would not have direct access to a de Bruijn sequence.

5.4 Designing Parameters for Cryptographic de Bruijn Sequences

This section presents a few examples of strong de Bruijn sequences with period 2^{n+k} that are generated by an $(n+k)$-stage NLFSR for $19 \leq n \leq 24$ and $k = 16$. In order to generate de Bruijn sequences with period 2^{40}, we choose $n = 24$ and $k = 16$.

5.4.1 Tradeoff Between n and k

We observe that the parameter n is the measure of unpredictability of a sequence and the parameter k is the measure of efficiency for computing the feedback function. In the composited construction, one can construct an $(n+k)$-stage recurrence

relation by choosing a small value of n and a large value of k, since for a small value of n it is easy to find a span n sequence and the success probability of approximating the feedback function is low (see Proposition 5.3.3). However, for such a choice of the parameters, the recurrence relation contains many product-of-sum terms, as a result, the function I_k^n may not be calculated efficiently. Thus, for generating a strong de Bruijn sequence of period 2^{n+k} efficiently, one needs to choose the parameters in such a way that the nonlinearly generated span n sequence is large enough and the number of product-of-sum terms in I_k^n is as small as possible.

5.4.2 Examples of de Bruijn Sequences with Large Periods

Let $\{x_j\}_{j\geq 0}$ be a binary span n sequence generated by an n-stage recurrence relation, defined in Section 4.2.1 of Chapter 4, for a suitable choice of a decimation number d, a primitive polynomial $p(x)$, and a t-tap position

$$x_n = x_0 + f_d(x_{r_1}, x_{r_2}, ..., x_{r_t}) \tag{5.5}$$

where $(r_1, r_2, ..., r_t)$ with $0 < r_1 < r_2 < \cdots < r_t < n$ is called a t-tap position and f_d is a WG transformation. Here a decimation number is a coset leader that is coprime with $2^t - 1$. Then the recurrence relation (5.3) with G as a WG transformation can be written as

$$\begin{aligned} R_{16}^n = {} & x_{n+16} + x_n + x_0 + x_{16} + f_d(x_{r_1} + x_{r_1+16}, ..., x_{r_t} + x_{r_t+16}) + J_{16}^{n-1} \\ & + X_{15}^n + X_{14}^{n+1} + \cdots + X_1^{n+14} + X^{n+15} = 0 \end{aligned} \tag{5.6}$$

where $J_{16}^{n-1} = \prod_{i=1}^{n-1}(x_i + x_{i+16})$ and $X_k^p = T_{o,k}^p \cdot T_{e,k}^p, n \leq p \leq n+15, 1 \leq k \leq 15$, $T_{o,k}^p$ and $T_{e,k}^p$ are given in Table 5.3. The recurrence relation (5.6) can generate a de Bruijn sequence for a suitable choice of a decimation number d, a primitive polynomial $p(x)$, and a t-tap position. Following the representation of span n sequences in Chapter 4, our de Bruijn sequences are uniquely represented by the following four parameters:

1. the decimation number d,

2. the primitive polynomial $p(x)$,

3. the t-tap position $(r_1, r_2, ..., r_t)$, and

4. I_k^n.

Table 5.4 presents a few examples of cryptographically strong de Bruijn sequences with periods in the range of 2^{35} and 2^{40}. In Table 5.4, the computations for the linear complexity of the 24-stage span n sequence has not finished yet. However, currently the lower bound of the linear complexity is at least 2^{22}. For more instances of span n sequences with an optimal or near-optimal linear span, see Chapter 4 and Appendix A.

Table 5.4: De Bruijn sequences with periods $\geq 2^{35}$

WG over $\mathbb{F}_{2^t}$ t	Decimation d	Basis Polynomial $(c_0, c_1, ..., c_{t-1})$	t-tap positions $(r_1, r_2, ..., r_t)$	span n n	Linear Span, span n	I_k, k	Period 2^{n+k}
13	1207	$(1,0,1,0,1,0,1,0,1,0,1,0,0)$	$(1,2,3,4,5,6,7,10,11,12,13,15,22)$	24	$--$	16	2^{40}
13	55	$(1,1,1,0,1,1,0,0,1,1,0,1,0)$	$(1,2,3,4,5,6,9,10,11,12,13,15,17)$	24	$--$	16	2^{40}
8	53	$(1,1,1,0,0,1,1,1)$	$(1,2,5,6,8,11,12,15)$	21	$2^{21}-5$	16	2^{37}
8	29	$(1,1,1,0,0,0,0,1)$	$(1,2,6,8,9,15,16,19)$	21	$2^{21}-26$	16	2^{37}
8	31	$(1,1,1,0,0,0,0,1)$	$(1,2,10,12,13,16,18,19)$	20	$2^{20}-6$	16	2^{36}
8	1	$(1,1,0,0,0,1,1,0)$	$(1,3,4,5,8,11,12,15)$	19	$2^{19}-2$	16	2^{35}
7	5	$(1,0,0,1,1,1,0)$	$(1,2,6,8,10,12,16)$	20	$2^{20}-7$	16	2^{36}
7	19	$(1,0,1,0,0,1,1)$	$(1,2,3,5,6,10,18)$	19	$2^{19}-2$	16	2^{35}
5	1	$(1,1,1,0,1)$	$(5,10,12,18,19)$	20	$2^{20}-2$	16	2^{36}

Remark 5.4.1 Any feedback function g that generates a span n sequence can be used in recurrence relation (5.3) for producing a long de Bruijn sequence. To the best of our knowledge, Table 5.4 contains a set of (longest) de Bruijn sequences whose algebraic representations of the recurrence relations are known. We use the structured search for producing span n sequences to be used in the composited construction as we have seen in Chapter 4 that the structured search can produce span n sequences with optimal linear complexity in a systematic manner.

5.5 Implementation of Function I_{16}^n

We note that I_k^n is the most complicated module in the feedback function $R_k^n = 0$. Moreover, the function $g \circ \psi^k$ can be chosen efficiently. For a fixed value of n

and k, I_k^n is fixed, but the function g is different for different span n sequences. This section provides some techniques for optimizing the number additions in the product-of-sum terms in I_k^n for $k = 16$, and give an estimation for the number of multiplications and the time required for computing the function I_k^n in terms of n and k.

5.5.1 Optimizing the Number of Additions for I_{16}^n

For $k = 16$, I_k^n in recurrence relation (5.6) contains 17 product-of-sum terms. For example, for $n = 24$ and $k = 16$, one needs 2116 addition operations for computing all product terms in I_k^n. In Table 5.3, we can observe that many partial-sum terms appear in different product terms. By reusing the result of a previously computed sum term, we can optimize the number of additions. For $k = 16$, three optimization rules are described in Table 5.5. According to the above three rules

Table 5.5: Optimization rules for addition

Optimization Rule I (OR-I)			
$Y^1_{1,i} = x_i + x_{i+1}$ $Y^1_{2,i} = x_{i+4} + x_{i+5}$	$Y^2_{1,i} = x_{i+2} + x_{i+3}$ $Y^2_{2,i} = x_{i+6} + x_{i+7}$	$Y^1_{3,i} = x_{i+8} + x_{i+9}$ $Y^1_{4,i} = x_{i+12} + x_{i+13}$	$Y^2_{3,i} = x_{i+10} + x_{i+11}$ $Y^2_{4,i} = x_{i+14} + x_{i+15}$
$Y_{1,i} = Y^1_{1,i} + Y^2_{1,i}$ $Y_{3,i} = Y^1_{3,i} + Y^2_{3,i}$	$Y_{2,i} = Y^1_{2,i} + Y^2_{2,i}$ $Y_{4,i} = Y^1_{4,i} + Y^2_{4,i}$	$Y_{0,2,i} = x_i + x_{i+2}$ $Y_{8,10,i} = x_{i+8} + x_{i+10}$	$Y_{4,6,i} = x_{i+4} + x_{i+6}$ $Y_{12,14,i} = x_{i+12} + x_{i+14}$
$Q_{0,i} = x_i$ $Q_{8,i} = x_i + x_{i+8}$	$Q_{4,i} = x_i + x_{i+4}$ $Q_{12,i} = Q_{4,i} + x_{i+8} + x_{i+12}$	$Q_{3,i} = Y_{1,i}$ $Q_{11,i} = Q_{3,i} + Y_{3,i}$	$Q_{7,i} = Q_{3,i} + Y_{2,i}$ $Q_{15,i} = Q_{7,i} + Y_{3,i} + Y_{4,i}$
$Q_{2,i} = Y_{0,2,i}$ $Q_{10,i} = Q_{2,i} + Y_{8,10,i}$	$Q_{6,i} = Q_{2,i} + Y_{4,6,i}$ $Q_{14,i} = Q_{6,i} + Y_{8,10,i} + Y_{12,14,i}$	$Q_{1,i} = Y^1_{1,i}$ $Q_{9,i} = Q_{1,i} + Y^1_{3,i}$	$Q_{5,i} = Q_{1,i} + Y^1_{2,i}$ $Q_{13,i} = Q_{5,i} + Y^1_{3,i} + Y^1_{4,i}$
Optimization Rule II (OR-II)			
$Y^1_{1,i} = x_i + x_{i+1}$ $Y^1_{2,i} = x_{i+4} + x_{i+5}$ $Y_{4,6,i} = x_{i+4} + x_{i+6}$	$Y^2_{1,i} = x_{i+2} + x_{i+3}$ $Y^2_{2,i} = x_{i+6} + x_{i+7}$	$Y_{1,i} = Y^1_{1,i} + Y^2_{1,i}$ $Y_i = Y_{1,i} + Y_{2,i}$ $Y_{8,10,i} = x_{i+8} + x_{i+10}$	$Y_{2,i} = Y^1_{2,i} + Y^2_{2,i}$ $Y_{0,2,i} = x_i + x_{i+2}$
$W_{0,i} = x_i$ $W_{2,i} = Y_{0,2,i}$ $W_{8,i} = x_i + x_{i+8}$	$W_{1,i} = Y^1_{1,i}$ $W_{3,i} = Y_{1,i}$ $W_{9,i} = Y^1_{1,i} + x_{i+8} + x_{i+9}$	$W_{4,i} = x_i + x_{i+4}$ $W_{6,i} = Y_{0,2,i} + Y_{4,6,i}$ $W_{10,i} = Y_{0,2,i} + Y_{8,10,i}$	$W_{5,i} = Y^1_{1,i} + Y^1_{2,i}$ $W_{7,i} = Y_{1,i} + Y_{2,i}$
Optimization Rule III (OR-III)			
$Y_{1,i} = x_i + x_{i+1}$ $Z_{2,i} = x_i + x_{i+2}$	$Y_{2,i} = x_{i+2} + x_{i+3}$ $Z_{3,i} = Y_{1,i} + Y_{2,i}$	$Z_{0,1} = x_i$ $Z_{4,i} = x_i + x_{i+4}$	$Z_{1,i} = Y_{1,i}$

given in Table 5.5, the product terms in Table 5.3 can be written as that are given in Table 5.6. Applying the rules given in Tables 5.5, the total number of additions

Table 5.6: Product terms of the recurrence relation (5.6)

$T^{n}_{o,15} = \prod_{i \in Z^{n}_{o}} Q_{15,i}$	$T^{n+1}_{o,14} = \prod_{i \in Z^{n+1}_{o}} Q_{14,i}$
$T^{n+2}_{o,13} = \prod_{i \in Z^{n+2}_{o}} Q_{13,i}$	$T^{n+3}_{o,12} = \prod_{i \in Z^{n+3}_{o}} Q_{12,i}$
$T^{n+4}_{o,11} = \prod_{i \in Z^{n+4}_{o}} Q_{11,i}$	$T^{n+5}_{o,10} = \prod_{i \in Z^{n+5}_{o}} Q_{10,i}$
$T^{n+6}_{o,9} = \prod_{i \in Z^{n+5}_{o}} Q_{9,i} \cdot W_{9,n+6}$	$T^{n+7}_{o,8} = \prod_{i \in Z^{n+5}_{o}} Q_{11,i} \prod_{i=n+6,odd}^{n+7} W_{8,i}$
$T^{n+8}_{o,7} = \prod_{i \in Z^{n+5}_{o}} Q_{7,i} \cdot \prod_{i=n+6,odd}^{n+8} W_{7,i}$	$T^{n+9}_{o,6} = \prod_{i \in Z^{n+5}_{o}} Q_{6,i} \cdot \prod_{i=n+6,odd}^{n+9} W_{6,i}$
$T^{n+10}_{o,5} = \prod_{i \in Z^{n+5}_{o}} Q_{5,i} \cdot \prod_{i=n+6,odd}^{n+10} W_{5,i}$	$T^{n+11}_{o,4} = \prod_{i \in Z^{n+5}_{o}} Q_{4,i} \cdot \prod_{i=n+6,odd}^{n+11} W_{4,i}$
$T^{n+12}_{o,3} = \prod_{i \in Z^{n+5}_{o}} Q_{3,i} \cdot \prod_{i=n+6,odd}^{n+11} W_{3,i} \cdot Z_{3,n+12}$	$T^{n+13}_{o,2} = \prod_{i \in Z^{n+5}_{o}} Q_{2,i} \cdot \prod_{i=n+6,odd}^{n+11} W_{2,i} \cdot \prod_{i=n+12,odd}^{n+13} Z_{2,i}$
$T^{n+14}_{o,1} = \prod_{i \in Z^{n+5}_{o}} Q_{1,i} \cdot \prod_{i=n+6,odd}^{n+11} W_{1,i} \cdot \prod_{i=n+12,odd}^{n+13} Z_{1,i} \cdot (x_{n+14} + x_{n+15})$	$T^{n+15}_{o,0} = \prod_{i \in Z^{n+16}_{o}} x_i$
$T^{n}_{e,15} = \prod_{i \in Z^{n}_{e}} (Q_{15,i} + 1)$	$T^{n+1}_{e,14} = \prod_{i \in Z^{n+1}_{e}} (Q_{14,i} + 1)$
$T^{n+2}_{e,13} = \prod_{i \in Z^{n+2}_{e}} (Q_{13,i} + 1)$	$T^{n+3}_{e,12} = \prod_{i \in Z^{n+3}_{e}} (Q_{12,i} + 1)$
$T^{n+4}_{e,11} = \prod_{i \in Z^{n+4}_{e}} (Q_{11,i} + 1)$	$T^{n+5}_{e,10} = \prod_{i \in Z^{n+5}_{e}} (Q_{10,i} + 1)$
$T^{n+6}_{e,9} = \prod_{i \in Z^{n+5}_{e}} (Q_{9,i} + 1) \cdot (W_{9,n+6} + 1)$	$T^{n+7}_{e,8} = \prod_{i \in Z^{n+5}_{e}} (Q_{11,i} + 1) \prod_{i=n+6,even}^{n+7} (W_{8,i} + 1)$
$T^{n+8}_{e,7} = \prod_{i \in Z^{n+5}_{e}} (Q_{7,i} + 1) \cdot \prod_{i=n+6,even}^{n+8} (W_{7,i} + 1)$	$T^{n+9}_{e,6} = \prod_{i \in Z^{n+5}_{e}} (Q_{6,i} + 1) \cdot \prod_{i=n+6,even}^{n+9} (W_{6,i} + 1)$
$T^{n+10}_{e,5} = \prod_{i \in Z^{n+5}_{e}} (Q_{5,i} + 1) \cdot \prod_{i=n+6,even}^{n+10} (W_{5,i} + 1)$	$T^{n+11}_{e,4} = \prod_{i \in Z^{n+5}_{e}} (Q_{4,i} + 1) \cdot \prod_{i=n+6,even}^{n+11} (W_{4,i} + 1)$
$T^{n+12}_{e,3} = \prod_{i \in Z^{n+5}_{e}} (Q_{3,i} + 1) \cdot \prod_{i=n+6,even}^{n+11} (W_{3,i} + 1) \cdot (Z_{3,n+12} + 1)$	$T^{n+13}_{e,2} = \prod_{i \in Z^{n+5}_{e}} (Q_{2,i} + 1) \cdot \prod_{i=n+6,odd}^{n+11} (W_{2,i} + 1) \cdot \prod_{i=n+12,even}^{n+13} (Z_{2,i} + 1)$
$T^{n+14}_{e,1} = \prod_{i \in Z^{n+5}_{e}} (Q_{1,i} + 1) \cdot \prod_{i=n+6,even}^{n+11} (W_{1,i} + 1) \cdot \prod_{i=n+12,even}^{n+13} (Z_{1,i} + 1) \cdot (x_{n+14} + x_{n+15} + 1)$	$T^{n+15}_{e,0} = \prod_{i \in Z^{n+16}_{e}} (x_i + 1)$

required for computing I^n_{16} is given by $(n - 1 + 32 \cdot \lceil \frac{n+5}{2} \rceil + 32 \cdot \lfloor \frac{n+5}{2} \rfloor + 3 \cdot 18 + 3 \cdot 19 + 2 \cdot 5 + 2 \cdot 6 + 3 + 16) = (32 \cdot (n + 5) + n + 151)$, since the numbers of additions required for OR-I, OR-II and OR-III in Table 5.5 are 32, 18 and 5, respectively. For $n = 24$, the number of additions after applying the above three rules is equal to 1103.

5.5.2 Total Number of Multiplications and Time Complexity for Computing I^n_k

The maximum number of multiplications required for computing I^n_k is given by $\sum_{i=n-1}^{n+k-1}(i-1) = (n(k+1) + \frac{(k-1)(k-2)}{2} - 3)$ as one requires $(i-1)$ multiplications to compute a product of i numbers. In the following proposition, we estimate the time required for computing the function I^n_k.

Proposition 5.5.1 *The time required for computing the function I^n_k is approximately given by $\lceil ((k+1)\log_2 n + \frac{k(k-1)}{2n}\log_2 e) \rceil$ if $k << n$.*

Proof To compute a product-of-sum term X^p_k, $n \leq p \leq n + k - 1$, one requires at most $\lceil \log_2 p \rceil$-time. Since the function I^n_k contains $(k+1)$ product terms, the time complexity for computing I^n_k is given by

$$\sum_{p=n-1}^{n+k-1} \lceil \log_2 p \rceil \approx \lceil \log_2 (n^{k+1}(1-\frac{1}{n}) \prod_{i=1}^{k-1} (1+\frac{i}{n})) \rceil$$
$$\approx \lceil ((k+1)\log_2 n + \frac{k(k-1)}{2n} \log_2 e) \rceil \text{ if } k << n.$$

□

5.6 Summary of Chapter 5

In this chapter, we first refined the composited construction for producing a long period de Bruijn sequence from a short period span n sequence through the composition operation. We then determined the linear complexity of a composited de Bruijn sequence and performed an analysis of the feedback function of a composited de Bruijn sequence from the cryptographic point of view. In our analysis, we studied an approximation of the feedback functions, the cycle structure of an approximated feedback function, and determined the linear complexity of a sequence generated by an approximated feedback function. In addition, we presented a compact algebraic representation of an $(n+16)$-stage NLFSR, and a few instances of composited de Bruijn sequences with periods in the range of 2^{35} and 2^{40} together with the discussions of their implementation issues. A long period de Bruijn sequence produced by the composited construction can be used as a building block to design secure lightweight cryptographic primitives such as pseudorandom sequence generators and stream ciphers with desired randomness properties.

Chapter 6

Warbler Family: A Lightweight PRNG Family for Smart Devices

In this chapter, we present Warbler family – a new pseudorandom number generator family based on nonlinear feedback shift registers with desired randomness properties. In Section 6.1, we provide a detailed architectural description of the Warbler family, which is composed of two building blocks, namely a combination of modified de Bruijn blocks, and a nonlinear feedback WG generator. Then, we derive the randomness properties of sequences produced by the combination of modified de Bruijn blocks in Section 6.1.1, and give a general description of the initilization and running phases in Section 6.3. Randomness properties of output sequences of the Warbler family are inherited from the combination of modified de Bruijn blocks. In Section 6.4, some criteria for the selection of parameters of the Warbler family are proposed to offer a maximum level of security. Finally, we conclude this chapter in Section 6.5. The contents of the chapter can be found in [75].

6.1 Description of the Warbler PRNG Family

This section describes the general architecture of the Warbler PRNG family. Warbler is composed of a *combination of modified de Bruijn blocks* (CMDB) and a *nonlinear feedback Welch-Gong (WG) generator* (NFWGG) where the CMDB can be regarded

as a combinatorial generator which consists of a number of primitive nonlinear feedback shift registers. On the other hand, the nonlinear feedback WG generator can be regarded as an NLFSR over an extension field and which is similar to the key initialization phase of the WG cipher family [94]. Randomness properties of output sequences are inherited from the CMDB. The CMDB is protected by two different filtering functions. We now explain the design of Warbler whose CMDB contains m primitive NLFSRs of different lengths, and nonlinear feedback WG generator is defined over $\mathbb{F}_{2^n}$. A block diagram of the Warbler family is provided in Figure 6.1.

