Cryptographic Applications of Non-Commutative Algebraic Structures and Investigations of Nonlinear Recursions

Petrides, George

2006

MIMS EPrint: **2007.115**

Manchester Institute for Mathematical Sciences
School of Mathematics

The University of Manchester

Reports available from: http://eprints.maths.manchester.ac.uk/
And by contacting: The MIMS Secretary
School of Mathematics
The University of Manchester
Manchester, M13 9PL, UK

ISSN 1749-9097

Cryptographic Applications of Non-Commutative Algebraic Structures and Investigations of Nonlinear Recursions

A thesis submitted to the University of Manchester for the degree of
Doctor of Philosophy
in the Faculty of Engineering and Physical Sciences

2006

George Petrides

School of Mathematics

Contents

13619 words

List of Tables

List of Figures

THE UNIVERSITY OF MANCHESTER

ABSTRACT OF THESIS submitted by **George Petrides**

For the degree of Doctor of Philosophy

and entitled **Cryptographic Applications of Non-Commutative Algebraic Structures and Investigations of Nonlinear Recursions.**

Month and Year of Submission: **June 2006**

In this thesis we investigate the application of non-commutative algebraic structures and nonlinear recursions in cryptography. To begin with, we demonstrate that the public key cryptosystem based on the word problem on the Grigorchuk groups, as proposed by M. Garzon and Y. Zalcstein [8], is insecure. We do this by exploiting information contained in the public key in order to construct a key which behaves like the private key and allows successful decryption of ciphertexts.

Further on, we present a new block cipher with key-dependent S-boxes, based on the Grigorchuk groups. To the best of our knowledge, it is the first time groups are used in a block cipher, whereas they have been extensively used in public key cryptosystems. The study of the cipher's properties is, at this stage, purely theoretical.

Finally, we investigate the notion of nonlinear complexity, or maximal order complexity as it was first defined in 1989 [15], for sequences. Our main purpose is to begin classification of periodic binary sequences into nonlinear complexity classes. Previous work on the subject also includes approximation of the size of each class, found in [7]. Once the classification is completed, we can use it to show how to perform checks for short cycles in large nonlinear feedback shift registers using our proposed algorithm.

Declaration

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Copyright

Acknowledgements

Many people have helped me through the long and difficult process of studying for my PhD. I would like to use this space to say a big thank you to all of them.

To begin with, I am indebted to my supervisor professor Alexandre Borovik for accepting me as his student and for guiding me through my research. I am also indebted to professor Tor Helleseth for allowing me to visit the University of Bergen in Bergen, Norway, under a Marie Curie Scholarship, and for introducing me to Johannes Mykkeltveit. Johannes devoted a lot of his time discussing his ideas and mine with me, which led to a co-authored paper forming part of this thesis. Tusen takk for that.

Next, I would like to thank my good friends who encouraged me all this time and lifted me up through my difficult times, even in a way that was not obvious to them: Alan, Aurelia, Albert, Aline, Andreas, Bente, Connie, Danijela, Eleni, Elisa, Fosca, Giorgos, Marcel, Marios and Marios, Matt, Zoltan, with special thanks to An, Chris, Elias, Kanaris, Simone and Stavros. Apologies to anyone who has been (unintentionally) left out.

I could never leave out the beloved members of my family who always showed their interest in my well being and their enthusiasm whenever we met: $\Sigma\alpha\varsigma\ \varepsilon\upsilon\chi\alpha\rho\iota\sigma\tau\acute{\omega}\ \pi o\lambda\acute{\upsilon}$. Special thanks go to my brother Petros, my cousins Paris and Natalia and their parents, uncle Antreas and aunt Maria, and my aunt Androulla who made me feel I have a second home in Britain.

Above all people, I am most thankful to my parents, John and Chrystalla. Without their moral support throughout my studies, I wouldn't have managed to finish

the hard task of writing this thesis. Without their financial support all these years, I would not even have made it to the level of pursuing a PhD. I thank them for bringing me up in the way they did and for providing me a healthy environment to receive my education. As a very small token of my appreciation, I dedicate this thesis to you.

Chapter 1

Introduction

In 1980, Rostislav Grigorchuk discovered the first Grigorchuk group G in [9], as a negative answer to Burnside's problem: 'Let Γ be a finitely-generated group such that each of its elements has finite order; is Γ necessarily a finite group?'. Later on, in 1985, he generalised his group construction in [10], this time as a negative solution to Milnor's question: 'Is the growth function of every finitely generated group equivalent either to a power function n^d or to the exponential function 2^n?'.

The first application of the Grigorchuk groups in cryptography was done by Garzon and Zalcstein in 1991 [8]. They proposed a public key cryptosystem based on the word problem on the Grigorchuk groups. This cryptosystem was similar to a cryptosystem proposed in 1984 by Wagner and Magyarik [32]. The latter one however, was based on finitely presented groups, whereas the Grigorchuk groups considered by Garzon and Zalcstein are not finitely presented.

In their discussion of security issues of their cryptosystem, amongst other things, Garzon and Zalcstein claimed that the public key does not contain enough information to uniquely determine the private key.

Cryptanalysis of this cryptosystem was first attempted by Hofheinz and Steinwandt in [31]. In their paper, they tried to exploit the public key to derive a secret key equivalent to the original one by an exhaustive search of all possibilities.