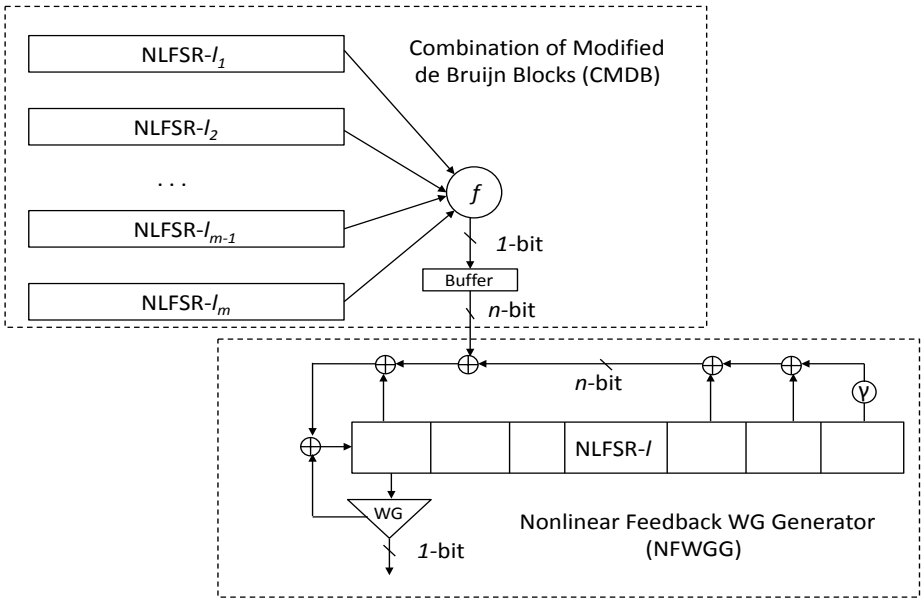

Figure 6.1: A general architecture of the Warbler family

6.1.1 Randomness Properties of the CMDB of Warbler Family

The combination of modified de Bruijn blocks of Warbler family is constituted by m distinct primitive NLFSRs whose recurrence relations are defined as (3.2). Let $l_1, l_2, ..., l_m$ be the lengths of the primitive NLFSRs where l_i's are pairwise coprime

with each other. We denote by NLFSR-l_i the NLFSR of length l_i with feedback function f_i, which generates a span n sequence of period $2^{l_i}-1$ that is denoted by $\mathbf{a}_i = \{a_{i,j}\}_{j\geq 0}$. We choose the NLFSRs in such a way that the linear complexities of the span n sequences are optimal or near-optimal. Let f be a Boolean function in m variables. Using sequences $\mathbf{a}_i$'s and function f, we generate a new binary sequence $\mathbf{s} = \{s_j\}$ as

$$s_j = f(a_{1,j}, a_{2,j}, ..., a_{m,j}), j \geq 0. \tag{6.1}$$

We note that the sequence $\mathbf{s}$ can be regarded as an output sequence of a combinatorial generator with NLFSRs. The randomness properties of sequence $\mathbf{s}$ depend on Boolean function f and primitive NLFSRs. We now present the period and linear complexity of the binary sequence $\mathbf{s}$ below.

Proposition 6.1.1 *Let l_1, l_2, ..., l_m be the lengths of the NLFSRs which are coprime with each other, then the period of the output sequence $\boldsymbol{s}$ is given by $\prod_{i=1}^{m}(2^{l_i}-1)$.*

The lengths of the NLFSRs are chosen to be co-prime with each other for achieving the maximum period of the sequence $\mathbf{s}$. Assume that the algebraic degree of f in Eq. (6.1) is k and the indices $\{i_i, i_2, ..., i_k\}$ appear in the leading monomial of f. For the maximum linear span or complexity of sequence $\mathbf{s}$, we make an arrangement of k NLFSRs in descending order as $l_{i_1} > l_{i_2} > \cdots > l_{i_k}$. Then we have the following proposition for the linear complexity.

Proposition 6.1.2 *Let each NLFSR produces a span n sequence with optimal linear span $(2^{l_i}-2)$, then the maximum linear complexity of sequence $\boldsymbol{s}$ is bounded below by approximately $\prod_{j=1}^{k}(2^{l_{i_j}}-2) \approx 2^{\sum_{j=1}^{k} l_{i_j}}$ for the above arrangement of the NLFSRs. The exact linear complexity of sequence $\boldsymbol{s}$ is equal to $f(2^{l_1}-2, \cdots, 2^{l_m}-2)$.*

For the NLFSR case, the proofs of the period and linear complexity can be done in the same way as proved for the LFSR case in [62]. For details, we refer the reader to [62].

Proposition 6.1.3 *Let l_1, $l_2, ..., l_m$ be the lengths of the NLFSRs which are co-prime with each other, then the number of occurrence of an m-tuple $(x_1, x_2, ..., x_m)$ produced by m NLFSRs is equal to $\frac{1}{2^m}\prod_{j=1}^{m}(2^{l_j} - 1 + (-1)^{x_j+1})$, $x_j \in \mathbb{F}_2$.*

Proof We prove the result by the mathematical induction on m. Let $m = 2$. Let $\mathbf{a} = \{a_0, a_1, ..., a_{M-1}\}$ and $\mathbf{b} = \{b_0, b_1, ..., b_{N-1}\}$ be two binary span n sequences of period $M = (2^{l_1} - 1)$ and $N = (2^{l_2} - 1)$, respectively with $\gcd(M, N) = 1$. We can write these two sequences in the interleave form as

$$\begin{pmatrix} (a_0, b_0) & (a_1, b_1) & ... & (a_{N-1}, b_{N-1}) \\ (a_N, b_0) & (a_{N+1}, b_1) & ... & (a_{2N-1}, b_{N-1}) \\ (a_{2N}, b_0) & (a_{2N+1}, b_1) & ... & (a_{3N-1}, b_{N-1}) \\ ... & ... & ... & ... \\ (a_{(M-1)N, b_0)} & (a_{(M-1)N+1}, b_1) & ... & (a_{MN-1}, b_{N-1}) \end{pmatrix}.$$

Denote $\delta = \frac{(M-1)(N-1)}{2^2}$. Then the number of (x, y), denoted as $\#(x, y)$, is given by

$$\#(x, y) = \begin{cases} \delta & \text{if } x = 0, y = 0 \\ \delta + 2(M-1) & \text{if } x = 0, y = 1 \\ \delta + 2(N-1) & \text{if } x = 1, y = 0 \\ \delta + 2(M+N-2) + 4 & \text{if } x = 1, y = 1. \end{cases}$$

Therefore, the result is true for $m = 2$. We now show that the result is also true for $m = 3$. Let $\mathbf{c} = \{c_0, c_1, ..., c_{P-1}\}$ be another binary span n sequence with period $P = 2^{l_3} - 1$ which is co-prime to both M and N. We define a new sequence $A_{jN+i} = (a_{jN+i}, b_i), 0 \leq i \leq N-1, 0 \leq j \leq M-1$ with period MN. In the similar fashion we can write the sequences A_{jN+i} and c_i in the interleave form as

$$\begin{pmatrix} (A_0, c_0) & (A_1, c_1) & ... & (A_{P-1}, c_{P-1}) \\ (A_P, c_0) & (A_{P+1}, c_1) & ... & (A_{2P-1}, c_{P-1}) \\ (A_{2P}, c_0) & (A_{2P+1}, c_1) & ... & (A_{3P-1}, c_{P-1}) \\ ... & ... & ... & ... \\ (A_{(MN-1)P}, c_0) & (A_{(MN-1)P+1}, c_1) & ... & (A_{MNP-1}, c_{P-1}) \end{pmatrix}.$$

From the interleave structure, the number of binary 3-tuple (x, y, z), denoted by $\#(x, y, z)$, is calculated as

$$\begin{aligned}\#(x,y,z) &= \frac{(M+(-1)^{x+1})(N+(-1)^{y+1})(P+(-1)^{z+1})}{2^3} \\ &= \frac{(2^{l_1}-1+(-1)^{x+1})(2^{l_2}-1+(-1)^{y+1})(2^{l_3}-1+(-1)^{z+1})}{2^3}.\end{aligned}$$

We assume that the result is true for $m = t$. Assume that $\{W_i\}$ is a t-tuple sequence of period $R = \prod_{i=1}^{t}(2^{l_i}-1)$ and each t-tuple $(x_1, x_2, ..., x_t)$ occurs $\frac{\prod_{i=1}^{t}(2^{l_i}-1+(-1)^{x_i+1})}{2^t}$ times. Let $\mathbf{q} = \{q_i\}$ be a span n sequence of period $Y = (2^{l_{t+1}} - 1)$. We now form a new sequence $S_i = (W_{jY+i}, q_i), 0 \le i \le Y-1, 0 \le j \le R-1$ with period YR, and the sequence can also be written in the form of an interleave structure as the above. The number of binary $(t+1)$-tuple $(x_1, x_2, ..., x_t, x_{t+1})$ is $\frac{\prod_{i=1}^{t}(2^{l_i}-1+(-1)^{x_i+1})}{2^t} \cdot \frac{(2^{l_{t+1}}-1+(-1)^{x_{t+1}+1})}{2}$ as the sequence $\mathbf{q}$ contains $\frac{(2^{l_{t+1}}-1+(-1)^{y+1})}{2}$ y's, $y = 0, 1$. Hence, the result is true for $m = (t+1)$. Thus, any m span n sequences of periods $2^{l_i} - 1$, $1 \le i \le m$, where each l_i's are co-prime to each other can be written in the form of the interleave structure and the number of occurrence of an m-tuple $x = (x_1, x_2, ..., x_m)$ is given by $\frac{1}{2^m}\prod_{j=1}^{m}(2^{l_j}-1+(-1)^{x_j+1})$. □

When the number of occurrences of each m-tuple is known, the imbalanced range of an output sequence can be calculated from the truth table of the function f. The imbalance range also depends on the arrangement of the NLFSRs. For sequence $\mathbf{s}$, the imbalance range needs to be minimized by keeping the linear complexity high. The randomness properties of the output sequence $\mathbf{s}$ are summarized as follows:

1. The period $P = \prod_{i=i}^{m}(2^{l_i}-1)$
2. The linear complexity at least $\prod_{j=1}^{k}(2^{l_{i_j}}-2)$
3. The imbalance range $\left|\frac{1}{2^m}\left(\sum_{x:f(x)=1}\prod_{j=1}^{m}(2^{l_j}-1+(-1)^{x_j+1}) - \sum_{x:f(x)=0}\prod_{j=1}^{m}(2^{l_j}-1+(-1)^{x_j+1})\right)\right|$.

We now produce a sequence $\mathbf{t}=\{t_k\}$ over $\mathbb{F}_{2^n}$ from sequence $\mathbf{s}$ as

$$t_k = (s_{nk}, s_{nk+1}, ..., s_{nk+n-1}) \in \mathbb{F}_{2^n}, k \geq 0$$

where $n \not\equiv 0 \bmod 3$ since a WG transformation is defined over $\mathbb{F}_{2^n}$ when $n \not\equiv 0 \bmod 3$.

Proposition 6.1.4 *The period of sequence* $\boldsymbol{t}$ *over* $\mathbb{F}_{2^n}$ *is equal to* P_t*,*

$$P_t = \begin{cases} \dfrac{n \cdot P}{\gcd(n, P)} & \text{if } n \nmid P \\ \dfrac{P}{n} & \text{if } n \mid P \end{cases}$$

where $P = \prod_{i=i}^{m}(2^{l_i} - 1)$*. For the maximum period of sequence* $\boldsymbol{t}$*,* $\gcd(n, 2^{l_i} - 1) = 1$ *for all* i*,* $1 \leq i \leq m$*.*

Since the characteristics of $\mathbb{F}_2$ and $\mathbb{F}_{2^n}$ are the same, the linear complexity of sequence $\mathbf{t}$ is bounded below by the linear complexity of sequence $\mathbf{s}$ [63]. We use the sequence $\mathbf{t}$ in the nonlinear feedback WG generator for providing nonlinearity, and to bound the period and linear complexity of the output sequence.

Remark 6.1.5 Span n sequences with optimal or near-optimal linear complexity for the CMDB can be found by the structured search.

6.1.2 Description of the Nonlinear Feedback WG Generator

The nonlinear feedback WG generator of Warbler has two components, namely a nonlinear recurrence relation and a WG transformation module. The nonlinear recurrence relation is composed of a primitive polynomial, the feedback sequence $\mathbf{t}$, and one bit feedback from the WG module, and that is used to update the internal state of the nonlinear feedback WG generator. Note that the WG transformation module contains two WG transformations where one WG transformation is used in the nonlinear recurrence relation and another one is used to filter the output

sequence. Let $p(x) = c_0 + c_1x + \ldots + c_{r-1}x^{l-1} + x^l$ be a primitive polynomial over $\mathbb{F}_{2^n}$. Let $\mathbf{z} = \{z_i\}$ be a sequence generated by an l-stage NLFSR whose nonlinear recurrence relation is defined as

$$z_{l+i} = c_0 z_i + c_1 z_{i+1} + \cdots + c_{l-1} z_{l+i-1} + t_i + w_i, i \geq 0 \tag{6.2}$$

where $w_i = (0, 0, \cdots, 0, g(z_{l-1+i}))$ and g is a WG transformation. The output sequence $\mathbf{o} = \{o_i\}$ of Warbler is obtained as

$$o_i = WG(z_{l+i-1}), i \geq 0 \tag{6.3}$$

where $WG(\cdot)$ is a WG transformation defined in Section 3.4.1. The reason for choosing WG transformations is that a WG transformation has excellent cryptographic properties such as high algebraic degree, high nonlinearity, 2-level autocorrelation and high linear span.

In Eq. (6.3), it can be noticed that each output bit o_i is related to $(n \cdot m)$ independent variables of the CMDB, since the function f is a function in m variables and the sequence $\mathbf{t}$ is constructed by taking n bits from each NLFSR. This can also be regarded as the CMDB is protected by a Boolean function with $(n \cdot m)$ variables.

Property 4 *The period of the output sequence* $\boldsymbol{o}$ *is a multiple of the period of sequence* $\boldsymbol{t}$.

The proof of the above property follows from Theorem 3 of [58]. Moreover, the linear span of the output sequence is greater than or equal to the linear span of the sequence produced by the CMDB since the output sequence $\mathbf{o}$ can be written in terms of the sequences $\mathbf{s}$ and $\mathbf{z}$.

We obtain u-bit random numbers from the binary sequence $\mathbf{o}$ by taking disjoint u-bit segments. Symbolically, u-bit random numbers $\{R_k\}_{k\geq 0}$ are generated from the output sequence $\mathbf{o} = \{o_i\}$ as

$$R_k = (o_{uk}, o_{uk+1}, ..., o_{uk+u-1}), k \geq 0.$$

We denote by Warbler(L, m, n, l) an instance of the Warbler family, which contains m primitive NLFSRs in the CMDB and the NLWGG is defined over $\mathbb{F}_{2^n}$, and the total length of the internal state of the CMDB is L and the length of the internal state of the NLWGG is $(n \cdot l)$. We present two lightweight instances, Warbler$(35, 2, 5, 6)$ and Warbler$(62, 3, 5, 6)$, of the Warbler family in Chapters 7 and 8, respectively. The parameters of the Warbler family are summarized in Table 6.1.

Table 6.1: Parameters description of the Warbler family

Description	Parameters
Number of NLFSRs in the CMDB	m
Length of NLFSR-l_i	l_i
Feedback function of NLFSR-l_i	f_i
Combining function in the CMDB	f
Finite field $\mathbb{F}_{2^n}$	n
Length of the NLFSR in the NFWGG	l
Primitive polynomial in the NFWGG	$p(x)$
Feedback function in the NFWGG	$g(x)$
Filtering function	$WG(x)$

6.2 Design Rationale

Warbler family is a lightweight PRNG family based on nonlinear feedback shift registers, and is designed for smart devices such as RFID tags. The objective of designing Warbler family is to design an NLFSR-based PRNG with guaranteed randomness properties such as period and linear complexity. The strength of our design is based on the difficulty of solving a large system of nonlinear multivariate equations over the binary field, since an NLFSR-based PRNG can be reduced to a system of nonlinear equations. The main reasons of employing NLFSRs in the design are to thwart known cryptanalytic attacks such as algebraic attacks, cube attacks, distinguishing attacks and discrete fourier transform (DFT) attacks against stream ciphers, and make compatible to resource-constrained environments with restriction on the speed, gate-count and power consumption. Another reason for

employing multiple NLFSRs in the CMDB is to generate shift distinct sequences, using one NLFSR it is impossible to generate shift distinct sequences for different initial states. Some applications, for instance the EPC C1 Gen2 standard, demand the output sequences to be shift distinct. Since the nonlinear feedback functions are used to update the internal states, the complexity of the algebraic attack would be high, and the attack may not be better than the exhaustive search. In our design, it is hard to determine the exact period and linear complexity of an output sequence and that depend on the initial state of the PRNG. As a result, the powerful DFT attack can be resisted. Since the CMDB determines randomness properties of output sequences, the CMDB is protected by two functions $f(x)$ and $WG(x)$. Moreover, the output sequence filtered by the WG transformation $WG(x)$ is related to the internal state of the NFWGG. Consequently, the divide-and-conquer attack cannot be mounted easily. The reason for selecting the nonlinear feedback WG generator is that a WG transformation has excellent cryptographic properties such as high algebraic degree, nonlinearity, linear span and WG transformations can be used for both feedback and filtering purposes.

6.3 Key Initialization Phase of Warbler

We note that the total number of bits in Warbler is $(\sum_{i=1}^{m} l_i + l \cdot n)$. Based on the lengths of the key and the initial vector (IV), we divide the whole internal state bits $(\sum_{i=1}^{m} l_i + l \cdot n)$ into two parts. The key can be uploaded at the predefined positions and the IV at the remaining positions. The internal state of Warbler in the initialization phase is updated as follows.

$$
\begin{aligned}
a_{1,l_1+i} &= a_{1,i} + f_1(a_{1,1+i}, a_{1,2+i}, ..., a_{1,l_1+i-1}) + o_i, i \geq 0, \\
a_{2,l_2+i} &= a_{2,i} + f_2(a_{2,1+i}, a_{2,2+i}, ..., a_{2,l_2+i-1}) + o_i, i \geq 0, \\
&\vdots \qquad \vdots \\
a_{m,l_m+i} &= a_{m,i} + f_m(a_{m,1+i}, a_{m,2+i}, ..., a_{m,l_m+i-1}) + o_i, i \geq 0, \\
s_{i+n-1} &= f(a_{1,i}, a_{2,i}, ..., a_{m,i}), s_j = 0, 0 \leq j \leq n-2, i \geq 0,
\end{aligned}
$$

$$
\begin{aligned}
t_i &= (s_i, ..., s_{i+n-1}) \in \mathbb{F}_{2^n}, i \geq 0, \\
z_{l+i} &= c_0 z_i + c_1 z_{1+i} + \cdots + c_{l-1} z_{l+i-1} + t_i + w_i, i \geq 0, \\
o_{1+i} &= WG(z_{l+i-1}), o_0 = 0, i \geq 0.
\end{aligned}
$$

Let $\ell = \max\{l_1, l_2, ..., l_m, l\}$ be the maximum value among the lengths of the NLFSRs. We must apply the above initialization process for 2ℓ rounds. The purpose of the key initialization phase is to make a complex algebraic relation among the key and IV bits. After ℓ rounds, all the key bits and IV bits would be in each NLFSR as the output bit o_i is fed in each NLFSR and the sequence $\mathbf{t}$ is added to the NFWGG. We remember that the construction of sequence $\mathbf{t}$ in the running phase and the initialization phase is different.

6.4 Optimal Security Conditions for the Warbler Family

This section provides a list of criteria for choosing the parameters for an Warbler instance in order to offer a maximum level of security. In [77], Mandal *et al.* proposed a set of criteria for choosing the optimal parameters for a WG transformation in the WG cipher family. As Warbler contains the NFWGG, the criteria for choosing the parameters of an Warbler PRNG is a combination of the criteria for choosing parameters of the NLFSRs in the modified de Bruijn block and the criteria for optimal parameters of WG transformations. The parameters for an Warbler PRNG are chosen as follows.

1. The lengths of the primitive NLFSRs in the CMDB should be as large as possible, and the number of NLFSRs in the CMDB should be as small as possible.

2. The linear span or linear complexity of a span n sequence generated by NLFSR-l_i should be optimal $(2^{l_i} - 2)$ or near-optimal $(2^{l_i} - 2 - k), k << 2^{l_i - 1}$.

3. The combining function f in Eq. (6.1) should have large algebraic degree,

correlation immunity, algebraic immunity, nonlinearity, and balance property. The algebraic degree and balance property of f determine the linear complexity and imbalanced range of the output sequence $\mathbf{s}$. Moreover, the high algebraic immunity and nonlinearity ensure the high linear complexity of the sequence. In other words, it prevents from approximating the output sequence to a low linear complexity as well as a low period sequence. Furthermore, the function must be chosen carefully so that the period and linear complexity of sequences produced by the CMDB cannot be reduced by setting an initial state to some special initial states, for example weak initial states.

4. The WG transformation used for the feedback purpose should have maximum algebraic degree as it would help to prevent algebraic attacks and cube attacks.

5. The WG transformation used for the filtering purpose must have the maximum algebraic degree and maximum algebraic immunity.

6. The WG transformation used for the filtering purpose should have nonlinearity as large as possible.

7. The WG transformation used for filtering should have low k-normal value as for large values k the internal state will have large bias. This condition is for resisting Mihaljević et al.'s attack.

Three criteria $(4) - (6)$ for the WG transformations are presented in [77].

6.5 Summary of Chapter 6

This chapter presented a family of pseudorandom number generators, named Warbler family for smart devices. Warbler family is a purely nonlinear feedback shift register based PRNG family with desirable randomness properties. Randomness properties of the output sequence of the Warbler family are derived. Parameter selection criteria for the Warbler family are proposed for offering a maximum level of security against known attacks. Two lightweight instances, Warbler$(35, 2, 5, 6)$

and Warbler$(62, 3, 5, 6)$, of Warbler family are presented in Chapters 7 and 8, respectively. It is worth to mention that the Warbler family is a general family, which can be applied to the case that requires higher security level by choosing suitable parameters.