In Chapter 3, we prove that the words used as public key, in combination with

Algorithm 2 presented in Section 3.1, give enough information to allow construction of a key which behaves like the private key and is capable of successfully decrypting ciphertexts. This part of the thesis appears in the proceedings of the Institute of Mathematics and its Applications Ninth International Conference on Cryptography and Coding [24].

The above mentioned cryptosystem is an example of group theory's extensive provision of tools for use in public key cryptography. Other examples include decision problems in *Braid* groups [1, 16]. In general, algebraic objects are used as cryptographic primitives quite often, with a recent successful example being the block cipher *Rijndael* [4], selected as the Advanced Encryption Standard (AES). Like many other cryptosystems, including the pioneer Diffie-Hellman Key Exchange Protocol [6], it uses computations in finite fields for its internal operations.

To the extend of our knowledge, groups have not so far been used in the construction of block ciphers. In this thesis we take advantage of this gap, and try to investigate theoretically the feasibility of such an application of groups. To this end, we present a new block cipher based on the Grigorchuk groups in Chapter 4. Our choice of these particular groups was influenced by the way they are defined (as groups of transformations of the binary tree), which makes them suitable for handling blocks of plaintext and ciphertext. Moreover, they can have a streamlined implementation using automata.

One of the characteristics of our proposed cipher is that it has key-dependent S-boxes. This feature makes differential [3], linear [20] and similar kinds of cryptanalysis very difficult to mount. Other ciphers following the key-dependent S-box philosophy include Khufu [22], Blowfish [27] and Twofish [28]. The alternative method of resistance against these cryptanalytic attacks is through careful design of the S-boxes, achieved, for instance, by the *Wide Trail Design Strategy* of Rijndael [4].

The counterpart to block ciphers are stream ciphers. The classical complexity measure assessing the cryptographic strength of binary sequences used in stream ciphers is the *linear complexity*. It can be calculated using the well known Berlekamp-Massey algorithm [19] and is used in statistical tests for the randomness of sequences.

In Chapter 5 we investigate the generalised notion of *nonlinear complexity* (or maximum order complexity as first introduced in [15]) which can be calculated using, for example, the directed acyclic word graph [15]. In particular, we try to classify periodic binary sequences of given period into nonlinear complexity classes.

In [7] an approximate number of sequences in each class was calculated and used in finding the approximate probability distribution of nonlinear complexity. Our results, though incomplete, give the exact number for the cases considered. The cases not yet dealt with are left for future work, with the idea and method of approach having been established.

A complete classification will be useful in the implementation of an algorithm checking for short cycles in large *nonlinear feedback shift registers* (NLFSRs). Feedback shift registers are the main components of stream ciphers, with more emphasis given on linear ones, for which the cycle structure is known. Our results constitute a new contribution to the theory of NLFSRs. In [17] it is claimed that for a given large NLFSR it is hard to check whether short cycles have been embedded by the given method: brute force is inefficient due to largeness, and so are the two algorithms which are given for such a check, based on algebraic approach. Our proposed algorithm checks for short cycles, regardless of whether they are embedded or not.

Another use for this classification (once complete) can be found in the construction of statistical tests for the randomness of sequences, as shown for example in [7], thus further upgrading the level of interest of nonlinear complexity from just theoretical to also practical. For instance, we could say that a sequence is considered random if it belongs to a large nonlinear complexity class. The results of Chapter 5 have been accepted for publication in the proceedings of the International Conference on Sequences and Their Applications 2006 [25].

All the background material necessary for understanding the problems considered in this thesis are provided in Chapter 2. They include a survey of the Grigorchuk groups and some information on introductory cryptography. Finally, in Chapter 6 we give our overall conclusions on the results of this thesis and discuss open problems left for future work.

Chapter 2

Introductory Material

2.1 The Grigorchuk Groups

In this section we will give a brief overview of the Grigorchuk groups and their main properties. We start with the first Grigorchuk group and then proceed to the generalised Grigorchuk groups.

2.1.1 The First Grigorchuk Group

The first Grigorchuk group G was discovered in 1980 by and named after the great mathematician Rostislav Grigorchuk [9]. Although in the original paper G was defined as a group of transformations of the interval [0,1], it is more efficient to consider transformations of the infinite binary tree.

Denote by T the set of one way infinite paths from the root $\emptyset$ of the complete infinite binary tree. Each $j \in T$ can be regarded as an infinite binary sequence $(j_i)_{i\geq 1}$ of vertices. We refer to these as left turns (denoted by 0) and right turns (denoted by 1), according to orientation from the parent. The empty sequence denotes the root vertex $\emptyset$. For an illustration of this see Figure 2.1.

The group G is a group of permutations of T generated by four bijections $a,\ b,\ c$ and d which act as follows on a given path $j = (j_1, j_2, j_3, \ldots, j_k, \ldots) \in T$:

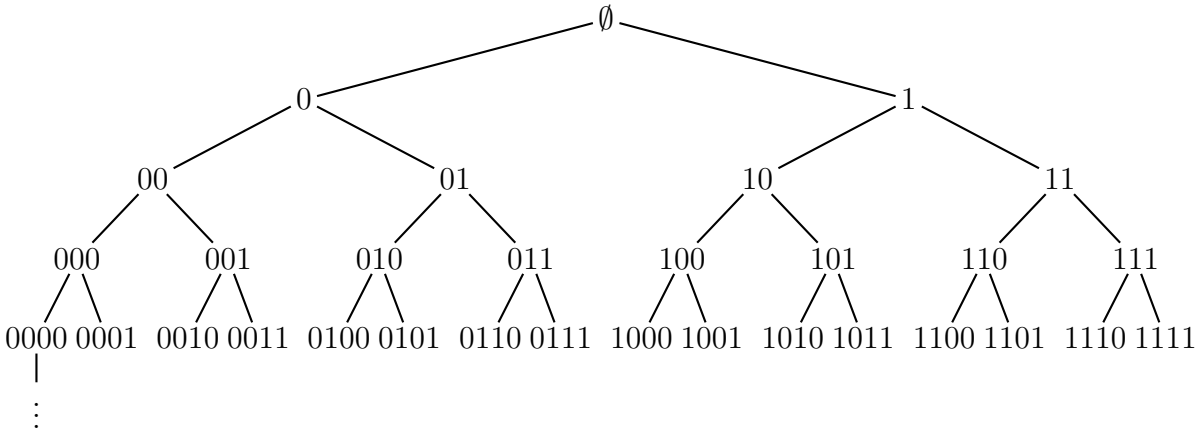


Figure 2.1: The first four levels of the infinite binary tree

$a(j_1, j_2, j_3, \ldots) = (\overline{j_1}, j_2, j_3, \ldots)$, where $\overline{0} = 1$ and $\overline{1} = 0$. In other words, a swaps the two halves of the tree. Generators b, c and d are defined recursively and simultaneously:

$$\begin{cases} b(0, j_2, j_3, \ldots) = (0, \overline{j_2}, j_3, \ldots) \\ b(1, j_2, j_3, \ldots) = (1, c(j_2, j_3, \ldots)) \\ \\ c(0, j_2, j_3, \ldots) = (0, \overline{j_2}, j_3, \ldots) \\ c(1, j_2, j_3, \ldots) = (1, d(j_2, j_3, \ldots)) \\ \\ d(0, j_2, j_3, \ldots) = (0, j_2, j_3, \ldots) \\ d(1, j_2, j_3, \ldots) = (1, b(j_2, j_3, \ldots)) \end{cases}$$

Example 2.1.1 $b(1,1,1,0,1,\ldots) = (1, c(1,1,0,1,\ldots)) = (1,1,d(1,0,1,\ldots))$
$= (1,1,1,b(0,1,\ldots)) = (1,1,1,0,0,\ldots)$.

Since the first left turns in the path j are not complemented by any of the generators b, c and d, a second application of the same generator will return the path to its original shape. This is also trivial for a. Hence all four generators are involutions:

$$a^2 = b^2 = c^2 = d^2 = 1 \ . \tag{2.1}$$

We also have that

$$bc = cb = d,\ bd = db = c \text{ and } cd = dc = b\ . \tag{2.2}$$

Therefore, anyone of the generators b, c and d can be dropped from the generating set $\langle a, b, c, d\rangle$.

The group G can be represented by a finite state automaton. An automaton can be thought of as a machine, which being in a state g and receiving as input a letter x, goes into state h and outputs a letter y, both depending on letter x.

Finite automata can be represented by Moore diagrams. Figure 2.2 shows a representation of the Grigorchuk group G using such a Moore diagram.

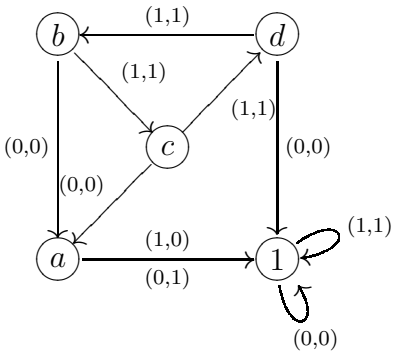
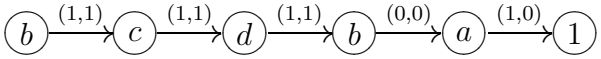


Figure 2.2: The Moore diagram representing G

The five states of the automaton, represented by the nodes of the diagram, consist of the four generators of G plus the identity element. From each state there are two outgoing arrows, each together with a pair (x, y), leading to another state. In each pair (x, y) x denotes input and y denotes output. Starting from the generator of G we want to apply to a path $j \in T$, we follow the arrow in the direction of the appropriate input (the first digit of j) to lead us to the next state, while at the same time receiving the corresponding output. Carrying on in this way for every new state and corresponding input until we reach the identity element, the output will yield a new path $j' \in T$. This denotes the action of the generator of G we started with on the path $j \in T$.

Example 2.1.2 *To determine the action of generator b on path $j = (1, 1, 1, 0, 1, \ldots)$,*

the automaton goes through the following states and corresponding input/output pairs:



The result is the output path $j' = (1, 1, 1, 0, 0, \ldots)$.

Further information on automata can be found in [5] and [12].