Chapter 7

Warbler-I: A Lightweight PRNG for the EPC C1 Gen2 RFID Tags

This chapter presents an instance, Warbler$(35, 2, 5, 6)$, of the Warbler family, named Warbler-I based on nonlinear feedback shift registers for low-cost EPCglobal Class-1 Generation-2 (EPC C1 Gen2 in short) RFID tags. The EPC C1 Gen2 standard uses a couple of 16-bit random numbers in the tag identification protocol for identifying tags [29]. In Section 7.1, we review the previous proposals for the PRNG in compliance to the EPC C1 Gen2 RFID tags . Then, we describe the details of the design of Warbler-I in Section 7.2. In Section 7.3.1, the security properties of the proposed PRNG are analyzed in great detail by employing cryptographic statistical tests specified by the EPC C1 Gen2 standard as well as the NIST test suite. Various cryptanalysis techniques have been applied to demonstrate the attack resistant properties of the proposed PRNG in Section 7.3.2. Furthermore, a hardware implementation on a Xilinx Spartan-3 FPGA device shows that the new PRNG can be implemented using 46 slices. The details of the hardware implementation can be found in [72]. Section 7.6 summarizes the contribution of this chapter. The research results of this chapter have been published in [72–74]. In [72,74], Warbler-I is known by the name Warbler.

7.1 Motivation and Related Work

For most RFID applications, the security and privacy are important and even crucial requirements [60]. Since most protocols for securing RFID systems proposed so far are based on the usage of an on-board true random and/or pseudorandom number generator (TRNG/PRNG), a number of solutions have been proposed in the literature for implementing TRNGs/PRNGs on RFID tags [4,18,57,85,97]. All of the proposals for TRNGs are based on analog circuits that sample a random physical phenomenon like thermal noise. To the best of our knowledge, only three PRNGs have been proposed for the EPC C1 Gen2 passive tags [18, 85, 97], among which two proposals use TRNGs as a component and the security properties of those two PRNGs rely on the security of TRNGs. The motivation for designing Warbler-I is to reduce the high power consumption and area, and to increase the throughput of the PRNG. The basic idea of our design is to replace the TRNG in [18, 85] by a lightweight pseudorandom sequence generator with good statistical properties.

7.1.1 Che *et al.*'s PRNG

Che *et al.* [18] designed a PRNG based on a combination of an oscillator-based TRNG and a linear feedback shift register (LFSR) with 16 stages. In their design, the TRNG is implemented using an analog circuit and exploits thermal noise of the circuit. To introduce randomness, one truly random bit from the TRNG is XORed with each bit of a 16-bit sequence generated from the LFSR. In 16 clock cycles, a 16-bit random number is generated by the PRNG. Due to the linear structure, Che *et al.*'s scheme has been attacked by Melia-Segui *et al.* in [85] with a high success probability $\frac{(n+1)}{8n}$, where n is the length of the LFSR.

7.1.2 Melia-Segui *et al.*'s PRNG

To avoid such an attack on Che *et al.*'s PRNG, Melia-Segui *et al.* [85] proposed a similar design by employing multiple primitive polynomials instead of one in the

LFSR. The design consists of a true random source, a module with eight primitive polynomials, and a decoding circuit taking inputs from the true random source, where the decoding circuit is designed in such a way that the same primitive polynomial is not chosen consecutively. At each clock cycle, one primitive polynomial is chosen according to the decoding logic and true random bits for producing a pseudorandom bit. Thus, the PRNG produces a 16-bit random number in 16 clock cycles, and the security of the PRNG relies on the TRNG. Recently, Melia-Segui *et al.* [86] proposed J3Gen which contains four instances of PRNG for different lengths of the LFSR with different numbers of primitive polynomials. The design principle of J3Gen is also based on an LFSR with multiple primitive polynomials and a true random source. The security properties of all PRNGs are analyzed by performing the statistical tests proposed by the EPC C1 Gen2 standard.

7.1.3 Peris-Lopez *et al.*'s PRNG

In [97], Peris-Lopez *et al.* proposed a PRNG named LAMED for RFID tags, which is in compliance with the EPC C1 Gen2 standard and can provide 32-bit as well as 16-bit random numbers. The basic operations for updating the internal state of LAMED consist of bitwise XOR operations, modular algebra, and bit rotations. The internal state of the LAMED is of 64-bit, including a 32-bit key and a 32-bit initial vector. The key length can be further increased by replacing the IV bits with the key bits. Note that LAMED always outputs a 32-bit random number and a 16-bit random number is obtained by dividing 32-bit number into two equal halves and XORing them together.

7.2 Description of Warbler-I

Warbler-I is an NLFSR-based PRNG, which is composed of two main building blocks. The first one consists of two NLFSRs of length 17 and 18 over $\mathbb{F}_2$, each one generating a span n sequence or modified de Bruijn sequence with optimal linear complexity, whereas the second one includes a NLFSR over $\mathbb{F}_{2^5}$ and each NLFSR uses one or two WG transformations. In our design, the binary sequence

generated by the first building block is converted to a sequence over $\mathbb{F}_{2^5}$ and this sequence is used in the recurrence relation in the second building block. The final output sequence is filtered by the WG transformation and n-bit random numbers are generated by taking disjoint n-bit sequences from the final output sequence. A high-level architecture of the proposed PRNG is illustrated in Figure 7.1.

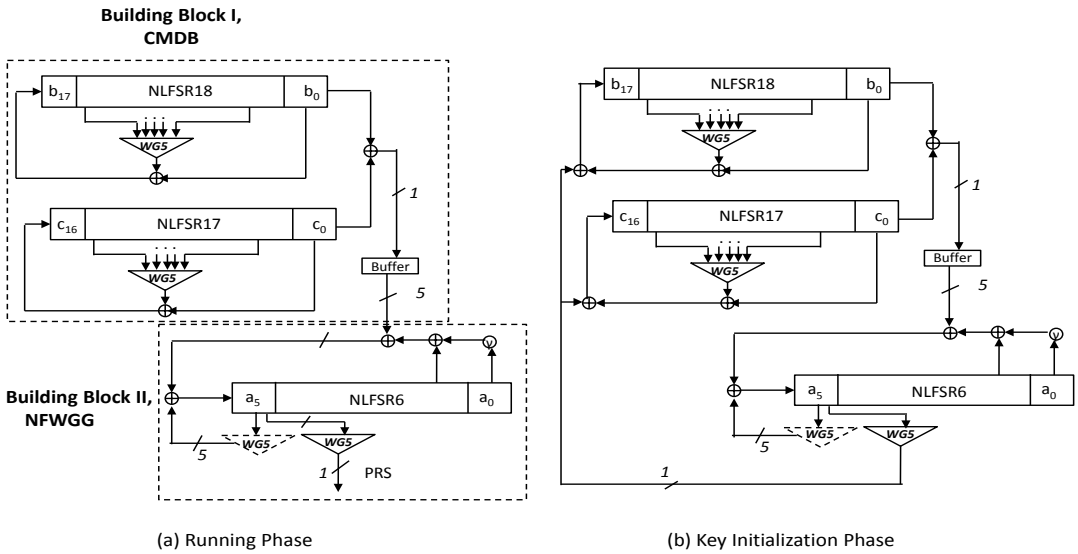

Figure 7.1: A diagram of Warbler-I for EPC C1 Gen2 tags

7.2.1 WG-5 Transformation

Finite field $\mathbb{F}_{2^5} = GF(2^5)$ is defined by a primitive element α such that $p(\alpha) = 0$ where $p(x) = 1 + x + x^3 + x^4 + x^5$ is a primitive polynomial over $\mathbb{F}_2$. The trace function, from $\mathbb{F}_{2^5} \to \mathbb{F}_2$, is given by $\mathrm{Tr}(x) = x + x^2 + x^{2^2} + x^{2^3} + x^{2^4}$. For $m = 5$, the WG permutation is

$$\mathrm{WGP}_5(x) = x + (x+1)^5 + (x+1)^{13} + (x+1)^{19} + (x+1)^{21}, x \in \mathbb{F}_{2^5},$$

and the WG transformation over $\mathbb{F}_{2^5}$, denoted by WG-5, is given by

$$WG(x) = f(x) = \mathrm{Tr}(\mathrm{WGP}_5(x)) = \mathrm{Tr}(x^{19}), x \in \mathbb{F}_{2^5}.$$

7.2.2 Building Block I: An Alternative to TRNG

The first building block contains two NLFSRs whose lengths are chosen to be co-prime in order to achieve the maximum period. The reason that two shorter NLFSRs are used instead of a long one is due to the impossibility of generating shift distinct sequences from a long NLFSR for different initial states. In other words, by XORing the output sequences from two NLFSRs we can obtain shift distinct sequences for different initial states. In our design, the WG transformation WG-5 over $\mathbb{F}_{2^5}$ is used as a nonlinear feedback function to generate span n sequences. We use the nonlinear recurrence relation for the structured search defined in Chapter 4 to generate span n sequences. Let $\mathbf{b} = \{b_i\}$ be a binary sequence generated by an n-stage NLFSR which is defined as

$$b_{n+k} = b_k \oplus f(x^d),\ x = (b_{r_1+k}, b_{r_2+k}, \ldots, b_{r_5+k}) \in \mathbb{F}_{2^5},\ b_i \in \mathbb{F}_2 \qquad (7.1)$$

for all $k \geq 0$, and $0 < r_1 < r_2 < \ldots < r_5 < n$ are tap positions of the NLFSR, where $\oplus$ denotes addition over $\mathbb{F}_2$. Using the parameters in Table 7.1 and recurrence relation (7.1), we can generate two span n sequences $\mathbf{b} = \{b_i\}_{i\geq 0}$ and $\mathbf{c} = \{c_i\}_{i\geq 0}$ with NLFSR18 and NLFSR17, respectively. These two span n sequences are obtained by the structured search in Chapter 4. The output sequence of the first building block is denoted by $\mathbf{s} = \{s_i \mid s_i = b_i \oplus c_i, i \geq 0\}$, which is almost balanced and has the following statistical properties:

a) The period is $(2^{18} - 1)(2^{17} - 1) \approx 2^{35}$;

b) The imbalance range is 4; and

c) The linear span is $(2^{17} - 2 + 2^{18} - 2) \approx 2^{18.585}$.

For different initial states of the NLFSRs, the number of shift distinct sequences ($\mathbf{s}$) is $(2^{18} - 1)(2^{17} - 1) - 2$.

We now generate a new sequence $\mathbf{t}$= $\{t_k\}_{k\geq 0}$ over $\mathbb{F}_{2^5}$ from $\mathbf{s}$ as follows

$$t_k = (s_{5k}, s_{5k+1}, s_{5k+2}, s_{5k+3}, s_{5k+4}) \in \mathbb{F}_{2^5}, \forall k \geq 0.$$

Table 7.1: Parameters and statistical properties of two primitive NLFSRs

NLFSR	Length n	Decimation d	Primitive polynomial $p(x)$ to generate $\mathbb{F}_{2^5}$	Tap positions $(r_1, r_2, r_3, r_4, r_5)$	Period	Linear Span
NLFSR18 (**b**)	18	3	$1 + x + x^3 + x^4 + x^5$	4, 7, 8, 10, 15	$2^{18} - 1$	$2^{18} - 2$
NLFSR17 (**c**)	17	3	$1 + x + x^3 + x^4 + x^5$	4, 7, 8, 9, 12	$2^{17} - 1$	$2^{17} - 2$

Table 7.2: Cryptographic properties of WG-5 transformations used in Warbler-I

$f(x)$, $p(x)$ to define $\mathbb{F}_{2^5}$	Cryptographic Properties
$f(x)$, $1 + x + x^3 + x^4 + x^5$	$deg(f) = 3$, $AI(f) = 3$, $NL_f = 12$
$f(x^3)$, $1 + x + x^3 + x^4 + x^5$	$deg(f(x^3)) = 3$, $AI(f(x^3)) = 3$, $N_{f(x^3)} = 12$

The period of the sequence $\mathbf{t}$ equals $2^{37.32}$. Note that the sequence $\mathbf{t}$ is a shift distinct sequence for different initial states of the NLFSRs and the linear complexity of sequence $\mathbf{t}$ is bounded below by $2^{18.58}$ [63]. The sequence $\mathbf{t}$ is used in the second building block for introducing nonlinearity in the recurrence relation in each 5 clock cycles (see Section 7.4 for details). This building block is used as an alternative to the TRNG in [18, 85].

7.2.3 Building Block II: Pseudorandom Number Generator

The second building block consists of an NLFSR and two WG transformation modules given by $f(x)$ and $f(x^3)$, respectively. Letting the length of NLFSR6 be $l = 6$ and the primitive polynomial be $g(x) = x^6 + x + \gamma$, where $\gamma = \alpha^{15} \in \mathbb{F}_{2^5}$, the recurrence relation is defined as

$$a_{k+6} = \gamma a_k + a_{k+1} + w_k + t_k, a_i \in \mathbb{F}_{2^5}, w_k = (0, 0, 0, 0, f(a_{k+5})), k \geq 0, \qquad (7.2)$$

where w_k is the nonlinear feedback with the least signification bit generated by WG transformation $f(x)$ and $\mathbf{t} = \{t_k\}_{k \geq 0}$ is the sequence over $\mathbb{F}_{2^5}$ that is defined in the previous subsection. While the WG transformation $f(x)$ is only used as a nonlinear feedback function in NLFSR6, the WG transformation $f(x^3)$ is employed as a nonlinear feedback for NLFSR18 and NLFSR17 as well as to filter the output

sequences. The cryptographic properties of the WG transformations $f(x)$ and $f(x^3)$ are provided in Table 7.2. In the above recurrence relation (7.2), the nonlinearity is introduced by t_k and w_k and those feedback will affect other bit positions after multiplying by γ. Note that the period of the sequence $\mathbf{a} = \{a_k\}_{k\geq 0}$ is a multiple of the period of $\mathbf{t}$. Moreover, the final output sequence $\mathbf{o} = \{o_k\}$ of the second building block is defined by $o_k = f(a^3_{5+k})$, for $k \geq 0$, where f is the WG transformation. The period of $\mathbf{o}$ is a multiple of $2^{37.32}$ and the linear complexity of $\mathbf{o}$ is lower bounded by the linear complexity of $\mathbf{t}$. The 16-bit random numbers $\{RN_k\}_{k\geq 0}$ are obtained using sequence $\{o_k\}$ as follows

$$RN_k = (o_{16k}, o_{16k+1}, \cdots, o_{16k+15}), k \geq 0.$$

7.2.4 System Initialization of Warbler-I

The proposed PRNG has an internal state of 65 bits, including a 45-bit secret seed as well as a 20-bit initial vector (IV). While the secret seed and the IV are preloaded into RFID tags at the very beginning, the 20-bit IV is also updated at the end of each protocol session. Before generating random numbers, a 36 rounds of initialization phase is applied to mix the key and IV properly. In our design, the secret seed and IV are preloaded as follows: the first consecutive 12, 11 and 22 positions of NLFSR18, NLFSR17 and NLFSR6 are respectively reserved for key bits, whereas the remaining positions in each NLFSR are for the IV. The initialization process is illustrated in Figure 7.1 (b). During the initialization phase the internal states of the three NLFSRs are updated as follows:

$$\begin{aligned}
b_{k+18} &= b_k \oplus f(x^3) \oplus o_k, x = (b_{k+4}, b_{k+7}, b_{k+8}, b_{k+10}, b_{k+15}), k \geq 0, o_0 = 0,\\
c_{k+17} &= c_k \oplus f(y^3) \oplus o_k, y = (c_{k+4}, c_{k+7}, c_{k+8}, c_{k+9}, c_{k+12}), k \geq 0, o_0 = 0,\\
s_{k+4} &= b_k \oplus c_k, k \geq 0, s_j = 0, j = 0, 1, 2, 3,\\
t_k &= (s_k, s_{k+1}, s_{k+2}, s_{k+3}, s_{k+4}), k \geq 0,\\
a_{k+6} &= \gamma a_k + a_{k+1} + w_k + t_k, w_k = (0, 0, 0, 0, f(a_{k+5})), k \geq 0,\\
o_{k+1} &= f(a^3_{5+k}), k \geq 0
\end{aligned}$$

where b_{k+18}, c_{k+17} and a_{k+6} are the updated values of NLFSR18, NLFSR17 and NLFSR6, respectively, and w_k is generated by the WG transformation $f(x)$. Sequence $\{s_k\}$ is the XOR of two output bits from NLFSR18 and NLFSR17 and five consecutive s_k's are collected to form a 5-bit vector t_k. The output o_k of NLFSR6 is used as a nonlinear feedback to affect the internal states of both NLFSR18 and NLFSR17.

Remark 7.2.1 The 20-bit IV can be generated from the initial SRAM state of tags when tags are powered up (see [57]). The entropy of IV can also be increased by employing the von Neumann technique, which can be efficiently implemented in hardware [109]. However, the implementation of these components needs additional hardware support.

7.3 Security Analysis of Warbler-I

The security analysis of the proposed PRNG is conducted in two steps. In the first step, we performed all cryptographic statistical tests that are specified in the EPC C1 Gen2 standard [29] and the NIST standard [103] on several sets of pseudorandom sequences generated by the proposed PRNG with different initial states. In the second step, we investigate the attack resistant properties of the new PRNG by launching the algebraic attacks, cube attacks, and time-memory-data tradeoff attacks.

7.3.1 Randomness Analysis of the PRNG

According to the EPC C1 Gen2 standard, a true random or pseudorandom number generator must satisfy the following three statistical properties:

– **Probability of a single sequence:** The probability that any 16-bit random sequence ($RN16$) drawn from the PRNG has value j, shall be bounded by $\frac{0.8}{2^{16}} < \Pr(RN16 = j) < \frac{1.25}{2^{16}}$, for any j.

– **Probability of simultaneously identical sequences:** For a tag population up to $10,000$, the probability that any of two or more tags simultaneously generate the same sequence of bits shall be less than 0.1%, regardless of when the tags are energized.

– **Probability of predicting a sequence:** A sequence drawn from the PRNG 10ms after the end of transmission shall not be predictable with a probability grater than 0.025% if the outcomes of prior draws from the PRNG, performed under identical conditions, are known.

We implemented our PRNG in software for checking whether the proposed PRNG meets the above three criteria. To verify the first criterion, we generated 18 different test sequences for different initial states of the NLFSRs and calculated the probability of occurrence of 16-bit numbers. Our experimental results show that the probability of any 16-bit number j, i.e., $\Pr(RN16 = j)$ lies between $\frac{0.9409}{2^{16}}$ and $\frac{1.0693}{2^{16}}$, which are better bounds than those obtained in [85]. The upper and lower bounds of probability values for different tests are given in Table 7.3a. With respect to the second criterion, our PRNG can generate up to $2^{45}-1$ shift distinct sequences for different keys to each tag, since the sequence $\mathbf{t}$ generated in Section 7.2.2 is shift distinct. Thus the probability that any two tags will generate the same sequence with period at least $2^{37.32}$ is $\approx 2^{-45}$ that is much less than 0.1%. For the third criterion, given a 16-bit random number, an attacker can recover the internal state of NLFSR6 with probability 2^{-24} after getting 80 bits of the sequence $\mathbf{s}$. To obtain the next 16-bit random number from the given one, the adversary needs to know the next consecutive 80 bits of the sequence $\mathbf{s}$ and the internal state of NLFSR6. The 80 bits can be obtained either by guessing or obtaining about $\frac{2^{18.58}}{5} = 2^{16.26}$ consecutive random numbers. Due to the high linear span of the sequence $\mathbf{s}$, it is impossible to generate the next consecutive 80 bits from previous known 80 bits in practice. Furthermore, it is also difficult for an adversary to intercept $2^{16.28}$ consecutive random numbers in one protocol session because the communication session in RFID systems is usually quite short and the IV is different. Moreover, the secret seed can also be updated for different sessions. Hence, the attacker can

guess the next 16-bit random number with the better probability 2^{-16}, which is much less than 0.025% as specified in the EPC C1 Gen2 standard.

To measure the linear dependency between an n-bit output and the previous n-bit output, we performed a serial correlation test [65] on the sequences generated by the PRNG. We generated 18 distinct sequences for different initial values of the NLFSRs, each one is of size 2^{26} bytes and calculated the serial correlation coefficient for 1-bit, 1-byte and 2-byte lag. Our experimental results demonstrate that the serial correlation coefficients are close to zero, which indicates the good randomness of the generated sequences. The serial correlation coefficients for different sequences are given in Table 7.3b.

Table 7.3: Successful fulfillment of the requirements of the EPC C1 Gen2 standard

(a) The first requirement

Sequences	**Upper**	**Lower**
S1	1.0471	0.9497
S2	1.0476	0.9530
S3	1.0444	0.9555
S4	1.0693	0.9517
S5	1.0468	0.9537
S6	1.0440	0.9545
S7	1.0457	0.9550
S8	1.0454	0.9560
S9	1.0533	0.9550
S10	1.0483	0.9544
S11	1.0541	0.9532
S12	1.0456	0.9514
S13	1.0487	0.9493
S14	1.0494	0.9523
S15	1.0506	0.9550
S16	1.0302	0.9850
S17	1.0499	0.9505
S18	1.0533	0.9409

(b) The third requirement

Sequences	**1-bit**	**1-byte**	**2-byte**
S1′	0.000098	-0.000080	-0.000061
S2′	-0.000012	0.0000025	-0.000055
S3′	0.000094	-0.000064	-0.000006
S4′	-0.000075	0.000106	-0.000046
S5′	0.000057	0.000041	-0.000041
S6′	-0.000012	0.000012	0.000078
S7′	-0.000063	-0.000028	0.000080
S8′	0.000025	0.000085	0.000032
S9′	-0.000002	-0.000005	-0.000042
S10′	0.000082	-0.000023	0.000023
S11′	0.000045	-0.000033	0.000046
S12′	0.000030	0.000026	0.0000012
S13′	-0.000006	0.000101	0.000071
S14′	-0.000053	-0.000047	0.000036
S15′	-0.000075	-0.000091	-0.000086
S16′	0.000015	0.000004	-0.000106
S17′	-0.000091	0.000025	-0.000067
S18′	0.000012	-0.000028	-0.000043

Different from the statistical tests in the EPC C1 Gen2 standard, the NIST test suite contains 15 demanding statistical tests for characterizing the randomness of a binary sequence. According to the NIST specification [103], a PRNG passes the

test suite successfully if it passes all the tests simultaneously with a proportion of 96%. In our experiment, 10 test sequence (TS) sets are generated, each of which has 100 different sequences with different initial values and each sequence has a length of 2^{25}. We computed the proportion values for each TS set and listed the test results[2] for 5 TS sets in Table 7.4. It is not difficult to find out that each TS set can pass the NIST test suite successfully.