2.1.2 Generalised Grigorchuk Groups

Grigorchuk generalised his group construction in 1984 [10], as follows: Given an infinite ternary sequence ω, the group G_ω is a group of permutations of T generated by four bijections $a,\ b_\omega,\ c_\omega$ and d_ω. The action of a on a path $j \in T$ consists of complementing the first turn as before (that is making 0 into 1 and vice versa). However, the actions of $b_\omega,\ c_\omega$ and d_ω depend on the sequence ω as follows:

We form three sequences $U,\ V$ and W by substituting each digit of ω with a vector as shown below, depending on whether it is 0, 1 or 2:

$$0 : \begin{cases} \text{S} \\ \text{S} \\ \text{I} \end{cases} \qquad 1 : \begin{cases} \text{S} \\ \text{I} \\ \text{S} \end{cases} \qquad 2 : \begin{cases} \text{I} \\ \text{S} \\ \text{S} \end{cases}$$

Here I = "Identity" and S = "Swap"; the latter means making 0 into 1 and vice versa.

We thus obtain three sequences:

$$\begin{aligned} U &= u_1\, u_2 \ldots u_n \ldots \\ V &= v_1\, v_2 \ldots v_n \ldots \\ W &= w_1 w_2 \ldots w_n \ldots \end{aligned}$$

The bijection b_ω (respectively $c_\omega,\ d_\omega$) leaves invariant all turns $j_1, \ldots, j_i$ including the first left turn j_i of j and complements j_{i+1} if u_i (respectively $v_i,\ w_i$) is S. Otherwise it leaves j invariant.

Example 2.1.3 *If* $\omega = 012012012\ldots$ *then*

$$U = \mathrm{SSISSISSI}\ldots$$
$$V = \mathrm{SISSISSIS}\ldots$$
$$W = \mathrm{ISSISSISS}\ldots$$

and $b_\omega(1,1,1,0,1,\ldots) = (1,1,1,0,0,\ldots)$ *since* $u_4 = \mathrm{S}$.

Using this construction, the first Grigorchuk group is generated by the periodic sequence $012012012\ldots$.

All four generators are once again involutions, and equations (2.1) and (2.2) become

$$a^2 = b_\omega^2 = c_\omega^2 = d_\omega^2 = 1 \ , \tag{2.3}$$

and

$$b_\omega c_\omega = c_\omega b_\omega = d_\omega,\ b_\omega d_\omega = d_\omega b_\omega = c_\omega \text{ and } c_\omega d_\omega = d_\omega c_\omega = b_\omega \ . \tag{2.4}$$

Therefore, any one of the generators b_ω, c_ω and d_ω can be dropped from the generating set $\langle a, b_\omega, c_\omega, d_\omega\rangle$.

It can be checked that the action of the four generators is nonlinear. That is to say that if $*(j) = j'$ and $*(t) = t'$ then $*(j+t) \neq j' + t'$, where $* \in \{a,\ b_\omega,\ c_\omega,\ d_\omega\}$ and $j, j', t, t' \in T$. Here, the addition of two paths $j = (j_0 j_1 j_2 \ldots)$ and $t = (t_0 t_1 t_2 \ldots) \in T$ is realised as follows: $j + t = (j_0 + t_0 \bmod 2,\ j_1 + t_1 \bmod 2,\ j_2 + t_2 \bmod 2,\ \ldots) \in T$.

If we restrict ourselves on the n^{th} level of the infinite binary tree, the generators $b_\omega,\ c_\omega$ and d_ω now act on binary sequences of finite length n. Therefore, only the first $n-1$ digits of the sequence ω are needed to determine their action: if the first left turn is the n^{th} turn of the now finite path, the path is left invariant.

Example 2.1.4 *If* $\omega = 012012012\ldots$ *then on the fifth level,* $b_\omega(1,1,1,0,1) = (1,1,1,0,0)$ *and on the fourth level,* $b_\omega(1,1,1,0) = (1,1,1,0)$.

Notice that every Grigorchuk group G_ω is a canonical homomorphic image of the *basis* group Γ, generated by four elements $a,\ b_\omega,\ c_\omega$ and d_ω satisfying only relations

(2.3) and (2.4). When a sequence ω is specified, it introduces extra relations between generators $a,\ b_\omega,\ c_\omega$ and d_ω and maps Γ onto the Grigorchuk group G_ω.

The Grigorchuk groups are infinite, residually finite (that is, given $g \neq 1$ in $G_\omega,\ \exists N \triangleleft G_\omega$ such that $g \notin N$ and $|G_\omega/N| < \infty$) and

(1) If all 3 symbols 0,1 and 2 repeat infinitely often in ω then G_ω is a 2-group, that is for $g \in G_\omega,\ \exists N \geq 0$ such that $g^{2^N} = 1$ (Theorem 2.1(1) in [10]).

(2) If at least two of 0,1 and 2 repeat infinitely often in ω then G_ω is not finitely presentable; that is, one needs infinitely many independent relators to define G_ω (Theorem 6.2 in [10]). For definition of relators see Subsection 2.1.3.

In the cases when ω is periodic, G_ω can be defined by a finite state automaton in a similar way as shown in Subsection 2.1.1 for G, which allows for an especially streamlined implementation.

2.1.3 Word Problem

A *word* in G_ω is any product of the four generators $a,\ b_\omega,\ c_\omega$ and d_ω and represents an element of G_ω. Let F be such a word and let $\partial(F)$ denote the *length* of the word F, that is the the number of generators in the product. Similarly, let $\partial_k(F)$ (respectively $\partial_{p,q}(F)$ etc) denote the number of occurrences of symbol k (respectively both symbols p and q etc) in the word F ($k,\ p$ and $q \in \{a,\ b_\omega,\ c_\omega,\ d_\omega\}$). Due to equations (2.3) and (2.4), each word can be uniquely reduced to $*'$ or the form

$$*'a * a \ldots a * a*' \ , \tag{2.5}$$

where $* \in \{b_\omega,\ c_\omega,\ d_\omega\}$ and $*' \in \{1_{G_\omega},\ b_\omega,\ c_\omega,\ d_\omega\}$. We call words of this form *reduced*. In fact, reduced words are elements of the basis group Γ.

The inverse F^{-1} of a word F is obtained by reversing the order of appearance of the letters in F.

By *word problem* we mean finding whether a given word F in G_ω is equal to 1 or not. The word problem is solvable when the sequence ω is recursive [10, Sect. 5] and we shall shortly give a description of the algorithm used to solve it.

Words equal to 1 are called *relators*. By definition, relators act trivially on any path $j \in T$. Recalling the action of generator a on such a path j, we can deduce that all relators must have an even number of occurrences of the generator a.

Let σ denote the left shift operator defined on a sequence $\omega = \omega_1\omega_2\omega_3\omega_4\ldots$ by $\sigma(\omega) = \omega_2\omega_3\omega_4\ldots$. We define the sequence of groups $G_n,\ n = 1, 2, \ldots$ by $G_n = G_{\sigma^{n-1}(\omega)}$ and denote by $a,\ b_n,\ c_n$ and d_n the respective generators. In particular $G_1 = G_\omega$.

For each n, consider the subgroup H_n of G_n consisting of all elements representable by a word with an even number of factors a. Then $|G_n : H_n| = 2$ and H_n is generated by $b_n,\ c_n,\ d_n,\ ab_na,\ ac_na$ and ad_na. Every element of H_n leaves set-wise invariant the left and right halves of the tree.

Restricting the action of H_n from the whole of the tree to the two halves induces two homomorphisms, $\phi_0^{(n)}$ and $\phi_1^{(n)}$ of H_n. These act on the elements of H_n according to Table 2.1 and play a key role in the word problem solving algorithm.

Table 2.1: The actions of $\phi_0^{(n)}$ and $\phi_1^{(n)}$. Here $\widetilde{u_n}$ (respectively $\widetilde{v_n},\ \widetilde{w_n}$) denotes a if the n^{th} digit of the sequence U (respectively $V,\ W$) is S, and 1 otherwise

	b_n	c_n	d_n	ab_na	ac_na	ad_na
$\phi_0^{(n)}$	$\widetilde{u}_n$	$\widetilde{v}_n$	$\widetilde{w}_n$	b_{n+1}	c_{n+1}	d_{n+1}
$\phi_1^{(n)}$	b_{n+1}	c_{n+1}	d_{n+1}	$\widetilde{u}_n$	$\widetilde{v}_n$	$\widetilde{w}_n$

Therefore we have $\phi_0^{(n)} \cong G_{n+1}$ and $\phi_1^{(n)} \cong G_{n+1}$ so that

$$|G_\omega| = |G_1| = 2|H_1| \geq 2|G_2| = 2^2|H_2| \geq \ldots \ .$$

Consequently, $|G_\omega| = \infty$.

Using the theory discussed above, we are now in a position to describe the *Word Problem Solving Algorithm* (Algorithm 1).

While running the algorithm, a finite tree will be created having the word F as root and depth at most $\lceil \log_2 \partial(F) \rceil$ (see Theorem 3.1.2, Chapter 3). Each round of the algorithm corresponds to a different level of the tree. Left branching occurs after

Algorithm 1 Word Problem Solving Algorithm (W.P.S.A.).

Given a word F and a sequence ω, in order to decide whether F is equal to 1 in G_ω we do the following:

1^{st} **Round**

1: Find $\partial_a(F)$. **If** it is odd **then** $F \neq 1_{G_\omega}$ and the algorithm **terminates**.

2: Reduce F to obtain F_r^1 (the index denotes the current round). **If** it is equal to 1_{G_ω} **then** so is word F and the algorithm **terminates**. Note that $\partial_a(F_r^1)$ will also be even.

3: Apply $\phi_0^{(1)}$ and $\phi_1^{(1)}$ to F_r^1 to obtain two words F_0^2 and F_1^2 of length at most $\left\lceil \frac{\partial(F_r^1)}{2} \right\rceil$ (see Theorem 3.1.1, Chapter 3).

4: Proceed to next round.

i^{th} **Round**

Each word from the previous round which is not equal to 1 yields, in its third step, two new words. Therefore, at most 2^i words are obtained to be used in this round. However, the total length of these words is bounded by the length of the word they originated from, which is less than the length of the input word F.

1: Find $\partial_a\left(F_0^i\right), \ldots, \partial_a\left(F_k^i\right)$, $k \leq 2^i$. **If** any one of them is odd **then** $F \neq 1_{G_\omega}$ and the algorithm **terminates**.

2: Reduce $F_0^i, \ldots, F_k^i$, where $k \leq 2^i$, to obtain $F_{r_0}^i, \ldots, F_{r_k}^i$. For each one of them which is equal to $*' \in \{1,\ b_\omega,\ c_\omega,\ d_\omega\}$, an *end node* has been reached. **If** an end node has been reached for all k words, **then** the algorithm **terminates**. $F = 1_{G_\omega}$ if and only if all the end nodes are equal to 1. Otherwise, $F \neq 1_{G_\omega}$.