Table 7.4: NIST test suite results of our proposal

Tests	TS1	TS2	TS3	TS4	TS5
	proportion	proportion	proportion	proportion	proportion
Frequency	0.97	1.00	0.99	0.98	1.00
Block-frequency	0.99	1.00	0.98	0.99	1.00
Cumulative-sum	0.97, 1.00	1.00, 1.00	0.97, 0.97	0.99, 0.99	0.99, 1.00
Runs	1.00	0.98	1.00	0.99	1.00
Longest-run	0.98	1.00	0.98	0.99	0.98
Rank	0.99	1.00	0.99	1.00	0.99
DFT	1.00	1.00	0.98	1.00	0.99
Overlapping-templates	0.96	0.97	0.97	0.97	0.99
Universal-stat.	0.99	0.98	1.00	1.00	0.99
Approx. entropy	0.99	1.00	0.98	0.97	0.99
Serial	0.99, 0.98	0.98, 0.98	1.00, 1.00	1.00, 1.00	0.99, 1.00
Linear-complexity	0.99	0.99	0.98	0.99	0.99
Random-excursions	0.97, 0.9	0.98, 1.00	0.98, 1.00	1.00, 0.99	0.99, 0.97
	0.97, 0.97	0.98, 0.97	1.00, 0.99	1.00, 0.98	0.98, 0.97
	0.98, 1.00	0.97, 0.97	1.00, 0.99	0.98, 0.97	0.99, 1.00
	0.97, 0.96	0.98, 0.97	0.98, 0.97	0.99, 0.98	1.00, 0.99
Random-excur-variant	0.98, 0.98, 0.98	1.00, 1.00, 1.00	1.00, 1.00, 1.00	0.99, 0.98, 0.99	0.98, 0.97, 0.99
	0.98, 0.98, 0.98	1.00, 0.97, 1.00	1.00, 1.00, 0.99	1.00, 1.00, 1.00	1.00, 1.00, 0.99
	1.00, 1.00, 0.99	1.00, 0.98, 0.98	1.00, 1.00, 1.00	1.00, 1.00, 1.00	0.99, 1.00, 0.99
	1.00, 1.00, 1.00	0.98, 0.98, 0.98	1.00, 1.00, 1.00	0.99, 1.00, 0.99	0.99, 0.99, 1.00
	0.98, 0.98, 0.98	0.98, 0.96, 0.96	1.00, 1.00, 1.00	0.97, 0.98, 1.00	1.00, 0.98, 1.00
	1.00, 1.00, 1.00	0.98, 0.98, 0.98	1.00, 0.99, 0.99	0.97, 0.96, 0.96	1.00, 0.99, 0.98

7.3.2 Cryptanalysis of **Warbler-I**

In this subsection, the attack resistant properties of the PRNG are investigated by considering the algebraic attacks, cube attacks, and time-memory-data tradeoff attacks in detail. Since our PRNG uses nonlinear feedback shift registers over

[2] Non-overlapping template matching test results are not given in Table 7.4 because of 148 entries. However, the proposed PRNG has passed the test successfully.

different fields, we also explain below why the correlation attacks [84], Discrete Fourier Transformation (DFT) attacks [46], and differential attacks [110] are not applicable.

Algebraic Attack

Algebraic attack [20] is a powerful attack against stream ciphers. In our PRNG design, nonlinear feedback functions are used to update the internal states of different NLFSRs and the output bits are filtered by the WG transformation. Noting that the length of the internal state of the PRNG is 65-bit and the length of the secret key is 45-bit, one can reduce the PRNG to a system of linear equations with about 2^{45} unknown variables, which can be solved by approximately $\frac{7}{64} \cdot (2^{45})^{\log_2 7}$ operations. As a result, the algebraic attack is not better than the exhaustive search in this case.

Cube Attack

Cube attack [24] is a generic key-recovery attack that can be applied to any cryptosystem, provided that the attacker can obtain a bit of information that can be represented by a low-degree decomposition multivariate polynomial in Algebraic Normal Form of the secret and public variables of the target cryptosystem. According to the cube attack, our PRNG can be regarded as a system of multivariate polynomials $p(k_1, ..., k_{45}, v_1, v_2..., v_{20})$ with public IV variables $v_1, v_2, ..., v_{20}$ and secret key variables $k_1, k_2, \ldots, k_{45}$. The polynomial

$$p(k_1, ..., k_{45}, v_1, v_2..., v_{20}) = t_I \cdot p_{S(I)} + q(k_1, ..., k_{45}, v_1, v_2..., v_{20})$$

is called a *master* polynomial, where $t_I = v_{i_1} v_{i_2} \cdots v_{i_k}$ is a monomial with $\{i_1, i_2, ..., i_k\} \subseteq \{1, 2, ..., 20\}$ and $p_{S(I)}$ is called a *superpoly* of t_I in p. The term t_I is called a *maxterm* if $\deg(p_{S(I)}) = 1$. We implemented the cube attack against our PRNG in CUDA and exploited the power of a GPU (i.e, a Tesla C2070 from NVIDIA) for accelerating the computation significantly. We took the first output bit after the 36-round initialization phase in order to find the maxterms in the master polyno-

mial and performed an exhaustive search over all possible cube dimensions ranging from 1 to 20. Our experiment was run for around 46 days on Tesla C2070 to exhaust all cube dimensions, but we did not find any linear and quadratic superpoly equations for different cube dimensions.

Time-Memory-Data Tradeoff Attack

Time-memory-data tradeoff attack is a generic cryptanalytic attack which can be applied to any cipher. In a stream cipher, the complexity of a time-memory-data tradeoff attack depends on the length of the internal state, which is given by $O(2^{\frac{n}{2}})$, where n is the length of the internal state [6]. We note that a stream cipher with low sampling resistance is vulnerable to a more flexible time-memory-data tradeoff attack. In our PRNG, the WG transformation is the filtering function as well as the internal state update function and the number of terms in the algebraic normal form representation of the WG transformation is 15, among which only two terms are linear and the remaining terms are either quadratic or cubic. Only by fixing four input variables in the WG transformation, one can obtain a linear function in one variable. Thus, the sampling resistance of the proposed PRNG is high. Since the length of the internal state is 65-bit in our PRNG, the expected complexity of the time-memory- data tradeoffs attack is $O(2^l)$, where $l = 32.5$.

Other Attacks

In the fast correlation attacks [84], the internal state of an LFSR based stream cipher can be recovered by first determining a system of linear equations according to a statistical model and then solving the system of linear equations. In our PRNG, the internal state is updated in a nonlinear way. Thus it is hard for an attacker to decide such a system of (non-)linear equations according to some statistical models.

For an LFSR based stream cipher, the DFT attacks [46] can be applied when the exact linear complexity of the output sequence and enough consecutive output bits are known. In our PRNG, the exact linear complexity of the output sequence is not known and hard to determine. Therefore, the DFT attacks cannot be applied

to our PRNG. Moreover, in the EPC C1 Gen2 standard protocol, it is hard for an attacker to obtain enough consecutive bits.

A chosen IV attack on the original version of WG cipher was presented in [110], where one can distinguish several bits of the output sequence by building a distinguisher based on differential cryptanalysis. In our PRNG, two nolinear terms w_k and t_k (i.e., an output from the WG transformation and a 5-bit tuple generated by the first building block) are added to the recurrence relation. Thus the differentials after 36 rounds of the initialization phase will contain most internal state bits. As a result, it would be hard for an attacker to distinguish output bits generated by the proposed PRNG.

7.4 Hardware Implementation of Warbler-I

To demonstrate the hardware complexity of the proposed lightweight PRNG, the PRNG module is implemented in VHDL for the low-cost Spartan-3 XC3S50 (Package PQ208 with speed grade -5) FPGA device from Xilinx, and our results are compared with other reported lightweight PRNG implementations. The hardware implementation shows that the proposed PRNG core totally occupies 46 slices (12 and 34 slices for building blocks I and II, respectively) on the target FPGA device and achieves a throughout of 45 Mbps. For the details of the implementation, we refer the reader to [72].

Table 7.5 presents a comparison with other PRNGs in terms of the hardware implementation and achieved randomness properties. One can notice that our PRNG has a lower hardware complexity than that in [97]. When compared to the PRNG proposed in [85], our design costs a similar number of logic gates with the usage of two NLFSRs replacing the TRNG in [85]. However, if we only compare the hardware implementation cost for the pseudorandom number generator module (i.e., the building block II in our design) in both proposals, our design only needs a half number of logic gates as that in [85]. Although the hardware complexity of our PRNG is slightly larger than that of SPONGENT-80, our design can provide desirable randomness properties such as period and linear complexity that cannot

be guaranteed by SPONGENT-80. For AKARI-1/2, the implementation cost of an instance depends on the length of the output random number, and in Table 7.5, we present the hardware implementation cost for the instances that generate 16-bit random numbers.

Table 7.5: A comparison with other PRNGs

Functions	Size of the internal state	Area	Device	Randomness Properties	
				Period	LS
Warbler-I	65	46 Slices/760 GE (est.)	XC3S50-PQ208	$\geq 2^{37.32}$	$\geq 2^{18.58}$
LAMED [97]	64	1585 GE (est.)	–	–	–
Melia-Segui *et al.* [85]	16	761 GE (est.)	–	–	–
SPONGENT-80 [7]	88	738 GE	0.13 μm CMOS	–	–
AKARI-1 A/B [78]	64	1018 (GE)/922 (GE)	90 nm CMOS	–	–
AKARI-2 A/B/C [78]	128	1861/1650/1620 (GE)	90 nm CMOS	–	–

7.5 Applications in RFID Systems

We have designed Warbler-I for the low-cost EPC C1 Gen2 passive RFID tags. Warbler-I can be used to generate 16-bit random numbers in the tag identification protocol. Warbler-I passed all the statistical tests specified the EPC C1 Gen2 standard as well as the NIST standard. Our PRNG is also resistant to the cryptanalytic attacks against stream ciphers.

In terms of the time delay for generating the first 16-bit pseudorandom number, our design totally requires 134 clock cycles, including 18 clock cycles for loading key and IV, 36 clock cycles for the initialization, and 80 clock cycles for generating the first 16-bit random number. After that, each 16-bit random number can be obtained every 80 clock cycles. Assuming that the EPC tags run at the clock frequency of 100 KHz and two 16-bit random numbers are needed for the tag identification protocol according to the EPC C1 Gen2 standard, one can identify about 510 tags in one second by using the proposed lightweight PRNG. Warbler-I perfectly meets the requirements on the gate-count/area and the security of the EPC C1 Gen2 standard.

Remark 7.5.1 In the proposed PRNG, we can update the 45-bit key at the end

of each session by generating 45 extra bits in 225 clock cycles and these 45 bits will be loaded at proper aforementioned key positions. This key updating procedure can be used to provide better security. In this way it is possible to generate at least $2^{16.26} \times 2^{20}$ consecutive random numbers for one key and for different IVs.

7.6 Summary of Chapter 7

In this chapter, we proposed a lightweight pseudorandom number generator, Warbler-I, which is in compliance to the EPC Class-1 Generation-2 standard and has guaranteed randomness properties such as period and linear span. Considering the high power-consumption, large area and low throughput of TRNGs, we replace the TRNG used in previous works by a PRNG with good statistical properties. In our design, the pseudorandom sequence is generated using a nonlinear feedback shift register. Moreover, the statistical tests specified by the EPC C1 Gen2 and the NIST standards, algebraic attacks, cube attacks and time-memory-data tradeoff attacks are employed to characterize the security properties of the proposed PRNG. A comparison with the sponge-based PRNGs is also conducted. In addition, an FPGA implementation shows that the proposed PRNG can be implemented using 46 slices (approximately 760 GE) and can generate a 16-bit random number every 80 clock cycles after an initialization process of 36 clock cycles.

Chapter 8

Warbler-II: A Lightweight PRNG for RFID Tags

In this chapter, we present another instance, Warbler$(62, 3, 5, 6)$, of the Warbler family, named Warbler-II, which contains three primitive NLFSRs in the combination of modified de Bruijn blocks, and one NLFSR over $\mathbb{F}_{2^5}$ of length 6 in the NFWGG. The goal of designing Warbler-II is to offer a better security level compare to Warbler-I. In Section 8.1, we describe the mathematical details and the details of the running and initialization phases of Warbler-II. Randomness properties of output sequences produced by Warbler-II are derived. We present a security analysis of Warbler-II in great detail, which is twofold. First, we perform the cryptographic statistical tests recommended by the EPC C1 Gen2 standard and NIST in Section 8.2.1. Then, in Section 8.2.2, we apply several cryptanalytic attacks such as algebraic attacks, cube attacks, time-memory-data tradeoff attacks and Mihaljević *et al.*'s attacks against Warbler-II. In addition, an implementation of Warbler-II in VHDL for the low-cost Spartan-3 XC3S50 FPGA device shows that the PRNG requires about 58 slices. Finally, we conclude this chapter in Section 8.6. The research results in this chapter can be found in [75].

8.1 Description of Warbler-II

This section presents the design details of Warbler-II. The CMDB of Warbler-II contains three NLFSRs, and the construction of Warbler-II is also based on the WG transformations over $\mathbb{F}_{2^5}$. The length of the internal state of the PRNG is 92 bits including 60-bit for the secret key and 32-bit for the initial vector. Our second PRNG is dedicated to the passive RFID tags. An overview of the architecture of Warbler-II is provided in Figure 8.1.

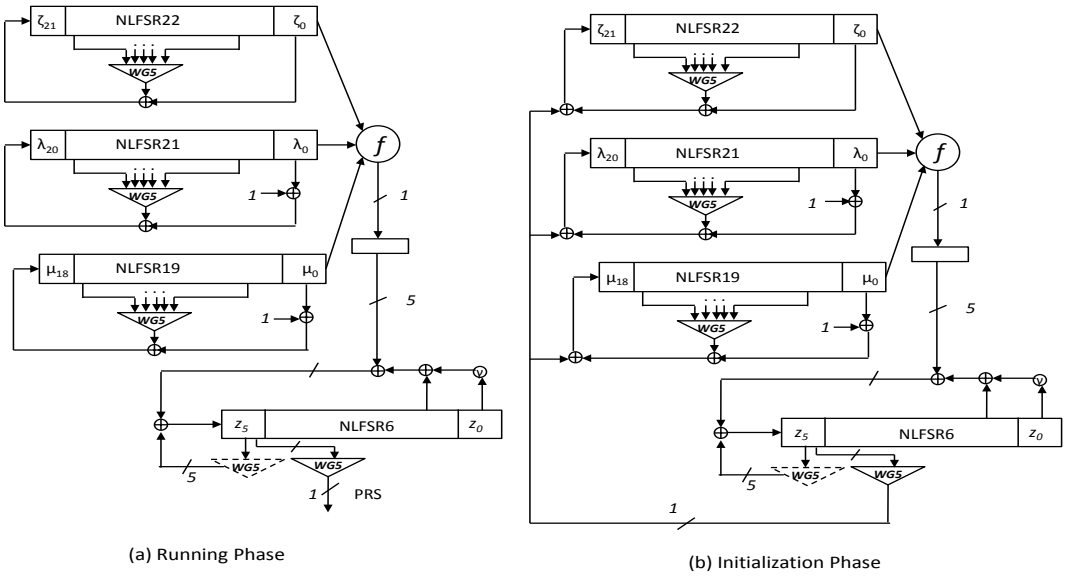

Figure 8.1: A block diagram of Warbler-II

8.1.1 Mathematical Functions of Warbler-II

In this section, we give the mathematical functions to be used in Warbler-II. We use a primitive polynomial $A(x)$ over $\mathbb{F}_2$ to define finite field $\mathbb{F}_{2^5}$, and the WG transformation is defined over $\mathbb{F}_{2^5}$. In our design, the WG transformations with decimations $d = 3, 7$ and 11 over $\mathbb{F}_{2^5}$ are used as nonlinear feedback functions to generate span n sequences, and the WG transformation with decimation $d = 3$ is used as a filtering function. For the definition of the WG transformation over

$\mathbb{F}_{2^5}$, we refer the reader to Section 7.2.1. Table 8.1 summarizes the cryptographic properties of all the WG transformations over $\mathbb{F}_{2^5}$ that are used in the PRNG.

Table 8.1: Cryptographic properties of WG-5 transformations used in Warbler-II

$WG(x)$, $A(x)$ to define $\mathbb{F}_{2^5}$	Cryptographic Properties
$WG(x^3)$, $1+x+x^3+x^4+x^5$	$deg(WG)=3$, $AI(WG)=3$, $N_{WG}=12$
$WG(x^{11})$, $1+x+x^3+x^4+x^5$	$deg(WG)=4$, $AI(WG)=3$, $N_{WG}=10$
$WG(x^{11})$, $1+x^3+x^5$	$deg(WG)=4$, $AI(WG)=3$, $N_{WG}=10$
$WG(x^{11})$, $1+x+x^2+x^4+x^5$	$deg(WG)=4$, $AI(WG)=3$, $N_{WG}=10$
$WG(x^7)$, $1+x^2+x^3+x^4+x^5$	$deg(WG)=2$, $AI(WG)=2$, $N_{WG}=12$

Let $\{b_i\}$ be a binary sequence generated by an n-stage NLFSR whose nonlinear recurrence relation is defined, in Chapter 4, as

$$b_{n+k} = b_k \oplus WG(x^d),\ x = (b_{r_1+k}, b_{r_2+k}, \ldots, b_{r_5+k}) \in \mathbb{F}_{2^5} \tag{8.1}$$

for all $k \geq 0$, and $0 < r_1 < r_2 < \ldots < r_5 < n$ are tap positions of the NLFSR, where $\oplus$ denotes addition over $\mathbb{F}_2$. Again, let $\{e_i\}$ be a binary sequence generated by an n-stage NLFSR whose nonlinear recurrence relation is defined as

$$\begin{aligned} e_{n+k} &= 1 \oplus e_k \oplus WG(x^d),\ x = (e_{r_1+k}, e_{r_2+k}, \ldots, e_{r_5+k}) \in \mathbb{F}_{2^5} \\ c_i &= e_i \oplus 1. \end{aligned} \tag{8.2}$$

where $WG(x)$ is the WG transformation over $\mathbb{F}_{2^5}$. Sequences $\{b_i\}$ and $\{c_i\}$ can be span n sequences for proper selections of parameters. Span n sequences generated by recurrence relations (8.1) and (8.2) are represented by three parameters, namely decimation number d, primitive polynomial $A(x)$, and tap position $(r_1, r_2, ..., r_5)$. We use these two types of recurrence relations in the combination of modified de Bruijn blocks to generate span n sequences.

8.1.2 Description of the CMDB of Warbler-II

The combination of modified de Bruijn blocks is composed of three NLFSRs, namely NLFSR22, NLFSR21, and NLFSR19 of lengths 22, 21 and 19, respectively are chosen

to be co-prime with each other to achieve the maximum period. We use the nonlinear recurrence relation (8.1) for NLFSR22, and nonlinear recurrence relation (8.2) for NLFSR21 and NLFSR19 for generating three span n sequences. Using nonlinear recurrence relations (8.1) and (8.2) and the parameters in Table 8.2, we can generate three span n sequences with optimal linear span. We denote by $\zeta = \{\zeta_i\}$, $\lambda = \{\lambda_i\}$ and $\mu = \{\mu_i\}$ the internal states of NLFSR22, NLFSR21, and NLFSR19, respectively.

Table 8.2: Parameters and statistical properties of three primitive NLFSRs

NLFSRs	Length n	Decimation d	Primitive polynomials $A(x)$ to generate $\mathbb{F}_{2^5}$	Tap Positions $(r_1, r_2, r_3, r_4, r_5)$	Period	Linear Span
NLFSR22 ($\zeta = \{\zeta_i\}$)	22	11	$1 + x^3 + x^5$	$(3, 4, 8, 12, 20)$	$2^{22} - 1$	$2^{22} - 2$
NLFSR21 ($\lambda = \{\lambda_i\}$)	21	11	$1 + x + x^2 + x^4 + x^5$	$(4, 10, 12, 15, 20)$	$2^{21} - 1$	$2^{21} - 2$
NLFSR19 ($\mu = \{\mu_i\}$)	19	7	$1 + x^2 + x^3 + x^4 + x^5$	$(3, 6, 14, 16, 18)$	$2^{19} - 1$	$2^{19} - 2$

We now combine the outputs of three NLFSRs by a 3-variable Boolean function to produce a new output sequence. The combining Boolean function is given by

$$f(x_0, x_1, x_2) = x_0x_1 + x_1x_2 + x_0x_2 + x_0 + x_1.$$

The function f is balanced, having maximum nonlinearity 2, and algebraic immunity 2. The reason for choosing a quadratic function is to increase the linear span of an output sequence produced by the combination of modified de Bruijn blocks. The output sequence $\mathbf{s} = \{s_i\}$ is defined by

$$s_i = f(\zeta_i, \lambda_i \oplus 1, \mu_i \oplus 1)$$

where $\{\lambda_i \oplus 1\}_{i\geq 0}$ and $\{\mu_i \oplus 1\}_{i\geq 0}$ are span n sequences according to recurrence relation (8.2), and $\{\zeta_i\}_{i\geq 0}$ is a span n sequence generated by recurrence relation (8.1). The statistical properties of sequence $\mathbf{s}$ are:

1. The period $(2^{22} - 1)(2^{21} - 1)(2^{19} - 1) \approx 2^{62}$
2. The linear span or complexity $2^{43.39}$
3. The imbalance range $2^{44.32}$.