3: Apply $\phi_0^{(i)}$ and $\phi_1^{(i)}$ to any of $F_{r_0}^i, \ldots, F_{r_k}^i$ for which an end node has not been reached yet, to obtain at most 2^{i+1} words $F_0^{i+1}, \ldots, F_{2k}^{i+1}$. These will have lengths at most $\left\lceil \frac{\partial(F_{r_0}^i)}{2} \right\rceil, \ldots, \left\lceil \frac{\partial(F_{r_k}^i)}{2} \right\rceil$ respectively.

4: Proceed to the next round.

Since the lengths of the words obtained are decreasing, the algorithm will eventually terminate.

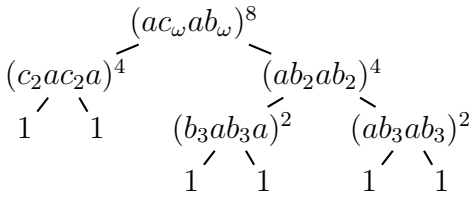


Figure 2.3: The finite tree obtained after applying the W.P.S.A. on the word $(ac_\omega ab_\omega)^8 = 1_{G_\omega},\ \omega = 012012\ldots$

application of $\phi_0^{(i)}$ and right branching after application of $\phi_1^{(i)}$ on words on the i^{th} level, with the resulting words as vertices. The word F will be equal to 1_{G_ω} if and only if all the end nodes of the tree are equal to 1. See Figure 2.3 for an illustration.

A feasible implementation of this algorithm constructs the finite tree depth-wise instead of level-wise, a branch at a time. In this way only $k \leq \lceil \log_2 \partial(F) \rceil$ words need to be stored instead of 2^k. The time complexity of the algorithm is $O(n \log n)$ in terms of the length $n = \partial(F)$ of the input word F.

Further information on the material in this subsection can be found in [9], [10] and [14].

2.2 Some Terminology in Cryptography

The word *cryptography* comes from the Greek words krypto ($\kappa\rho\acute{\upsilon}\pi\tau\omega$) which means hide, and grapho ($\gamma\rho\acute{\alpha}\phi\omega$) which means write. Formally [21], it is the study of mathematical techniques related to aspects of information security. Informally it can be referred to as the art and science of keeping messages secure.

In this section we will briefly describe only those topics within this vast subject area that will be useful for the exposition and understanding of our results in the following chapters. Further on introductory cryptography can be found in [18, 21, 26] and on feedback shift registers in [13].

A message, in our case a binary string, is called *plaintext* and after *encryption*, that is after it has been transformed into a form bearing no resemblance to its original, it becomes *ciphertext*. *Decryption* is the process of obtaining the original

plaintext from ciphertext. Encryption and decryption are dependent on a *key*, also a binary string. We will denote by $\mathcal{M}$ the set of plaintexts, by $\mathcal{C}$ the set of ciphertexts and by $\mathcal{K}$ and $\mathcal{K}'$ the set of encryption and decryption keys respectively.

An *encryption algorithm* is a function

$$\mathcal{E} : \mathcal{K} \times \mathcal{M} \to \mathcal{C}$$

used for encryption. Similarly, a *decryption algorithm* is a function

$$\mathcal{D} : \mathcal{K}' \times \mathcal{C} \to \mathcal{M}$$

used for decryption.

A *cryptosystem* consists of the set of plaintexts $\mathcal{M}$, the set of ciphertexts $\mathcal{C}$, the sets of encryption and decryption keys $\mathcal{K}$ and $\mathcal{K}'$, an encryption algorithm $\mathcal{E}$ and a decryption algorithm $\mathcal{D}$.

We denote the encryption of plaintext $m \in \mathcal{M}$ under an encryption key $k_e \in \mathcal{K}$ by

$$\mathcal{E}(k_e, m) = c \ ,$$

where ciphertext $c \in \mathcal{C}$. Analogously, we denote the decryption of ciphertext $c \in \mathcal{C}$ under the decryption key $k_d \in \mathcal{K}'$ by

$$\mathcal{D}(k_d, c) = m \ ,$$

where plaintext $m \in \mathcal{M}$.

For a cryptosystem to make practical sense, it is a requirement that for each plaintext $m \in \mathcal{M}$ and encryption key $k_e \in \mathcal{K}$ there exists a decryption key $k_d \in \mathcal{K}'$ such that

$$\mathcal{D}(k_d, \mathcal{E}(k_e, m)) = m \ . \tag{2.6}$$

An *asymmetric key* or *public key* cryptosystem is a cryptosystem whose encryption and decryption algorithms use different keys. In other words, in (2.6) we have that $k_e \neq k_d$. The encryption key k_e is made public so that anyone can encrypt

messages whereas the corresponding decryption key k_d is kept private. For the cryptosystem to be secure, it must be computationally infeasible to deduce the decryption (or *private*) key k_d from the encryption (or *public*) key k_e.

Symmetric key or *private key* cryptosystems use the same keys for encryption and decryption. In particular, in (2.6) we have that $k_e = k_d$. Such cryptosystems are divided into two main categories, namely *block ciphers* and *stream ciphers*.