Note that for an all-zero initial state of the NLFSRs, the output sequence $\mathbf{s}$ is a nonzero sequence. The output sequence can be a zero sequence when the initial state of NLFSR22 is all-zero and the initial states of NLFSR21 and NLFSR19 are all-one. The probability of occurring such a situation is $\frac{1}{2^{62}}$. We have chosen the combining function f with three quadratic terms for keeping the period and linear span of sequence $\mathbf{s}$ approximately at least 2^{40} when one of three NLFSRs produces an all-zero or all-one sequence.

We now generate a new sequence $\mathbf{t} = \{t_i\}$ as follows

$$t_i = (s_{5i}, s_{5i+1}, s_{5i+2}, s_{5i+3}, s_{5i+4}) \in \mathbb{F}_{2^5}, \forall i \geq 0.$$

According to the design, the shift distinct sequence $\mathbf{t}$ is added into the nonlinear recurrence relation of the nonlinear feedback WG generator. Note that the period of sequence $\mathbf{t}$ is equal to approximately $2^{64.32}$, which follows from Proposition 6.1.4.

8.1.3 Description of the NFWGG of Warbler-II

The mathematical functions used in the recurrence relation in the nonlinear feedback WG generator are the same as the mathematical functions used in the nonlinear feedback WG generator of Warbler-I with one exception. Here we use the WG transformation $WG(x^{11})$ instead of $WG(x)$ as nonlinear feedback for introducing more nonlinearity in the internal state. The 6-stage NLFSR over $\mathbb{F}_{2^5}$, denoted by NLFSR6, is defined as

$$a_{k+6} = \gamma a_k + a_{k+1} + w_k + t_k, a_i \in \mathbb{F}_{2^5}, w_k = (0, 0, 0, 0, WG(a_{k+5}^{11})), k \geq 0, \quad (8.3)$$

where $g(x) = x^6 + x + \gamma$ with $\gamma = \alpha^{15} \in \mathbb{F}_{2^5}$ is a primitive polynomial over $\mathbb{F}_{2^5}$, w_k is the nonlinear feedback with the least signification bit generated by WG transformation $WG(x^{11})$ and $\mathbf{t} = \{t_k\}_{k\geq 0}$ is a sequence over $\mathbb{F}_{2^5}$ that is produced in the CMDB. We choose the decimation $d = 11$ in $WG(x^{11})$ because the function $WG(x^{11})$ has the maximum algebraic degree 4 and that will rapidly increase the algebraic degree of the key bits and IV bits in the internal state. While the

WG transformation $WG(x^{11})$ is used as a nonlinear feedback function in NLFSR6, the WG transformation $WG(x^3)$ is employed to filter the output sequences. The cryptographic properties of $WG(x^3)$ can be found in Table 8.1. In recurrence relation (8.3), the nonlinearity is introduced by t_k and w_k, and the feedback w_k will affect other components and bit positions after multiplying by γ. Note that the period of the sequence $\mathbf{a} = \{a_k\}_{k\geq 0}$ is a multiple of the period of $\mathbf{t}$. Moreover, the final output sequence $\mathbf{o} = \{o_k\}$ of the NFWGG is defined by $o_k = WG(a_{5+k}^3)$ for $k \geq 0$. The period of $\mathbf{o}$ is a multiple of $2^{64.32}$ and the linear complexity of $\mathbf{o}$ is lower bounded by the linear complexity of $\mathbf{t}$ [63].

We obtain n-bit random numbers by taking disjoint n-bit segments from the output sequence. In particular, the 16-bit random numbers are obtained from the output sequence $\mathbf{o}$ as follows

$$RN_i = (o_{16i}, o_{16i+1}, ..., o_{16i+15}), i \geq 0.$$

8.1.4 Key Initialization Phase of Warbler-II

Our PRNG has an internal state of 92 bits, uses a 60-bit secret key (seed) as well as a 32-bit initial vector (IV). While the secret seed and the IV are preloaded into RFID tags at the very beginning, but the 32-bit IV can also be updated at the end of each protocol session. Before generating random numbers, we must execute the generator for 44 rounds to mix the key and IV properly. In our design, the secret seed and IV are preloaded as follows: the first consecutive 14, 13, 13 and 20 positions of NLFSR22, NLFSR21, NLFSR19 and NLFSR6 are respectively reserved for key bits, whereas the remaining positions in each NLFSR are reserved for the IV. The initialization process is illustrated in Figure 8.1 (b). During the initialization phase the internal states of three NLFSRs are updated as follows:

$$\begin{aligned}
\zeta_{k+22} &= \zeta_k \oplus WG(x^{11}) \oplus o_k, x = (\zeta_{k+3}, \zeta_{k+4}, \zeta_{k+8}, \zeta_{k+12}, \zeta_{k+20}),\\
\lambda_{k+21} &= 1 \oplus \lambda_k \oplus WG(y^{11}) \oplus o_k, y = (\lambda_{k+4}, \lambda_{k+10}, \lambda_{k+12}, \lambda_{k+15}, \lambda_{k+20}),\\
\mu_{k+19} &= 1 \oplus \mu_k \oplus WG(z^7) \oplus o_k, z = (\mu_{k+3}, \mu_{k+6}, \mu_{k+14}, \mu_{k+16}, \mu_{k+18}),
\end{aligned}$$

$$
\begin{aligned}
s_{k+4} &= f(\zeta_i, \lambda_i \oplus 1, \mu_i \oplus 1), \\
&= \zeta_i(\lambda_i \oplus 1) \oplus (\lambda_i \oplus 1)(\mu_i \oplus 1) \oplus (\mu_i \oplus 1)\zeta_i \oplus \zeta_i \oplus \lambda_i \oplus 1, s_j = 0, 0 \leq j \leq 3, \\
t_k &= (s_k, s_{k+1}, s_{k+2}, s_{k+3}, s_{k+4}), k \geq 0, \\
a_{k+6} &= \gamma a_k + a_{k+1} + w_k + t_k, w_k = (0, 0, 0, 0, WG(a_{k+5}^{11})), k \geq 0, \\
o_{k+1} &= WG(a_{5+k}^{3}), k \geq 0, o_0 = 0
\end{aligned}
$$

where $\zeta_{k+22}, \lambda_{k+21}$, μ_{k+19} and a_{k+6} are the updated values of NLFSR22, NLFSR21, NLFSR19 and NLFSR6 respectively, and w_k is generated by the WG transformation $WG(x^{11})$. Sequence $\{s_k\}$ is the output of f which takes three bits from NLFSR22, NLFSR21 and NLFSR19 as input and five consecutive s_k's are collected to form a 5-bit vector t_k. The output o_k of NLFSR6 is used as a nonlinear feedback to affect the internal states of NLFSR22, NLFSR21 and NLFSR19.

8.2 Security Analysis of Warbler-II

This section conducts a detailed security analysis of Warbler-II. Our analysis is twofold. We start our analysis by performing the statistical tests proposed by the EPC C1 Gen2 standard and the NIST standard. Then, in the second step, we conduct a detailed cryptanalysis on the proposed PRNG by considering algebraic attacks, cube attacks, time-memory-data tradeoff attacks, Mihaljević *et al.*'s attacks, and weak internal state and fault injection attacks.

8.2.1 Cryptographic Statistical Tests

In this section, we present the results obtained by performing the statistical tests recommended by the EPC C1 Gen2 standard and the NIST standard.

EPC C1 Gen2 Statistical Test Results

The EPC C1 Gen2 standard specified three statistical properties that a PRNG must satisfy in to order to be used in that standard. Three statistical properties are provided in Section 7.3.1. We implemented Warbler-II in software for checking

whether Warbler-II PRNG meets three EPC standard's criteria as well as NIST's randomness test criteria. To verify the first criterion, we generated 18 different test sequences for different keys and initial vectors of the NLFSRs and the lengths of test sequences lie in the range of 2^{26} to 2^{29}. We calculated the probability of occurrence of 16-bit numbers. Our experimental results show that the probability of any 16-bit number j, i.e., $\Pr(RN16 = j)$ lies between $\frac{0.8769}{2^{16}}$ and $\frac{1.0981}{2^{16}}$. The upper and lower bounds of probability values for different tests are given in Table 8.3a. With respect to the second criterion, our PRNG can generate up to $2^{62}-1$ shift distinct sequences for different keys to each tag, since the sequence $\mathbf{t}$ generated in Section 8.1.2 is shift distinct. Thus the probability that any two tags will generate the same sequence with period at least $2^{64.32}$ is $\approx 2^{-62}$ that is much less than 0.1%. For the third criterion, given a 16-bit random number, an attacker can recover the internal state of NLFSR6 with probability 2^{-24} after getting 80 bits of the sequence $\mathbf{s}$. To obtain the next 16-bit random number from the given one, the adversary needs to know the next consecutive 80 bits of the sequence $\mathbf{s}$ and the internal state of NLFSR6. The 80 bits can be obtained either by guessing or obtaining about $\frac{2^{43.39}}{5} = 2^{41.07}$ consecutive random numbers. Due to the high linear span of the sequence $\mathbf{s}$, it is impossible to generate the next consecutive 80 bits from previous known 80 bits in practice. Furthermore, it is also difficult for an adversary to intercept $2^{41.07}$ consecutive random numbers in one protocol sessions because the communication session in RFID systems is usually quite short and the IV is different. Moreover, the secret seed can also be updated for different sessions. Hence, the attacker can guess the next 16-bit random number with the better probability 2^{-16}, which is much less than 0.025% as specified in the EPC C1 Gen2 standard.

To measure the linear dependency between an n-bit output and the previous n-bit output, we performed a serial correlation test [65] on the sequences generated by the PRNG. We generated 18 distinct sequences for different initial values of the NLFSRs, each one is of size either 2^{25} or 2^{26} bytes and calculated the serial correlation coefficient for 1-bit, 1-byte and 2-byte lag. Our experimental results demonstrate that the serial correlation coefficients are close to zero, which indicates the good randomness of the generated sequences. The serial correlation coefficients

for different sequences are given in Table 8.3b.

Table 8.3: Successful fulfillment of the requirements of the EPC C1 Gen2 standard

(a) The first requirement

Sequences	**Upper**	**Lower**
S1	1.0637	0.9399
S2	1.0666	0.9289
S3	1.0664	0.9282
S4	1.0637	0.9333
S5	1.0598	0.9396
S6	1.0644	0.9404
S7	1.0693	0.9387
S8	1.0673	0.9335
S9	1.0981	0.9033
S10	1.0971	0.9130
S11	1.0855	0.8769
S12	1.0693	0.9372
S13	1.0467	0.9555
S14	1.0472	0.9547
S15	1.0502	0.9563
S16	1.0442	0.9537
S17	1.0442	0.9478
S18	1.0455	0.9547

(b) The third requirement

Sequences	**1-bit**	**1-byte**	**2-byte**
S1′	-0.000187	-0.000109	-0.000109
S2′	-0.000095	-0.000166	-0.000095
S3′	-0.000233	-0.000144	-0.000001
S4′	-0.000188	-0.000188	-0.000188
S5′	-0.000017	-0.000061	-0.000012
S6′	-0.000115	0.000025	0.000013
S7′	0.000209	-0.000178	-0.000219
S8′	0.000127	0.000097	0.000046
S9′	0.000040	0.000052	0.000244
S10′	0.000021	-0.000038	0.000074
S11′	-0.000006	0.000162	0.000010
S12′	0.000183	-0.000155	-0.000122
S13′	-0.000198	0.000019	-0.000057
S14′	-0.000097	-0.000074	-0.000262
S15′	0.000226	-0.000000	-0.000255
S16′	0.000069	0.000035	0.000125
S17′	-0.000203	-0.000203	-0.000203
S18′	-0.000085	0.000039	0.000094

NIST Statistical Test Results

Different from the statistical tests in the EPC C1 Gen2 standard, the NIST test suite contains 15 demanding statistical tests for characterizing the randomness of a binary sequence. According to the NIST specification [103], a PRNG passes the test suite successfully if it passes all the tests simultaneously with a proportion of 96%. In our experiment, 10 test sequence (TS) sets are generated, each of which has 100 different sequences with different seeds and each sequence has a length of 2^{25}. We computed the proportion values for each TS set and listed the test results[1]

[1]Non-overlapping template matching test results are not given in Table 8.4 because of 148 entries. However, Warbler-II has passed the test successfully.

for 5 TS sets in Table 8.4. It is not difficult to find out that each TS set can pass the NIST test suite successfully.

Table 8.4: NIST test suite results of Warbler-II

Tests	TS1	TS2	TS3	TS4	TS5
	proportion	proportion	proportion	proportion	proportion
Frequency	1.00	0.99	0.99	1.00	0.97
Block-frequency	1.00	0.99	0.97	1.00	0.99
Cumulative-sum	1.00, 1.00	1.00, 0.99	0.98, 0.99	1.00, 0.99	0.98, 0.97
Runs	0.99	1.00	1.00	1.00	1.00
Longest-run	0.99	0.99	0.98	0.98	1.00
Rank	0.99	0.99	1.00	0.96	1.00
DFT	0.98	0.98	1.00	0.99	0.99
Overlapping-templates	0.99	0.97	0.96	0.98	0.96
Universal-stat.	0.99	0.98	0.99	1.00	1.00
Approx. entropy	1.00	1.00	0.98	1.00	1.00
Serial	0.99, 0.99	0.99, 1.00	0.98, 0.98	0.99, 1.00	0.99, 0.99
Linear-complexity	0.97	1.00	0.99	1.00	1.00
Random-excursions	0.98, 0.99	1.00, 0.99	1.00, 0.99	0.99, 0.99	0.99, 0.99
	0.99, 1.00	0.96, 0.99	0.99, 0.99	0.98, 1.00	0.99, 0.99
	0.99, 1.00	1.00, 1.00	0.99, 1.00	0.99, 0.99	1.00, 1.00
	0.99, 1.00	0.99, 0.99	1.00, 1.00	0.99, 0.98	1.00, 0.92
Random-excur-variant	1.00, 1.00, 1.00	1.00, 1.00, 1.00	0.99, 0.99, 0.99	0.99, 1.00, 1.00	0.98, 0.98, 0.98
	1.00, 1.00, 1.00	1.00, 1.00, 1.00	0.99, 0.99, 1.00	1.00, 1.00, 1.00	0.97, 0.97, 0.98
	1.00, 0.99, 0.98	1.00, 1.00, 0.99	1.00, 1.00, 1.00	1.00, 0.99, 0.99	0.99, 1.00, 1.00
	1.00, 0.99, 1.00	1.00, 1.00, 1.00	0.99, 0.99, 0.99	0.97, 0.98, 0.99	0.99, 1.00, 0.99
	1.00, 1.00, 1.00	1.00, 1.00, 1.00	1.00, 1.00, 1.00	1.00, 1.00, 1.00	1.00, 1.00, 1.00
	1.00, 1.00, 1.00	0.97, 0.99, 0.98	1.00, 1.00, 1.00	1.00, 1.00, 1.00	1.00, 1.00, 1.00

8.2.2 Cryptanalysis of Warbler-II

In this section, we perform a detailed cryptanalysis against Warbler-II PRNG by considering algebraic attacks, cube attacks, time-memory-data-tradeoff attacks, the attacks proposed by Mihaljević *et al.* and weak internal states and fault injection attacks. We also argue that some attacks such as correlation attacks and distinguishing attacks cannot be applied to our PRNG.

Resistance against Algebraic Attacks

Warbler family was designed to resist algebraic attacks [20]. In our PRNG design, nonlinear feedback functions are used to update the internal states of different

NLFSRs, and the output bits are filtered by the WG transformation. For an Warbler instance, the total length of the internal state is $L = (\sum_{i=1}^{m} l_i + l \cdot n)$ where K-bit is reserved for the key and $(L - K)$-bit is for the initial vector. We remember that Warbler family fully exploits NLFSRs in the design. According to the algebraic attack technique, an Warbler instance can be reduced to a system of linear equations with 2^K unknowns while it is assumed that the initial vector is known, and the system of linear equations can be solved by approximately $\frac{7}{64} \cdot (2^K)^{\log_2 7}$ operations. In particular, the total length of the internal state of Warbler-II PRNG is 92-bit and the length of the secret key is 60-bit, one can reduce the PRNG to a system of linear equations with about 2^{60} unknown variables, which can be solved by approximately $\frac{7}{64} \cdot (2^{60})^{\log_2 7}$ operations. We have chosen the feedback functions of the maximum algebraic degree for NLFSR22, NLFSR21 and NLFSR6. As a result, the algebraic degree of the system will grow rapidly during the initialization phase as well as the running phase. Thus, the algebraic attack is not better than the exhaustive search in this case.

Resistance against Cube Attacks

Cube attack [24] is a generic key-recovery attack that can be applied to any cryptosystem, provided that the attacker can obtain a bit of information that can be represented by a low-degree decomposition multivariate polynomial in Algebraic Normal Form of the secret and public variables of the target cryptosystem. According to the cube attack, our PRNG can be regarded as a system of multivariate polynomials $p(k_1, ..., k_{60}, v_1, v_2..., v_{32})$ with public IV variables $v_1, v_2, ..., v_{32}$ and secret key variables $k_1, k_2, \ldots, k_{60}$. The polynomial

$$p(k_1, ..., k_{60}, v_1, v_2..., v_{32}) = t_I \cdot p_{S(I)} + q(k_1, ..., k_{60}, v_1, v_2..., v_{32})$$

is called a *master* polynomial, where $t_I = v_{i_1} v_{i_2} \cdots v_{i_k}$ is a monomial with $\{i_1, i_2, ..., i_k\} \subseteq \{1, 2, ..., 32\}$ and $p_{S(I)}$ is called a *superpoly* of t_I in p. The term t_I is called a *maxterm* if $\deg(p_{S(I)}) = 1$. We implemented the cube attack against Warbler-II in CUDA and exploited the power of a GPU (i.e, a Tesla C2070 from NVIDIA) for

accelerating the computation significantly. We took the first output bit after the 44-round initialization phase in order to find the maxterms in the master polynomial and performed an exhaustive search over all possible cube dimensions ranging from 1 to 32. Our PRNG was run on the GPU for around 85 days to exhaust all initial vectors, but we did not find any linear and quadratic superpoly equations for different cube dimensions.

Resistance against Time-Memory-Data Tradeoff Attacks

Time-memory-data tradeoff attack is a generic cryptanalytic attack which can be applied to any cipher. In a stream cipher, the complexity of a time-memory-data tradeoff attack depends solely on the length of the internal state, which is given by $O(2^{\frac{n}{2}})$, where n is the length of the internal state [6]. In particular, the complexity of the time-memory-data tradeoff attack against Warbler family is lower bounded by $O(2^{\frac{L}{2}})$ where L is the total length of the internal state of Warbler. We note that a stream cipher with low sampling resistance is vulnerable to a more flexible time-memory-data tradeoff attack. In our PRNG, the WG transformation $WG(x^3)$ is the filtering function and the number of terms in the algebraic normal form representation of $WG(x^3)$ is 15, among which only two terms are linear and the remaining terms are either quadratic or cubic. Only by fixing four input variables in the WG transformation, one can obtain a linear function in one variable. Thus, the sampling resistance of the proposed PRNG is high. Since the length of the internal state is 92-bit in Warbler-II, the expected complexity of the time-memory-data tradeoffs attack is bounded below by 2^{46}.

Resistance against Mihaljević *et al.*'s Attacks

Recently Mihaljević *et al.* [89] proposed an attack on Grain v1 to recovered an internal state by exploiting the normality of the filtering function used in the stream cipher. As the architecture of our PRNG is similar to Grain v1, it seems natural that the attack can be applied to our PRNG. The key idea of the attack is to determine the bias of the internal state using the normality of the filtering function and then apply the strategy of the generic time-memory-data tradeoff attack. For

details of the attack, we refer the reader to [89]. Note that the WG transformation $WG(x^3)$ has the lowest normality which is equal to 1, whereas the filtering function of stream cipher Grain v1 has normality 2. In the attack, we assume that the attacker can collect a set of D sequences and the length of each sequence is S and the attacker is observed the pattern of occurring the 16-bit random number zero. In this sample, the expected number of random number zero is $2^{log_2 D+log_2 S-16}$ as the length of each random number is equal to 16. We also assume that the attacker does not know any internal state bits of the PRNG. But, in Grain v1, the attacker can recover 18 internal state bits of the NLFSR using some bits of the LFSR and NLFSR states. However, such a trick cannot be applied to Warbler-II. Then, the following is the complexity of the attack for recovering an internal state of the PRNG.

Table 8.5: The processing and pre-processing attack complexities

Required samples	Time complexity of processing	Pre-processing time and space complexities
$D = 2^{37}$, $S = 2^{43}$	$2^{4\times16} = 2^{64}$	$2^{L-(4\times16)} = 2^{28}$, $2^{L-(5\times16)} = 2^{12}$
$D = 2^{39}$, $S = 2^{41}$	$2^{4\times16} = 2^{64}$	$2^{L-(4\times16)} = 2^{28}$, $2^{L-(5\times16)} = 2^{12}$
$D = 2^{46}$, $S = 2^{35}$	$2^{4\times16} = 2^{64}$	$2^{L-(4\times16)} = 2^{28}$, $2^{L-(5\times16)} = 2^{12}$

In the above table, we can observe that the time complexity of the processing phase is much greater than the time complexity at the pre-precessing phase, and an attacker requires a huge amount of sample to launch the attack. For lightweight applications, it is impossible to collect that amount of data. Thus, the attack cannot be a practical attack on our PRNG.

Weak Internal States and Fault Injection

In Warbler-II, the CMDB contains three NLFSRs namely NLFSR22, NLFSR21 and NLFSR19 of lengths 22, 21, and 19, respectively. For the initial state all-zero $\mathbf{0} = (0, 0, ..., 0)$, NLFSR22 generates the zero sequence, and for initial state all-one $\mathbf{1} = (1, 1, ..., 1)$, NLFSR21 and NLFSR19 generate the all-one sequence. For any other initial states, the output sequence from the CMDB is a nonzero sequence. We call

initial states **0** and **1** *weak initial states* of respective NLFSRs, which are vulnerable only for the running phase of the PRNG, not for the initialization phase. For the combining function $f(x_0, x_1, x_2) = x_0x_1 + x_1x_2 + x_0x_2 + x_0 + x_1$, the sequence **s** is a zero sequence for the following initial states of NLFSRs: a) when the initial state of NLFSR22 is **0** and initial states of NLFSR21 and NLFSR19 are **1**; b) when the initial state of NLFSR22 is **0** and the initial state of NLFSR21 is **1**. The scenario (a) can occur with probability 2^{-62} and the scenario (b) can occur with probability 2^{-43}. Note that the period and linear complexity of the output sequence can be reduced by setting the initial states of the NLFSRs to weak internal states. However, the above scenarios can be detected and avoided by adding an OR gate to NLFSR22 and two NAND gates to NLFSR21 and NLFSR19 and then adding the final outputs of OR and NAND gates to the finite state machine (FSM). An overview of the PRNG after adding OR and NANG gates is provided in Figure 8.2. Based on the outputs of OR and NAND gates, the FSM preforms an action such as update the seed or apply the initialization round again. In other words, when any of NLFSRs in the CMDB contains a weak initial state after the initialization phase, the seed of the PRNG is updated, followed by the initialization round or only the initialization phase is applied again. On the other hand, an attacker might inject faults to the internal states of three NLFSRs and set the initial states to the weak initial states. In the best case, injecting faults to produce the sequence **s** to be a zero sequence, an attacker needs to have a complete knowledge of internal states of NLFSR22 and NLFSR21. If the attacker is successful in setting the internal states to weak internal states, according to the above strategy, the seed of the PRNG will be updated and the initilization round will be applied. Hence, the fault injection can be prevented.

Other Cryptanalytic Attacks

In the fast correlation attacks [84], the internal state of an LFSR based stream cipher can be recovered by first determining a system of linear equations according to a statistical model and then solving the system of linear equations. In Warbler-II, the internal state is updated in a nonlinear way. Thus it is hard for an attacker to

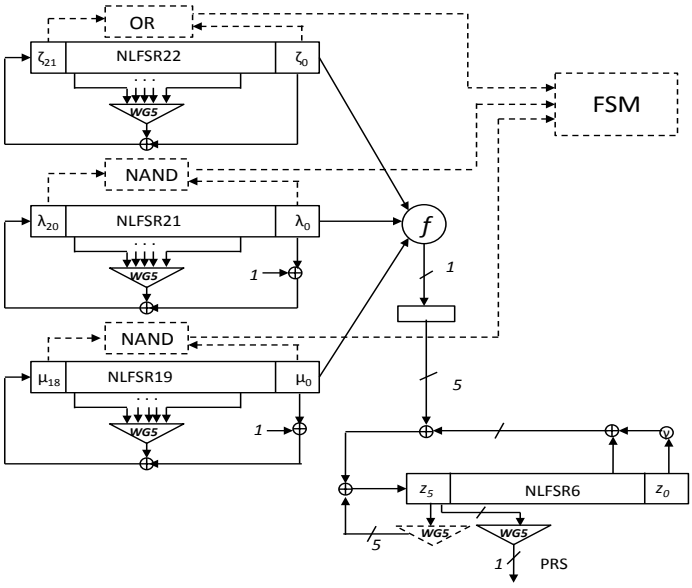

Figure 8.2: Warbler-II after adding the control circuit

decide such a system of (non-)linear equations according to some statistical models.

For an LFSR-based stream cipher, the DFT attacks [46] can be applied when the exact linear complexity of the output sequence and enough consecutive output bits are known. In Warbler-II, the exact linear complexity of the output sequence is not known. Moreover, the period of a sequence is dependent on the initial state of the PRNG. Therefore, the DFT attacks cannot be applied to our PRNG. Furthermore, for lightweight applications such as the EPC C1 Gen2 standard protocol, it is hard for an attacker to obtain enough consecutive bits.

A chosen IV attack on the original version of WG cipher was presented in [110], where one can distinguish several bits of the output sequence by building a distinguisher based on differential cryptanalysis. In our PRNG, two nonlinear terms, t_k, w_k (i.e., an output from the WG transformation as well as a 5-bit tuple generated by the first building block) are added to the recurrence relation. Furthermore, the WG permutation is not added in the recurrence relation, only w_k is added. Thus the differentials after 44 rounds of the initialization phase will contain most internal state bits. As a result, it would be hard for an attacker to distinguish output bits generated by the Warbler-II PRNG.

8.3 Hardware Implementation and Comparisons

Warbler-II is implemented in VHDL for the low-cost Spartan-3 XC3S50 (Package PQ208 with speed grade -5) FPGA device from Xilinx for measuring the hardware complexity. The hardware implementation shows that the PRNG core totally occupies about 58 slices (19 and 39 slices for the CMDB and the NFWGG, respectively) on the target FPGA device and achieves a throughout of 45 Mbps. For the details of the hardware implementation, we refer the reader to [75].

Table 8.6 presents a comparison with other PRNGs in terms of hardware implementation and achieved randomness properties. We notice that Warbler-II requires 12 more slices, compared to the PRNG Warbler-I, but Warbler-II provides a better security level. If we compare the hardware implementation cost for Warbler-II with Grain-128 and Trivium, our design needs 10 and 17 more slices, respectively [9]. Randomness properties period and linear complexity of Warbler-II are bounded below by $2^{64.32}$ and $2^{43.39}$, respectively, but the period and linear complexity of keystreams of Trivium are not guaranteed. Warbler-II can easily be converted to a sponge-based PRNG architecture like U-QUARK architecture [3]. However, it is hard to promise the randomness properties of output sequences produced by the sponge-based structure.

Table 8.6: A comparison with other PRNGs

Functions	Size of the internal state	Area	Device	Randomness Properties	
			μm	Period	LS
Warbler-II	92	58 Slices	XC3S50-PQ208	$\geq 2^{64.32}$	$\geq 2^{43.39}$
Warbler-I [72]	65	46 Slices/760 GE (est.)	XC3S50-PQ208	$\geq 2^{37.32}$	$\geq 2^{18.58}$
LAMED [97]	64	1585 GE (est.)	–	–	–
Melia-Segui *et al.* [85]	16	761 GE (est.)	–	–	–
Grain-128 [9]	256	48 Slices	Virtex-II	$\geq 2^{128}$	–
Trivium [9]	288	41 Slices	Virtex-II	–	–
U-QUARK [3]	136	1379 GE	0.18 μm CMOS	–	–
KECCAK [61]	128	1300 GE	0.13 μm CMOS	–	–
PHOTON-80/20/16 [52]	100	865 GE	0.18 μm CMOS	–	–
SPONGENT-80 [7]	88	738 GE	0.13 μm CMOS	–	–

8.4 Application to the RFID Tags and Protocols

Warbler-II is designed for the resource-constrained environments such as RFID tags. Warbler-II can be used to generate random numbers in the automatic identification

protocols and authentication protocols, for example the Flyweight RFID protocol [10]. Several lightweight authentication protocols for RFID systems can be found in [112]. The authentication process requires 3 to 5 random numbers, and the security of the protocol depends on the on-chip random number generator. Since an output sequence produced by Warbler-II has good randomness properties such as period at least $2^{64.32}$ and linear complexity at least $2^{43.39}$, Warbler-II can be used as a keystream generator in a stream cipher for lightweight applications.

When Warbler-II is used in the automatic tag identification protocol of the EPC C1 Gen2 standard, Warbler-II totally requires 146 clock cycles, including 22 clock cycles for loading the key and IV into the registers, 44 clock cycles for the initialization phase, and 80 clock cycles for generating the first 16-bit random number. After that, each 16-bit random number can be obtained in every 80 clock cycles. Assuming that the EPC tags run at the clock frequency of 100 KHz and two 16-bit random numbers are needed for the tag identification protocol according to the EPC C1 Gen2 standard, one can identify about 443 tags in one second by using the proposed lightweight PRNG.

8.5 Comparisons with Other PRNGs

In this section, we provide a comparison between Warbler-II and Warbler-I, a composited de Bruijn sequence and the WG-5 stream cipher.

8.5.1 Comparison with Warbler-I

The main aim of designing Warbler-II is to offer a better security level, compared to Warbler-I. Designs of both Warbler-I and Warbler-II are similar. The CMDB of Warbler-II contains three primitive NLFSRs of lengths 19, 21 and 22, and the CMDB of Warbler-I contains two primitive NLFSRs of lengths 17 and 18, but the nonlinear feedback WG generators over $\mathbb{F}_{2^5}$ for both cases are the same except for the feedback function. In the NFWGG of Warbler-I, $WG(x)$ is used as a feedback function, whereas in the NFWGG of Warbler-II, $WG(x^{11})$ is used as a feedback function. Warbler-II produces output sequences with period at least $2^{64.32}$ and

linear complexity at least $2^{43.39}$, whereas Warbler-I produces output sequences with period at least $2^{37.32}$ and linear complexity at least $2^{18.58}$. The attack resistant properties of Warbler-II are much better than the attack resistant properties of Warbler-I. The hardware implementation of Warbler-I requires 46 slices, but the hardware implementation of Warbler-II requires 12 more slices. We restrict the application of Warbler-I to the EPC C1 Gen2 RFID tags, but Warbler-II can be used in the EPC C1 Gen2 tags as well as RFID applications where more security level is required.

8.5.2 Comparisons with a Composited De Bruijn Sequence and WG-5 Stream Cipher

If we compare Warbler-II with a composited de Bruijn sequence of period 2^{92}, then the composited de Bruijn sequence can generate all 92-tuples exactly once in a period, but Warbler-II cannot generate all 92-tuples exactly once. Warbler-II generates different sequences with period a multiple of $2^{64.32}$ for different initial states, but the composited de Bruijn sequence is of period 2^{92} for all initial states. We note that the composited de Bruijn sequence is generated by an NLFSR with direct feedback. On the other hand, the period of an output sequence produced by Warbler-II is controlled by the CMDB that is composed of three primitive NLFSRs. The feedback function of a composited de Bruijn sequence would contain a number of product-of-sum terms, as a result, the generation of the composited de Bruijn sequence would be cost effective compared to Warbler-II.

WG-5 stream cipher [1] is a filtering generator based on an LFSR over $\mathbb{F}_{2^5}$ and a WG-5 transformation. The internal state of the WG-5 stream cipher is updated using a linear function that generates an m-sequence, and the output sequence is obtained by filtering the m-sequence through the nonlinear WG-5 transformation. Contrariwise, the internal states of Warbler-II are updated using nonlinear functions and the output sequence is produced by filtering the sequence of the NFWGG through the WG-5 transformation. For the WG-5 cipher, the m-sequence guarantees the period of an output sequence, but for Warbler-II, the primitive NLFSRs of the

CMDB promise the lower bound of the period of an output sequence. Due to the nonlinear internal state update, Warbler-II has better attack resistance properties compared to the WG-5 stream cipher.

8.6 Summary of Chapter 8

This chapter presented Warbler-II, a new lightweight pseudorandom number generator based on nonlinear feedback shift registers with desirable randomness properties. We provided a detailed mathematical description of Warbler-II including its mode of operations. We performed a security analysis of Warbler-II in two steps. First, we performed the statistical tests on the sequences generated by the PRNG specified by the EPC C1 Gen2 standard and the NIST standard. Our PRNG passed all the statistical tests. We then characterized our PRNG by applying algebraic attacks, cube attacks, time-memory-data tradeoff attacks, Mihaljević *et al.*'s attacks and weak initial states and fault injection attacks. A hardware implementation of Warbler-II in VHDL for the low-cost Spartan-3 XC3S50 FPGA device shows that Warbler-II can be implemented using about 58 slices. Warbler-II can be employed as a random number generator in the automatic tag identification protocol as well as the authentication protocols for RFID systems.

Chapter 9

Conclusions and Future Research

In this chapter, we summarize the research contributions of this thesis, and present the future research directions related to the subjects therein. The main contributions of each chapter are presented.

9.1 Conclusions

In this thesis, we concentrated on the design and analysis of cryptographically strong pseudorandom sequence and number generators. Specifically, we focused on the generation of de Bruijn sequences and span n sequences, which have good randomness properties such as maximum period, balance, and high linear complexity, and which are suitable for cryptographic applications. We fully exploited nonlinear feedback shift registers for generating de Bruijn sequences and span n sequences, and for designing random number generators.

We first studied the generation span n sequences using nonlinear feedback shift registers whose feedback functions are composed of a permutation and a trace function over a finite field, a decimation number, and a t-tap position. Considering these parameters, a class of feedback functions in an NLFSR is formed and a number of span n sequences are produced. The span n sequence generation by this technique is called the structured search. In the structured search, we used WG transformations, three-term functions, five-term functions, monomial functions with Kasami

exponent, and MCM functions as nonlinear feedback functions, and presented the number of span n sequences produced by each class of functions for $6 \leq n \leq 20$. We study the linear span or complexity of new span n sequences. The linear complexity of a span n sequence lies in the range of $(2^n - 2 - 3n)$ and $(2^n - 2)$. The success probability of obtaining a span n sequence in the structured search is empirically compared with the success probability of obtaining a span n sequence in a random generation method. The comparison showed that one can obtain a span n sequence with optimal or near-optimal linear complexity in the structured search with a better success probability. New span n sequences or span n sequences generated by the structured search can be used to design lightweight pseudorandom number generators and stream ciphers. Moreover, they can be used in the composition method to generate long de Bruijn sequences.

We first refined the composition method so that we could generate long de Bruijn sequences, and then determined the linear complexity of a composited de Bruijn sequence. We conducted an analysis of a composited nonlinear feedback function that generates a de Bruijn sequence. In the analysis, we studied an approximation of the feedback function by setting some product terms as constant functions. The cycle structure of an approximated feedback function and the linear complexity of a sequence generated by an approximated feedback function are determined. Our analysis also indicated that a composited de Bruijn sequence can be cryptographically strong if the starting span n sequence is of long period and optimal linear complexity. Moreover, we presented a few example of de Bruijn sequences of periods in the range of 2^{35} and 2^{40} with their algebraic forms. Furthermore, the implementation issues of a feedback function of a composited NLFSR are taken into consideration.

We proposed a new pseudorandom number generator family, named Warbler family for resource-constrained smart devices such as RFID tags. Warbler family is a purely NLFSR-based PRNG family with desirable randomness properties. Warbler family is composed of two building blocks, namely a combination of modified de Bruijn blocks (CMDB) and a nonlinear feedback WG generator (NFWGG). The combination of modified de Bruijn blocks consists of a number of primitive NLFSRs.

The nonlinear feedback WG generator contains an NLFSR over an extension field and two WG transformation modules used for the feedback as well as filtering purpose. Randomness properties of an output sequence produced by the Warbler family are derived, followed by a description of the initilization and running phases of Warbler family. Some parameter selection criteria for an instance of the Warbler family are proposed to offer the best security level against known attacks.

We presented an instance, Warbler-I, of the Warbler family for the EPC Class-1 Generation-2 passive RFID tags. Considering the high power-consumption, large area and low throughput of TRNGs, we replace the TRNG used in previous works by a PRNG with good statistical properties. Warbler-I can generate sequences with good randomness properties such as period at least $2^{37.32}$ and linear span at least $2^{18.58}$. In our design, the pseudorandom sequences are generated using nonlinear feedback shift registers. The statistical tests specified by the EPC C1 Gen2 and NIST standards, algebraic attacks, cube attacks and time-memory-data tradeoff attacks are employed to characterize the security properties of the proposed PRNG. A comparison with the sponge-based PRNGs is also conducted. In addition, an FPGA implementation shows that Warbler-I can be implemented using 46 slices and can generate a 16-bit random number every 80 clock cycles after an initialization process of 36 clock cycles. Warbler-I perfectly satisfies the requirements of the EPC C1 Gen2 standard and hence a suitable candidate for the EPC C1 Gen2 standard.

We proposed another instance, Warbler-II, of the Warbler family, which is designed for providing a better security level compared to Warbler-I. We described Warbler-II with its mathematical functions in detail. The CMDB of Warbler-II contains three primitive NLFSRs. The period and linear complexity of an output sequence produced by Warbler-II are at least $2^{64.32}$ and $2^{43.38}$, respectively. We performed a detailed security analysis of Warbler-II in two steps. First, we performed the statistical tests on the sequences generated by Warbler-II specified by the EPC C1 Gen2 standard and the NIST standard. Our PRNG passed all the statistical tests. Then, we characterized Warbler-II by applying algebraic attacks, cube attacks, time-memory-data tradeoff attacks, Mihaljević *et al.*'s attacks, and weak initial states and fault injection attacks. A hardware implementation of Warbler-II

in VHDL for the low-cost Spartan-3 XC3S50 FPGA device shows that Warbler-II can be implemented using about 58 slices. Warbler-II can be used as a random number generator in the automatic tag identification and authentication protocols for RFID systems.

9.2 Future Research

Nonlinear feedback shift registers are of great importance, especially in resource-constrained environments such as RFID tags and sensor networks due to their efficient hardware implementation. Well-designed NLFSR-based cryptographic primitives such as stream ciphers and PRNGs can provide good randomness in communication systems and would be resistant to the known cryptanalytic attacks because of the hardness of solving a system of nonlinear equations over the binary field. Only limited results exist in the theory of nonlinear feedback shift registers. In this section, we provide the reader with an overview of various interesting research directions where the future research can be conducted.

On Generation of Span n Sequences

A primitive NLFSR or an NLFSR that generates a span n sequence is an important component in a keystream generator, since it ensures the periodicity of a keystream. Span n sequences are also fundamental building blocks of a keystream generator like a combinatorial generator. In Chapter 4 we dealt with the generation of good span n sequences using NLFSRs. The current technique of checking the primitivity of a nonlinear feedback function is the exhaustive search algorithm whose time complexity is exponential in n. For a large value of n, it is impossible to verify the primitivity of a nonlinear feedback function in real time. A sub-exponential or polynomial time algorithm for testing the primitivity of some special nonlinear feedback function needs to be developed in order to design NLFSR-based stream ciphers and PRNGs with high level security and guaranteed randomness properties.

For a fixed n and t of the recurrence relations, the numbers of span n sequences for different permutations are bounded by certain numbers, and the upper and

lower bounds mainly depend on n and t. The problem of determining the upper and lower bounds of the number of span n sequences for a fixed n and t is interesting from a theoretical point of view. It seems to be a difficult problem when there is no polynomial algorithm for testing the primitivity of a nonlinear feedback function.

We have observed that, in the structured search, many span n sequences have the same t-tap position, primitive polynomial and the same linear complexity, but their decimation numbers are different. The problem of determining the relation between the decimation numbers of those span n sequences is significant. If such a relation is established, then from one span n sequence, many other span n sequences of the same length can be produced.

On the Composited Construction and de Bruijn Sequences

The efficient construction of long de Bruijn sequences is a challenging problem. For cryptographic applications, a de Bruijn sequence must be strong (long period and high linear complexity) and should have an efficient implementation. The composited construction based on a span n sequence is the only known construction in the literature for generating long de Bruijn sequences. The feedback function of a composited de Bruijn sequence contains a number of product-of-sum terms for which the evaluation of the feedback function becomes cost effective. If an efficient hardware implementation of composited feedback functions is found, then they can be used in RFID tags for generating random numbers with desirable randomness properties. Therefore, an efficient hardware implementation of the product-of-sum terms is crucial for the use of composited de Bruijn sequences in resource-constrained environments.

Games's generalized construction where a de Bruijn sequence is constructed from two different de Bruijn sequences can be written as the composited construction. As a future research, Games's generalized construction can be investigated from the composited construction point of view.

On Warbler-I and Warbler-II PRNGs

Warbler-I and Warbler-II are two hardware-based pseudorandom number generators designed for RFID tags. In Warbler-II, when any of the internal states of three NLFSRs in the CMDB is at a weak internal state, we either apply an initialization round or update the seed, followed by an initialization round, and this is controlled by the finite state machine. The side channel attack is a cryptanalytic attack which exploits information from the physical implementation of a primitive. Our PRNGs can be analyzed from the side channel attack point of view to examine its attack resistance property.

Since Warbler-I and Warbler-II are based on WG-5 transformations, Warbler-I and Warbler-II require five clock cycles to output one bit. To optimize the number of clock cycles required for outputting one bit, a Galois configuration of the NLFSRs in the CMDB can be found as a future work. As a result, Warbler-I and Warbler-II can produce one bit in one clock cycle.