2.2.1 Block Ciphers

A block cipher is a cryptosystem in which every message is broken in strings (called *blocks*) of fixed length N before encryption and decryption. In case the last block of the message is of smaller length, it is completed with, for example, appending zeroes. These blocks constitute $\mathcal{M}$ and $\mathcal{C}$. The key is also a string of some fixed length K. Common values for K are 64, 128, 192 and 256, and for N 64 and 128.

The majority of modern block ciphers operate by iterating their internal functions a number of times, each iteration being called a *round*. A block cipher's structure contains the following components:

Key schedule. Its purpose is to spread the effect of the user defined key in all rounds by generating from it the *round keys*, that is a different key to be used in each round.

The substitution layer. It is responsible for applying a nonlinear substitution, often called an *S-box*, to the text block. The gain is a nonlinear and hard to analyse connection between plaintext and ciphertext. This idea originates from and describes the concept of *confusion* as explained by Shannon in [29].

The permutation layer. It mixes the text block contents together using a function, usually linear. The idea behind this layer is that a change in a single digit of a text block will cause, after application of this layer, a change in almost all the digits of the block and implements the concept of *diffusion* proposed by Shannon [29].

The key mixing layer. It is addition (modular or XORing) of the key to the text blocks. This layer serves the key dependency of the ciphertext.

The binary operation *XOR* (eXclusive OR), usually denoted by $\oplus$ is defined as follows:

$$0 \oplus 0 = 0,\ 0 \oplus 1 = 1,\ 1 \oplus 0 = 1 \text{ and } 1 \oplus 1 = 0\ .$$

2.2.2 Stream Ciphers

In contrast to block ciphers, modern stream ciphers encrypt single bits of plaintext under a time-varying key, known as *keystream*. Crucial role in the design of keystream generators for stream ciphers is played by *feedback shift registers* (FSRs). A length-n feedback shift register consists of n *stages* numbered left to right from 0 to $n-1$. The contents of each stage are called *states* and consist of a single bit each. The output of the register is calculated as follows: First we set $st_{new} = f(st_0, st_1, \ldots, st_{n-1})$, where function $f : \{0,1\}^n \to \{0,1\}$ and st_l is state l of the FSR, $0 \leq l \leq n-1$. Such a function is called *boolean*. After this, st_0 is sent as the first output bit and stage 0 becomes empty. Then, for $1 \leq l \leq n-1$, state st_l is moved to stage $l-1$. Finally, st_{new} is entered in stage $n-1$ to be the new state st_{n-1}. See Figure 2.4 for an illustration.

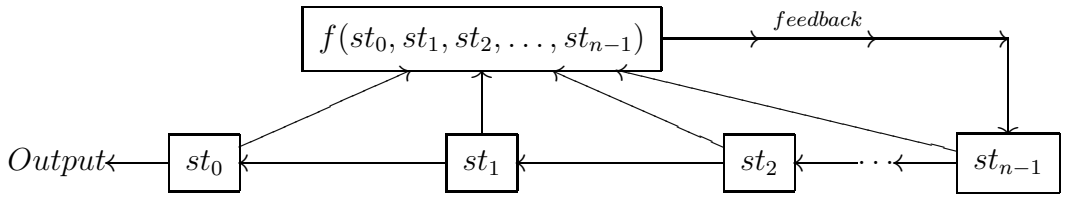


Figure 2.4: A feedback shift register

An FSR with feedback function $f(s_{j-1}, s_{j-2}, \ldots, s_{j-L})$ is said to be *non-singular* if and only if each and every of its possible output sequences is periodic. This happens if and only if f is of the form $f = s_{j-1} \oplus g(s_{j-2}, \ldots, s_{j-L})$ for some boolean function g.

Depending on whether the function f is linear or nonlinear, we have linear or nonlinear feedback shift registers (LFSRs and NLFSRs respectively). The majority of stream ciphers use LFSRs since they are the most studied and exhibit nice and mathematically controllable properties. NLFSRs however, although more difficult to analyse, generate more complicated keystreams, a property advantageous for cryptographic purposes.

2.2.3 Cryptanalysis

Cryptanalysis is the study of mathematical techniques for attempting to defeat cryptographic techniques. In other words, it is the science of *breaking* the ciphertext, that is recovering from it the plaintext or the encryption/decryption key, without access to the key. Below we list the different categories of cryptanalytic attacks on cryptosystems.

Ciphertext only. This is an attack in which the attacker has knowledge of some ciphertexts only, without any further information.

Known plaintext. With this attack, the attacker has access to plaintext/ciphertext pairs.

Chosen plaintext. In this kind of attack, the cryptanalyst has access to several pairs of plaintexts of his choice and their corresponding ciphertext before he attempts to break the ciphertext.

Adaptive chosen plaintext. This is a similar attack to the chosen plaintext attack where the attacker is allowed to choose the plaintexts after analyzing previously chosen plaintext/ciphertext pairs.

Chosen ciphertext. This attack is similar to a chosen plaintext attack. This time though, the attacker chooses the ciphertexts instead of the plaintexts.

Adaptive chosen ciphertext. It is an attack similar to adaptive chosen ciphertext attack, with plaintexts swapped with ciphertexts.