Appendix A

Span n Sequences and Linear Complexity Bounds

A.1 Example of Span n Sequences

In this section, we present some examples of span n sequences produced by recurrence relation (4.1) defined in Chapter 4 using WG transformations, 5-term functions, 3-term functions, monomial functions with Kasami exponent, and MCM functions. We represented the span n sequences by three parameters, namely decimation number d, primitive polynomial $p(x)$ over $\mathbb{F}_2$ and a t-tap position. Tables A.1 and A.2 contain all span n sequences produced by WG transformations over $\mathbb{F}_{2^5}$ for different lengths of NLFSRs. Table A.3 contains span n sequences produced by WG transformations over $\mathbb{F}_{2^7}$, where span n sequences for different lengths of the NLFSRs are provided. Tables A.4 - A.7 present some instances of span n sequences produced by 5-term functions, 3-term functions, monomial functions with Kasami exponents, and MCM functions, respectively.

Table A.1: WG span n sequences generated using rec. rel. (4.1)

n	Decimation d	Polynomial $(c_0, c_1, c_2, c_3, c_4)$	Tap position $(r_1, r_2, r_3, r_4, r_5)$
8	1	1 0 1 0 0	1 2 4 5 7
	1	1 1 1 1 0	1 3 4 5 6
	1	1 1 1 1 0	2 4 5 6 7
	3	1 1 0 1 1	1 2 3 5 6
	7	1 0 1 1 1	1 2 3 5 7
	7	1 0 1 0 0	2 3 4 6 7
	15	1 1 1 1 0	2 3 4 6 7
9	1	1 1 1 0 1	1 2 5 6 8
	1	1 1 1 0 1	1 3 6 7 8
	1	1 1 1 1 0	2 3 5 7 8
	1	1 1 1 0 1	4 5 6 7 8
	3	1 1 0 1 1	1 2 4 5 6
	3	1 0 1 0 0	1 2 4 5 8
	3	1 0 1 0 0	2 4 6 7 8
	7	1 0 1 0 0	1 2 3 4 6
	11	1 1 1 0 1	1 4 6 7 8
	11	1 1 1 1 0	2 4 5 6 7
	11	1 1 1 1 0	2 4 5 6 8
	11	1 1 1 0 1	2 4 6 7 8
	15	1 1 1 1 0	1 2 3 4 6
	15	1 1 1 0 1	1 2 5 7 8
10	1	1 1 0 1 1	1 2 4 5 8
	1	1 1 1 0 1	1 3 4 6 7
	1	1 1 1 0 1	1 3 4 6 9
	3	1 1 0 1 1	1 2 3 4 8
	7	1 0 0 1 0	1 2 4 7 8
	11	1 0 1 1 1	1 2 3 4 5
	11	1 0 0 1 0	1 2 3 7 8
	11	1 1 1 1 0	1 4 5 8 9
11	1	1 1 1 0 1	1 2 7 8 10
	1	1 1 1 1 0	3 4 5 8 10
	1	1 1 1 0 1	6 7 8 9 10
	7	1 0 1 1 1	1 2 3 6 7
	7	1 0 0 1 0	1 3 7 8 10
	7	1 0 1 1 1	2 3 4 7 10
	7	1 1 0 1 1	2 3 7 9 10
	7	1 0 0 1 0	2 4 5 6 10
	7	1 1 0 1 1	3 4 5 8 9
	11	1 1 1 1 0	1 2 4 5 8
	11	1 1 1 0 1	1 3 4 6 10
12	1	1 1 1 1 0	2 3 4 5 6
	1	1 0 1 0 0	2 3 4 5 8
	1	1 1 1 0 1	2 3 5 7 9
	1	1 0 1 0 0	2 3 6 9 10
	1	1 1 1 0 1	4 6 9 10 11
	3	1 1 0 1 1	1 2 3 4 5
	3	1 1 0 1 1	2 5 7 8 10
	3	1 0 1 0 0	4 5 6 9 11
	7	1 0 1 0 0	1 2 4 7 8
	7	1 1 0 1 1	1 2 5 6 8
	11	1 0 0 1 0	1 3 4 6 10
	11	1 1 1 0 1	1 3 4 9 11
	11	1 1 1 1 0	1 4 5 8 9
	11	1 1 1 0 1	2 3 6 7 10
	11	1 1 1 1 0	3 5 7 8 9
	11	1 1 1 1 0	4 6 7 9 10
	15	1 1 1 1 0	1 2 4 7 8

Table A.2: WG span n sequences generated using rec. rel. (4.1)

n	Decimation d	Polynomial $(c_0, c_1, c_2, c_3, c_4)$	Tap position $(r_1, r_2, r_3, r_4, r_5)$
13	1	1 0 1 0 0	1 3 4 5 9
	1	1 0 1 0 0	5 8 9 11 12
	3	1 1 0 1 1	5 6 10 11 12
	7	1 0 1 0 0	1 2 3 6 8
	7	1 1 0 1 1	3 5 7 10 12
	7	1 1 0 1 1	6 7 9 10 12
	11	1 0 0 1 0	1 2 3 5 10
	11	1 1 1 0 1	1 2 5 10 12
	11	1 1 1 0 1	1 5 6 10 12
	11	1 1 1 0 1	4 5 7 8 9
	15	1 1 1 1 0	1 2 3 6 8
14	1	1 0 1 0 0	1 3 5 7 9
	1	1 1 1 1 0	2 6 8 9 13
	1	1 1 1 0 1	3 4 6 8 10
	1	1 1 1 0 1	3 5 8 10 13
	3	1 1 0 1 1	1 8 10 11 13
	7	1 0 0 1 0	1 2 6 9 12
	7	1 0 0 1 0	1 3 10 12 13
	7	1 0 0 1 0	1 6 9 12 13
	7	1 0 1 0 0	3 5 7 8 9
	11	1 1 1 1 0	1 2 4 11 12
	11	1 1 1 1 0	1 2 9 10 11
	15	1 1 1 0 1	3 5 6 8 13
	15	1 1 1 1 0	3 5 7 8 9
15	1	1 1 1 0 1	4 5 12 13 14
	3	1 0 1 0 0	2 6 8 9 10
	3	1 0 1 0 0	4 5 6 7 14
	7	1 0 1 1 1	2 5 7 10 13
	7	1 0 1 1 1	2 5 8 11 14
	7	1 0 0 1 0	3 4 5 7 12
	11	1 0 0 1 0	2 3 6 7 13
	11	1 1 1 0 1	2 4 9 11 13
	11	1 0 1 1 1	2 9 10 11 12
	15	1 1 1 0 1	1 2 3 5 6
16	1	1 1 0 1 1	1 10 11 12 14
	1	1 1 1 0 1	1 10 11 12 14
	15	1 1 1 0 1	3 6 9 12 14
17	3	1 0 1 0 0	1 6 7 8 9
	3	1 1 0 1 1	4 7 8 9 12
	7	1 0 1 0 0	1 3 12 13 14
	7	1 1 0 1 1	1 4 10 11 13
	7	1 0 0 1 0	1 5 11 12 13
	11	1 1 1 0 1	1 3 6 12 13
	15	1 1 1 1 0	1 3 12 13 14
18	1	1 1 1 0 1	1 2 12 13 14
	3	1 1 0 1 1	4 7 8 10 15
	3	1 1 0 1 1	5 10 11 14 17
	7	1 0 0 1 0	1 2 5 7 11
	7	1 1 0 1 1	5 7 8 11 17
	11	1 0 0 1 0	1 8 9 11 15
	15	1 1 1 0 1	2 9 12 15 17
20	1	1 1 1 0 1	5 10 12 18 19

Table A.3: WG span n sequences for $t = 7$

Length n	Decimation d	Polynomial $(c_0, c_1, ..., c_5, c_6)$	t-tap position $(r_1, r_2, ..., r_6, r_7)$
8	5	$(1,1,0,0,0,0,0)$	$(1,2,3,4,5,6,7)$
9	1	$(1,0,1,1,1,1,1)$	$(1,2,3,4,5,6,7)$
10	27	$(1,1,1,1,0,1,1)$	$(1,2,3,4,5,6,7)$
11	1	$(1,1,1,1,0,1,1)$	$(1,2,3,5,8,9,10)$
12	1	$(1,0,1,1,1,0,0)$	$(1,2,4,5,8,10,11)$
13	9	$(1,1,0,0,1,0,1)$	$(1,2,3,4,5,6,8)$
14	43	$(1,1,1,0,1,1,1)$	$(1,2,3,4,5,6,7)$
15	31	$(1,1,0,0,0,0,0)$	$(1,2,3,4,7,12,14)$
16	27	$(1,1,1,1,0,1,1)$	$(1,2,3,5,6,8,14)$
17	1	$(1,0,1,1,1,0,0)$	$(1,2,3,4,7,9,13)$
18	1	$(1,0,1,1,1,0,0)$	$(1,2,3,4,6,9,16)$
19	3	$(1,1,1,1,1,1,0)$	$(1,2,3,5,7,15,17)$
20	31	$(1,1,1,1,1,1,0)$	$(1,2,3,7,8,12,15)$

Table A.4: 5-term span n sequences for $t = 7$

Length n	Decimation, d	Primitive polynomial $(c_0, c_1, ..., c_6)$	m-tap positions $(r_0, r_1, ..., r_6)$
8	13	$(1,1,0,0,0,0,0)$	$(1,2,3,4,5,6,7)$
9	5	$(1,1,0,0,0,0,0)$	$(1,2,3,4,5,6,7)$
10	43	$(1,1,0,0,1,0,1)$	$(1,2,3,4,5,6,7)$
11	7	$(1,1,1,0,0,1,0)$	$(1,2,3,4,5,6,8)$
12	9	$(1,0,1,0,1,0,1)$	$(1,2,3,4,5,6,7)$
13	47	$(1,1,1,0,0,1,0)$	$(1,2,3,4,5,6,10)$
14	63	$(1,0,0,0,1,1,1)$	$(1,2,3,4,5,7,9)$
15	63	$(1,0,1,1,1,0,0)$	$(1,2,3,4,5,9,13)$
16	47	$(1,1,0,0,1,0,1)$	$(1,2,3,4,5,6,7)$
17	31	$(1,1,1,1,0,0,0)$	$(1,2,3,4,9,14,16)$
18	5	$(1,0,0,1,1,1,0)$	$(1,2,3,4,5,11,17)$
19	5	$(1,0,1,1,1,0,0)$	$(1,2,3,6,7,10,18)$

Table A.5: 3-term span n sequences for $t = 7$

Length n	Decimation, d	Primitive polynomial $(c_0, c_1, ..., c_6)$	m-tap positions $(r_0, r_1, ..., r_6)$
8	31	$(1,1,0,1,0,1,0)$	$(1,2,3,4,5,6,7)$
9	21	$(1,0,0,1,0,0,0)$	$(1,2,3,4,5,6,7)$
10	55	$(1,0,0,1,1,1,0)$	$(1,2,3,4,5,6,7)$
11	13	$(1,1,1,1,1,1,0)$	$(1,2,3,4,5,6,7)$
12	11	$(1,0,0,0,0,0,1)$	$(1,2,3,4,5,7,11)$
13	31	$(1,1,1,1,1,1,0)$	$(1,2,3,4,5,6,9)$
14	55	$(1,0,1,1,1,1,1)$	$(1,2,3,4,5,6,12)$
15	43	$(1,0,1,0,0,1,1)$	$(1,2,3,4,5,6,14)$
16	3	$(1,0,0,0,1,0,0)$	$(1,2,3,4,5,12,15)$
17	63	$(1,1,1,1,0,0,0)$	$(1,2,3,4,10,12,14)$

Table A.6: Span n sequences generated by monomial functions for $t = 9$

Length n	Decimation, d	Primitive polynomial $(c_0, c_1, ..., c_6, c_7, c_8)$	m-tap positions $(r_0, r_1, ..., r_6, r_7, r_8)$
10	29	$(1,0,0,1,1,1,0,1,1)$	$(1,2,3,4,5,6,7,8,9)$
11	125	$(1,0,1,1,0,1,1,0,1)$	$(1,2,3,4,5,6,7,8,9)$
12	85	$(1,1,1,1,1,1,1,0,1)$	$(1,2,3,4,5,6,7,8,10)$
13	45	$(1,1,0,1,1,0,0,0,0)$	$(1,2,3,4,5,6,7,8,9)$
14	59	$(1,1,0,0,1,1,0,0,0)$	$(1,2,3,4,5,6,7,8,9)$
15	27	$(1,0,0,0,1,1,0,0,1)$	$(1,2,3,4,5,6,7,8,12)$
16	5	$(1,1,0,1,1,1,0,0,1)$	$(1,2,3,4,5,6,7,8,9)$

Table A.7: MCM span n sequences for $k = 3$ and $t = 7$

Length n	Decimation, d	Primitive polynomial $(c_0, c_1, ..., c_6)$	m-tap positions $(r_0, r_1, ..., r_6)$
8	19	$(1,1,0,0,0,0,0)$	$(1,2,3,4,5,6,7)$
9	1	$(1,1,1,1,0,0,0)$	$(1,2,3,4,5,6,7)$
10	21	$(1,0,1,1,1,1,1)$	$(1,2,3,4,5,7,8)$
11	5	$(1,1,1,1,0,1,1)$	$(1,2,3,4,5,6,8)$
12	55	$(1,0,0,0,1,1,1)$	$(1,2,3,4,5,6,9)$
13	19	$(1,0,0,0,0,0,1)$	$(1,2,3,4,5,7,10)$
14	9	$(1,0,1,1,1,0,0)$	$(1,2,3,4,7,10,12)$
15	23	$(1,1,0,0,0,0,0)$	$(1,2,3,4,5,7,12)$
16	23	$(1,0,1,0,1,0,1)$	$(1,2,3,4,6,7,9)$
17	13	$(1,0,0,0,0,0,1)$	$(1,2,3,4,7,15,16)$
18	3	$(1,1,1,0,0,1,0)$	$(1,2,3,4,7,12,14)$
19	27	$(1,0,1,1,1,1,1)$	$(7,9,14,15,16,17,18)$

A.2 Linear Complexity of New Span n Sequences

This section presents the upper and lower bounds of the linear complexity of new span n sequences generated using WG transformations, three-term, five-term, monomial functions with Kasami exponent, and MCM functions, for different values of n and t. Tables A.8 - A.12 exhibit the upper and lower bounds of the linear complexity of span n sequences produced by WG transformations, five-term, three-term, monomial functions with Kasami exponents, and MCM functions, respectively. We observe that the linear complexity of a span n sequence produced by the structured search lies between $(2^n - 2 - 3n)$ (near-optimal) and $(2^n - 2)$ (optimal).

Table A.8: The bounds of the linear span of WG span n sequences

By recurrence relation (4.1)			
Range on n	t	Upper bound of LS	Lower bound of LS
$7 \leq n \leq 20$	5	$2^n - 2$	$2^n - 2 - 2n$
$8 \leq n \leq 20$	7	$2^n - 2$	$2^n - 2 - 2n$
$9 \leq n \leq 20$	8	$2^n - 2$	$2^n - 2 - 3n$
$11 \leq n \leq 17$	10	$2^n - 2$	$2^n - 2 - 3n$
$12 \leq n \leq 17$	11	$2^n - 2$	$2^n - 2 - 2n$
By recurrence relation (4.2)			
Range on n	t	Upper bound of LS	Lower bound of LS
$7 \leq n \leq 20$	5	$2^n - 2$	$2^n - 2 - 2n$
$8 \leq n \leq 20$	7	$2^n - 2$	$2^n - 2 - 3n$
$9 \leq n \leq 20$	8	$2^n - 2$	$2^n - 2 - 3n$
$11 \leq n \leq 17$	10	$2^n - 2$	$2^n - 2 - 3n$
$12 \leq n \leq 16$	11	$2^n - 2$	$2^n - 2 - 3n$

Table A.9: The bounds of the linear span of five-term span n sequences

By recurrence relation (4.1)			
Range on n	t	Upper bound of LS	Lower bound of LS
$7 \leq n \leq 19$	5	$2^n - 2$	$2^n - 2 - 2n$
$8 \leq n \leq 19$	7	$2^n - 2$	$2^n - 2 - 2n$
$9 \leq n \leq 19$	8	$2^n - 2$	$2^n - 2 - 3n$
$11 \leq n \leq 17$	10	$2^n - 2$	$2^n - 2 - 3n$
$12 \leq n \leq 16$	11	$2^n - 2$	$2^n - 2 - 2n$
By recurrence relation (4.2)			
Range on n	t	Upper bound of LS	Lower bound of LS
$7 \leq n \leq 20$	5	$2^n - 2$	$2^n - 2 - 2n$
$8 \leq n \leq 20$	7	$2^n - 2$	$2^n - 2 - 3n$
$9 \leq n \leq 20$	8	$2^n - 2$	$2^n - 2 - 3n$
$11 \leq n \leq 17$	10	$2^n - 2$	$2^n - 2 - 2n$
$12 \leq n \leq 16$	11	$2^n - 2$	$2^n - 2 - 3n$

Table A.10: The bounds of the linear span of three-term span n sequences

By recurrence relation (4.1)			
Range on n	t	Upper bound of LS	Lower bound of LS
$7 \leq n \leq 17$	5	$2^n - 2$	$2^n - 2 - 2n$
$8 \leq n \leq 17$	7	$2^n - 2$	$2^n - 2 - 3n$
$8 \leq n \leq 17$	9	$2^n - 2$	$2^n - 2 - 3n$
$12 \leq n \leq 17$	11	$2^n - 2$	$2^n - 2 - 3n$
By recurrence relation (4.2)			
Range on n	t	Upper bound of LS	Lower bound of LS
$7 \leq n \leq 17$	5	$2^n - 2$	$2^n - 2 - 2n$
$8 \leq n \leq 17$	7	$2^n - 2$	$2^n - 2 - 2n$
$8 \leq n \leq 17$	9	$2^n - 2$	$2^n - 2 - 3n$
$12 \leq n \leq 17$	11	$2^n - 2$	$2^n - 2 - 2n$

Table A.11: The bounds of the linear span of span n sequences produced by monomial functions with Kasami exponents

By recurrence relation (4.1)			
Range on n	t	Upper bound of LS	Lower bound of LS
$7 \leq n \leq 19$	5	$2^n - 2$	$2^n - 2 - 2n$
$8 \leq n \leq 19$	7	$2^n - 2$	$2^n - 2 - 3n$
$8 \leq n \leq 17$	9	$2^n - 2$	$2^n - 2 - 3n$
$12 \leq n \leq 16$	11	$2^n - 2$	$2^n - 2 - 3n$
By recurrence relation (4.2)			
Range on n	t	Upper bound of LS	Lower bound of LS
$7 \leq n \leq 19$	5	$2^n - 2$	$2^n - 2 - 2n$
$8 \leq n \leq 19$	7	$2^n - 2$	$2^n - 2 - 3n$
$8 \leq n \leq 17$	9	$2^n - 2$	$2^n - 2 - 3n$
$12 \leq n \leq 16$	11	$2^n - 2$	$2^n - 2 - 3n$

Table A.12: The upper and lower bounds of the linear span of MCM span n sequences

By recurrence relations (4.1) and (4.2)				
m	k	Range on n	Upper bound	Lower bound
7	3	$8 \leq n \leq 19$	$2^n - 2$	$2^n - 2 - 3n$
	5	$8 \leq n \leq 19$	$2^n - 2$	$2^n - 2 - 2n$
9	5	$10 \leq n \leq 16$	$2^n - 2$	$2^n - 2 - 3n$
	7	$10 \leq n \leq 16$	$2^n - 2$	$2^n - 2 - 3n$
11	3	$12 \leq n \leq 16$	$2^n - 2$	$2^n - 2 - 3n$
	5	$12 \leq n \leq 16$	$2^n - 2$	$2^n - 2 - 3n$
	7	$12 \leq n \leq 16$	$2^n - 2$	$2^n - 2 - 3n$
	9	$12 \leq n \leq 16$	$2^n - 2$	$2^n - 2 - 3n$

Bibliography

[1] M. Aagaard, G. Gong, and R. K. Mota, "Hardware Implementation of the WG-5 Cipher for Passive RFID Tags", *6th IEEE International Symposium on Hardware-Oriented Security and Trust*, pp. 24 – 29, June 2013.

[2] F.S. Annexstein, "Generating de Bruijn Sequences: An Efficient Implementation", *IEEE Transactions on Computers*, Vol. 46, No. 2, pp. 198 – 200, February 1997.

[3] J. Aumasson, L. Henzen, W. Meier, and M. Naya-Plasencia, "QUARK: A Lightweight Hash", *Cryptographic Hardware and Embedded Systems - CHES 2010*, LNCS, Vol. 6225, pp. 1 – 15, Springer-Verlag, 2010. `http://131002.net/quark/`

[4] G.K. Balachandran, and R.E. Barnett, "A 440-nA True Random Number Generator for Passive RFID Tags", *IEEE Transactions on Circuits and Systems I: Regular Papers*, Vol. 55, No. 11, pp. 3723 – 3732, December 2008.

[5] E.R. Berlekamp, *Algebraic Coding Theory*, McGraw-Hill, New York, ch. 7, 1968.

[6] A. Biryukov, and A. Shamir, "Cryptanalytic Time/Memory/Data Tradeoffs for Stream Ciphers", *Advances in Cryptology-ASIACRYPT'00*, LNCS 1976, pp. 1 – 13. Springer-Verlag, 2000.

[7] A. Bogdanov, M. Knežević, G. Leander, D. Toz, K. Varici, and I. Verbauwhede, "SPONGENT: A Lightweight Hash Function", *Cryptographic Hardware and*

Embedded Systems -CHES 2011, Vol. 6917, pp. 312 – 325, Springer-Verlag, 2011.

[8] N.G. de Bruijn, "A Combinatorial Problem", *Koninklijke Nederlandse Akademie v. Wetenschappen*, Vol. 49, pp. 758 – 764, 1946.

[9] P. Bulens, and K. Kalach, "FPGA Implementations of eSTREAM Phase-2 Focus Candidates with Hardware Profile", *State of the Art of Stream Ciphers Workshop (SASC 2007)*, the ECRYPT Stream Cipher Project Report, Vol. 24, 2007.

[10] M. Burmester and J. Munilla, "Lightweight RFID Authentication with Forward and Backward Security", *ACM Transactions on Information and System Security*, Vol. 14, No. 1, pp. 11, 2011.

[11] C. De Canniére, and B. Preneel, "TRIVIUM – A Stream Cipher Construction Inspired by Block Cipher Design Principles", `http://www.ecrypt.eu.org/stream/trivium.html`.

[12] C. De Canniére, and B. Preneel, "TRIVIUM Specifications", `http://www.ecrypt.eu.org/stream/ciphers/trivium/trivium.pdf`.

[13] C. Carlet, "Vectorial Boolean Functions for Cryptography", `http://www.math.univ-paris13.fr/~carlet/chap-vectorial-fcts-corr.pdf`.

[14] A.H. Chan, R.A. Games, and E.L. Key, "On the Complexities of de Bruijn Sequences", *Journal of Combinatorial Theory, Series A*, Vol. 33, No. 3, pp. 233 – 246, 1982.

[15] A.H. Chan, and R.A. Games, "On the Quadratic Spans of de Bruijn Sequences", *IEEE Transactions on Information Theory*, Vol. 36, No. 4, pp. 822 – 829, July 1990.

[16] A.H. Chan, R.A. Games, and J.J. Rushanan, "On Quadratic m-sequences", *IEEE International Symposium on Information Theory* , pp. 364, July 1994.

[17] T. Chang, B. Park, Y. H. Kim, and I. Song, "An Efficient Implementation of the D-Homomorphism for Generation of de Bruijn Sequences", *IEEE Transactions on Information Theory*, Vol. 45, No. 4, pp. 1280 – 1283, May 1999.

[18] W. Che, H. Deng, W. Tan, and J. Wang, "A Random Number Generator for Application in RFID Tags", *Networked RFID Systems and Lightweight Cryptography*, pp. 279 – 287, Springer-Verlag, 2008.

[19] L. Chen, and G. Gong, *Communication System Security*, Boca Raton, Florida, USA: Chapman & Hall/CRC, 2012.

[20] N. Courtois, "Fast Algebraic Attacks on Stream Ciphers with Linear Feedback", *Advances in Cryptology-CRYPTO 2003,* LNCS 2729, pp. 176 – 194, Springer-Verlag, 2003.

[21] N. Courtois, and W. Meier, "Algebraic Attacks on Stream Ciphers with Linear Feedback", *Advances in Cryptology-EUROCRYPT 2003,* LNCS 2656, pp. 644 – 644, Springer-Verlag, 2003.

[22] N. Courtois, A. Klimov, J. Patarin, and A. Shamir, "Efficient Algorithms for Solving Overdefined Systems of Multivariate Polynomial Equations", *Advances in Cryptology-EUROCRYPT 2000*, LNCS 1807, pp. 392 – 407, Springer-Verlag, 2000.

[23] T. W. Cusick and P. Stanica, *Cryptographic Boolean Functions and Applications*, Academic Press, 2009.

[24] I. Dinur, and A. Shamir, "Cube Attacks on Tweakable Black Box Polynomials", *Advances in Cryptology-EUROCRYPT '09*, LNCS, pp. 278 – 299, Springer-Verlag, 2009.

[25] J. Dillon and H. Dobbertin, "New Cyclic Difference sets with Springer parameters", *Finite Fields and Their Application,* 10, pp. 342 – 389, August 1999.

[26] H. Dobbertin, "Kasami Power Functions, Permutation Polynomials and Cyclic Difference Sets", *Proceedings of the NATO-A.S.I. Workshop Difference Sets,*

Sequences and their Correlation Properties, Bad Windsheim, August 3 -14, 1998, Kluwer, Dordrecht, pp. 133 – 158, 1999.

[27] E. Dubrova, "A List of Maximum Period NLFSRs", Report 2012/166, Cryptology ePrint Archive, 2012. `http://eprint.iacr.org/2012/166.pdf`

[28] P. Ekdahl, and T. Johansson, "SNOW - A New Stream Cipher", *Proceedings of First NESSIE Workshop*, Heverlee, Belgium, 2000.

[29] EPCglobal - The EPC Radio-Frequency Identifiction Protocol Class-1 Generation-2 UHF RFID for Communication at 860-960 MHz, 2008.

[30] eSTREAM - The ECRYPT Stream Cipher Project, `http://www.ecrypt.eu.org/stream/.`

[31] T. Etzion, "Linear Complexity of de Bruijn Sequences – Old and New Results", *IEEE Transactions on Information Theory,* Vol. 45, No. 2, pp. 693 – 698, March 1999.

[32] T. Etzion and A. Lempel, "Construction of de Bruijn Sequences of Minimal Complexity", *IEEE Transactions on Information Theory,* Vol. 30, No. 5, pp. 705 – 709, September 1984.

[33] H. Fredricksen, "The Lexicographically Least de Bruijn Cycle", *Journal Combinatorial Theory* Vol. 9, pp. 1 – 5, 1970.

[34] H. Fredricksen, "Generation of the Ford Sequence of Length 2^n, n Large", *Journal Combinatorial Theory*, Series A 12, pp. 153 – 154, 1972.

[35] H. Fredricksen, "A Class of Nonlinear de Bruijn Cycles", *Journal of Combinatorial Theory*, Series A, Vol. 19, Issue 2, pp. 192 – 199, September 1975.

[36] H. Fredricksen, "A Survey of Full Length Nonlinear Shift Register Cycle Algorithms", *SIAM Review*, Vol. 24, No. 2, pp. 195 – 221, 1982.

[37] H. Fredricksen and I. Kessler, "Lexicographic Compositions and de Bruijn Sequences", *Journal Combinatorial Theory*, Series A 22, pp. 17 – 30, 1977.

[38] H. Fredricksen and J. Maiorana, "Necklaces of Beads in k Colors and k-ary de Bruijn Sequences", *Discrete Mathematics*, Vol. 23, Issue 3, pp. 207 – 210, 1978.

[39] R. A. Games, "A Generalized Recursive Construction for de Bruijn Sequences", *IEEE Transactions on Information Theory,* Vol. 29, No. 6, pp. 843 – 850, September 1983.

[40] B. M. Gammel, R. Göttfert, and O. Kniffler, "The Achterbahn Stream Cipher", 2005. `http://www.ecrypt.eu.org/stream/ciphers/achterbahn/achterbahn.pdf`

[41] B. M. Gammel, R. Göttfert, and O. Kniffler, "Achterbahn-128/80", 2006. `http://www.ecrypt.eu.org/stream/p2ciphers/achterbahn/achterbahn_p2.pdf`

[42] B. M. Gammel, R. Göttfert, and O. Kniffler, "An NLFSR-based Stream Cipher", *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS'2006)*, pp. 4 – 8, 2006.

[43] S.W. Golomb, *Shift Register Sequences*, Aegean Park Press, Laguna Hills, CA, USA, 1981.

[44] S.W. Golomb, "On the Classification of Balanced Binary Sequences of Period $2^n - 1$", *IEEE Transformation on Information Theory,* Vol. 26, No. 6, pp. 730 – 732, November 1980.

[45] S.W. Golomb, and G. Gong, *Signal Design for Good Correlation: For Wireless Communication, Cryptography, and Radar*, Cambridge University Press, New York, NY, USA, 2004.

[46] G. Gong, S. Rønjom, T. Helleseth, and H. Hu, "Fast Discrete Fourier Spectra Attacks on Stream Ciphers", *IEEE Transactions on Information Theory*, Vol 57, No. 8, pp. 5555 – 5565, August 2011.

[47] G. Gong, and A. Youssef, "Cryptographic Properties of the Welch-Gong Transformation Sequence Generators", *IEEE Transactions on Information Theory*, Vol. 48, No. 11, pp. 2837 – 2846, November 2002.

[48] G. Gong, "Randomness and Representation of Span n Sequences", *Proceedings of the 2007 International Conference on Sequences, Subsequences, and Consequences, SSC'07*, pp. 192 – 203, Springer-Verlag, 2007.

[49] I.J. Good, "Normal Recurring Decimals", *Journal of London Math. Soc.*, Vol. 21 (Part 3), 1946.

[50] D. H. Green and K. R. Dimond, "Nonlinear Product-Feedback Shift Registers", *Proceeding IEE 117*, pp. 681 – 686, 1970.

[51] D. H. Green and K. R. Dimond, "Some Polynomial Compositions of Nonlinear Feedback Shift Registers and their Sequence-Domain Consequences", *Proc. IEE 117*, pp. 1750 – 1756, 1970.

[52] J. Guo, T. Peyrin, and A. Poschmann, "The PHOTON Family of Lightweight Hash Functions", *Advances in Cryptology-CRYPTO'11*, pp. 222 – 239, Springer-Verlag, 2011.

[53] E. R. Hauge and T. Helleseth, "De Bruijn Sequences, Irreducible Codes and Cyclotomy", *Discrete Mathematics*, Vol. 159, Issues 1 – 3, pp. 143 – 154, November 1996.

[54] E. R. Hauge, J. Mykkeltveit, "On the Classification of de Bruijn Sequences", *Discrete Mathematics*, Vol. 148, Issues 1 – 3, pp. 65 – 83, January 1996.

[55] M. Hell, T. Johansson, and W. Meier, "Grain: A Stream Cipher for Constrained Environments", *Int. J. Wire. Mob. Comput.*, Vol. 2, pp. 86 – 93, May 2007.

[56] S. Hellebrand, J. Rajski, S. Tarnick, S. Venkataraman, and B. Courtois, "Built-In Test for Circuits with Scan Based on Reseeding of Multiple-Polynomial

Linear Feedback Shift Registers", *IEEE Transactions on Computers*, Vol. 44, pp. 223 – 233, February 1995.

[57] D.E. Holcomb, W.P. Burleson, and K. Fu, "Initial SRAM State as a Fingerprint and Source of True Random Numbers for RFID Tags", *Proceedings of the Conference on RFID Security*, 2007.

[58] H. Hu and G. Gong, "Periods on Two Kinds of Nonlinear Feedback Shift Registers with Time Varying Feedback Functions", *International Journal of Foundations of Computer Science*, Vol. 22, No. 6, pp. 1317 – 1329, September 2011.

[59] C.J.A. Jansen, W.G. Franx, and D.E. Boekee, "An Efficient Algorithm for the Generation of de Bruijn Cycles", *IEEE Transactions on Information Theory*, Vol. 37, No. 5, pp. 1475 – 1478, September 1991.

[60] A. Juels, "RFID Security and Privacy: A Research Survey", *IEEE Journal on Selected Areas in Communications (J-SAC)*, Vol. 24, No. 2, pp. 381 – 394, February 2006.

[61] E. Kavun, and T. Yalcin, "A Lightweight Implementation of Keccak Hash Function for Radio-Frequency Identification Applications", In: Ors Yalcin, S.B. (ed.) *RFIDSec 2010*, LNCS, Vol. 6370, pp. 258 – 269, Springer-Heidelberg, 2010.

[62] E. L. Key, "An Analysis of the Structure and Complexity of Nonlinear Binary Sequence Generators", *IEEE Transactions on Information Theory*, Vol 22, No. 6, pp. 732 – 736, November 1976.

[63] A. Klapper, "Linear Complexity of Finite Field Sequences over Different Fields", *International Workshop on Sequence Design and Applications (IWSDA)*, Fukuoka, Japan, October 2005.

[64] A. Klimov and A. Shamir, "A New Class of Invertible Mappings", *CHES 2002*, LNCS, Vol. 2523, pp. 470 – 483, Springer-Heidelberg, 2003.

[65] D.E. Knuth, *The Art of Computer Programming*, Volume 2, Seminumerical Algorithms, Addison-Wesley, 1969.

[66] C. Lam, M. Aagaard and G. Gong, "Hardware Implementations of Multi-output Welch-Gong Ciphers", Technical Report CACR, 2011, `http://www.cacr.math.uwaterloo.ca/`.

[67] A. Lempel, "On a Homomorphism of the de Bruijn Graph and its Applications to the Design of Feedback Shift Registers", *IEEE Transactions on Computers*, Vol. C-19, Issue 12, pp. 1204 – 1209, December 1970.

[68] R. Lidl, and H. Niederreiter, *Finite Fields*, Cambridge University Press, 1997.

[69] K. Mandal, and G. Gong, "Probabilistic Generation of Good Span n Sequences from Nonlinear Feedback Shift Registers", Technical Report CACR 2012-06, University of Waterloo, 2012.

[70] K. Mandal, and G. Gong, "Cryptographically Strong de Bruijn Sequences with Large Periods", In: L.R. Knudsen, H. Wu (Eds.), *Selected Areas in Cryptography, SAC'12*, LNCS, Vol. 7707, pp. 104 – 118, Springer, Heidelberg, 2012.

[71] K. Mandal, and G. Gong, "Cryptographic D-morphic Analysis and Fast Implementations of Composited De Bruijn Sequences", Technical Report CACR 2012-27, University of Waterloo, 2012.

[72] K. Mandal, X. Fan, and G. Gong, "Warbler: A Lightweight Pseudorandom Number Generator for EPC C1 Gen2 RFID Tags", *Cryptology and Information Security Series - The 2012 Workshop on RFID and IoT Security (RFIDsec'12 Asia)*, Vol. 8, N.W. Lo and Y. Li (Eds.), Amsterdam, Netherlands: IOS Press, pp. 73-84, 2012.

[73] K. Mandal, X. Fan, and G. Gong, "A Lightweight Pseudorandom Number Generator for EPC C1 Gen2 RFID Tags", *WEWoRC 2011,* `http://www.uni-weimar.de/cms/fileadmin/medien/medsicherheit/WEWoRC2011/files/conference_record3.pdf`

[74] K. Mandal, X. Fan, and G. Gong, "Warbler: A Lightweight Pseudorandom Number Generator for EPC C1 Gen2 Passive RFID Tags", submitted at IJRFIDSC, 2013.

[75] K. Mandal, X. Fan, and G. Gong, "Warbler Family of Lightweight Pseudorandom Number Generators for Smart Devices". In submission, 2013.

[76] K. Mandal, G. Gong, X. Fan, and M. Aagaard, "Optimal Parameters for the WG Stream Cipher Family", Technical Report CACR 2013-15, University of Waterloo, 2013. To appear at CCDS.

[77] K. Mandal, G. Gong, X. Fan, and M. Aagaard. "On Selection of Optimal Parameters for the WG Stream Cipher Family", *Proceedings of 13th Canadian Workshop on Information Theory (CWIT'13)*, pp. 17 – 21, June 2013.

[78] H. Martin, E. San Millan, L. Entrena, P.P. Lopez, J.C.H. Castro, "AKARI-X: A Pseudorandom Number Generator for Secure Lightweight Systems", *2011 IEEE 17th International On-Line Testing Symposium (IOLTS)*, Vol. 228, No. 233, pp. 13 – 15, July 2011.

[79] J.L. Massey, "Shift-Register Synthesis and BCH Decoding", *IEEE Transactions on Information Theory*, Vol. 15, No. 1, pp. 122 – 127, 1969.

[80] G.L. Mayhew, and S.W. Golomb, "Linear Spans of Modified de Bruijn Sequences", *IEEE Transactions on Information Theory*, Vol. 36, No. 5, pp. 1166 – 1167, September 1990.

[81] G.L. Mayhew, "Weight Class Distributions of de Bruijn Sequences", *Discrete Math.*, Vol. 126, pp. 425 – 429, March 1994.

[82] G.L. Mayhew, and S.W. Golomb, "Characterizations of Generators for Modified de Bruijn Sequences", *Advanced Applied Mathematics*, Vol. 13, pp. 454 – 461, December 1992.

[83] G.L. Mayhew, "Clues to the Hidden Nature of de Bruijn Sequences", *Computers and Mathematics with Applications*, Vol. 39, No. 11, pp. 57 – 65, 2000.

[84] W. Meier, and O. Staffelbach, "Fast Correlation Attacks on Certain Stream Ciphers", *Journal of Cryptology*, pp. 159 – 176, 1989.

[85] J. Melia-Segui, J. Garcia-Alfaro, and J. Herrera-Joancomarti, "Analysis and Improvement of a Pseudorandom Number Generator for EPC Gen2 Tags", *Proceedings of the 14th International conference on Financial Cryptography and Data Security, FC'10*, pp. 34 – 46, Springer-Verlag, 2010.

[86] J. Meliá-Seguí , J. Garcia-Alfaro, and J. Herrera-Joancomartí, "J3Gen: A PRNG for Low-Cost Passive RFID", *Sensors*, Vol. 13, No. 3, pp. 3816 – 3830, 2013.

[87] A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1997.

[88] M.J. Mihaljevic, and J.D. Gólic, "A Fast Iterative Algorithm for a Shift Register Initial State Reconstruction Given the Noisy Output Sequence", *Advances in Cryptology-AUSCRYPT'90,* LNCS, pp. 165 – 175, Springer-Verlag, 1990.

[89] M.J. Mihaljević, S. Gangopadhyay, G. Paul, and H. Imai, "Internal State Recovery of Grain-v1 Employing Normality Order of the Filter Function", *IET Information Security*, Vol.6, No.2, pp. 55 – 64, June 2012.

[90] J. Mykkeltveit, "Generalization of a Theorem on Linear Recurrence to the Nonlinear Case", Internal Report, University of Bergen, Bergen, 1976.

[91] J. Mykkeltveit, "Generating and Counting the Double Adjacencies in a Pure Cycling Shift Register, *IEEE Trans. Electronic Computers*, C-24, pp. 299 – 304, 1975.

[92] J. Mykkeltveit, M-K. Siu, and P. Tong, "On the Cycle Structure of Some Nonlinear Shift Register Sequences", *Information and Control*, pp. 202 – 215, 1979.

[93] Y. Nawaz, and G. Gong, "The WG Stream Cipher", *Workshop on Symmetric Key Encryption,* Aarhus, Denmark, May 26 - 27, 2005.

[94] Y. Nawaz, and G. Gong, "WG: A Family of Stream Ciphers with Designed Randomness Properties", *Information Science*, Vol. 178, No. 7, pp. 1903 – 1916, April 2008.

[95] J.L. Ng, "Binary Nonlinear Feedback Shift Register Sequence Generator using the Trace Function", Master's Thesis, University of Waterloo, 2005.

[96] J.S. No, S.W. Golomb, G. Gong, H.K. Lee, and P. Gaal, "New Binary Pseudorandom Sequences of Period $2^n - 1$ with Ideal Autocorrelation", *IEEE Transactions on Information Theory*, Vol. 44, No. 2, pp. 814 – 817, March 1998.

[97] P. Peris-Lopez, J. Hernandez-Castro, J.M. Estevez-Tapiador, and A. Ribagorda, "LAMED - A PRNG for EPC Class-1 Generation-2 RFID Specification", *Computer Standards and Interfaces*, pp. 88 – 97, January 2009.

[98] D.C. Ranasinghe, and P.H. Cole, "An Evaluation Framework", *Networked RFID Systems and Lightweight Cryptography*, pp. 157 – 167, Springer-Verlag, 2008.

[99] T. Rachwalik, J. Szmidt, R. Wicik, and J. Zablocki, "Generation of Nonlinear Feedback Shift Registers with Special-Purpose Hardware", Report 2012/314, Cryptology ePrint Archive, 2012, `http://eprint.iacr.org/`.

[100] S. Rønjom, and T. Helleseth, "A New Attack on the Filter Generator, *IEEE Transactions on Information Theory*, Vol. 53, No. 5, pp. 1752 – 1758, May 2007.

[101] S. Rønjom, G. Gong, and T. Helleseth, "On Attacks on Filtering Generators using Linear Subspace Structures", *Proceedings of the 2007 International Conference on Sequences, Subsequences, and Consequences, SSC'07*, pp. 204 – 217, Springer-Verlag, 2007.

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

[103] A. Rukhin, J. Soto, J. Nechvatal, E. Barker, S. Leigh, M. Levenson, D. Banks, J. Dray, S. Vo, M. Smid, M. Vangel, A. Heckert, and L.E. Iii, "A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications", 2001. `http://csrc.nist.gov/groups/ST/toolkit/rng/index.html`

[104] M. O. Saarinen, "A Time-Memory Tradeoff Attack Against LILI-128, *Fast Software Encryption (FSE) 2002*, LNCS 2365, pp. 231 – 236, Springer-Verlag, 2002.

[105] T. Siegenthaler, "Correlation-immunity of Nonlinear Combining Functions for Cryptographic Applications", *IEEE Transactions on Information Theory*, Vol. 30, No. 5, pp. 776 – 780, September 1984.

[106] G.J. Simmons, *Contemporary Cryptology: The Science of Information Integrity*, IEEE Press, 1994.

[107] M.-K. Siu and P. Tong, "Generation of Some de Bruijn Sequences", *Discrete Mathematics*, Vol. 31, Issue 1, pp. 97 – 100, 1980.

[108] D.R. Stinson, *Cryptography Theory and Practice*, CRC Press, 2005.

[109] V.B. Suresh, and W.P. Burleson, "Entropy Extraction in Metastability-Based TRNG", *IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pp. 135 – 140, June 2010.

[110] H. Wu, and B. Preneel, "Chosen IV Attack on Stream Cipher WG", *ECRYPT Stream Cipher Project Report 2005/045*, Available at `http://cr.yp.to/streamciphers/wg/045.pdf`.

[111] Jun-H. Yang and Zong-D. Dai, "Construction of m-ary de Bruijn Sequences (extended abstract)", *Advances in Cryptology – AUSCRYPT '92*, LNCS, pp. 357 – 363, Springer-Heidelberg, 1993.

[112] `http://www.avoine.net/rfid/`