Exhaustive Key Search is a ciphertext only attack where the attacker tries to obtain the plaintext by decrypting the ciphertext with all possible decryption keys. For this reason, the set of keys $\mathcal{K}$ and $\mathcal{K}'$ have to be sufficiently large to avoid such an attack.

Unlike exhaustive key search, an attack applicable to all cryptosystems, several cryptanalytic attacks have been designed specifically for use against block ciphers. These are the following:

Differential Cryptanalysis, Linear Cryptanalysis, Higher Order Differentials, Truncated Differentials, Differential-Linear Attack, Interpolation Attack, Integral Cryptanalysis, Related Key Attacks, Key Schedule Attacks, The Square Attack, The Boomerang Attacks, Multiset Attacks, Slide Attacks and Slide with a Twist Attack and Weak Keys.

2.3 Number Theory

In this section we give some number theoretic results that we are going to use in Chapter 5. For more information we refer the reader to any text book on number theory.

Definition 2.3.1 *The Möbius function $\mu(n)$ is defined for any positive integer n by:*

$$\mu(n) = \begin{cases} 1 & \text{if } n = 1 \\ 0 & \text{if } n \text{ has repeated prime facors} \\ (-1)^{\nu} & \text{if } n \text{ is square free and has } \nu \text{ different prime factors} \end{cases} .$$

Definition 2.3.2 *An arithmetical function is a function $f : \mathbb{N} \to \mathbb{C}$.*

Theorem 2.3.3 (Möbius Inversion Formula) *If F(n) is an arithmetical function and*

$$G(n) = \sum_{d|n} F(d) \ ,$$

then inversely

$$F(n) = \sum_{d|n} \mu(d) G\left(\frac{n}{d}\right) \quad .$$

Definition 2.3.4 *The Euler's totient function $\phi(n)$ is defined on any positive integer n as the number of positive integers less than or equal and coprime to (that is having no common factors with) n.*

2.4 The Garzon–Zalcstein Public Key Cryptosystem

In 1991, Garzon and Zalcstein proposed a public key cryptosystem (see Section 2.2 for definition) based on the word problem of the Grigorchuk groups [8]. In this section we will describe it and discuss some of its security issues.

2.4.1 Description.

The public key cryptosystem proposed in [8] is the following:

Alice chooses an efficient procedure which generates a ternary sequence ω such that at least two of its digits repeat infinitely often. Recall that in this case G_ω is not finitely presentable (see Subsection 2.1.2). This sequence ω is her **private key**.

She then publishes a finite subset of relators (that is words equal to 1 in G_ω) and two words $w_0,\ w_1$ representing distinct group elements of G_ω. These comprise the **public key**.

Bob encrypts a bit $i \in \{0, 1\}$ of his message as a word w_i^* obtained from w_i by a sequence of random additions and/or deletions of relators as found in the public key. Concatenation of the encrypted bits yields the encrypted message (with a separator between the encryption of successive bits). He then sends the encrypted message to Alice.

Alice decrypts the message by checking whether w_i^* is equal to w_0 or w_1. This can be done by checking whether, for example, $w_0^{-1} w_i^* = 1_{G_\omega}$ using the W.P.S.A..

2.4.2 Security Claims

When discussing the security of their cryptosystem, Garzon and Zalcstein [8, Sect. 4] claim that the public key will not contain enough information to uniquely determine the private key ω. They also claim that in order to establish in polynomial time that a guess of a key is correct, it is necessary to follow a chosen plaintext attack.

Chapter 3

Cryptanalysis of the Garzon–Zalcstein Public Key Cryptosystem

This chapter is devoted to the cryptanalysis of the public key cryptosystem proposed by Garzon and Zalcstein and described in Section 2.4. A first attempt of cryptanalysis based on exhaustive search was made by Hofheinz and Steinwandt in [31]. Our results however, are based on the properties of the Grigorchuck groups.

We begin with some auxiliary results (Section 3.1) that allow us to present the cryptanalytic attack in Section 3.2. Section 3.3 illustrates the cryptanalysis by means of an example before our conclusions in Section 3.4. The results of this chapter appear in [24].

3.1 Auxiliary Results

The notation used in this section has been carried over from Section 2.1. To avoid ambiguity, from now on, when we say a word F is given, we refer to it as a word in the basis group Γ. We begin with a small theorem, whose proof is trivial:

Theorem 3.1.1 *Given a sequence ω, let F be a reduced word in G_ω with even*

$\partial_a(F)$. *If* $\partial_a(F) < \partial_{b,c,d}(F)$ (*respectively* $>, =$) *then, after a successful round of the W.P.S.A. we get two words of length at most* $\left\lceil \frac{\partial(F)}{2} \right\rceil$ $\left(\text{respectively } \left\lfloor \frac{\partial(F)}{2} \right\rfloor, \frac{\partial(F)}{2}\right)$.

Proof. When $\partial_a(F) < \partial_{b,c,d}(F)$ then the word is of the same form as

$$*a*a\ldots a*a*$$

and of odd length. It can then be broken into a product of quadruples $*a*a$ of total length $\partial(F)-1$, and a single $*$ at the end. Under the action of $\phi_0^{(n)}$ and $\phi_1^{(n)}$ each quadruple goes into $*$, $*a$ or $a*$ and the asterisk at the end goes to $*$, a or 1. There are $\frac{\partial(F)-1}{4}$ quadruples, and so the maximum possible length of the word obtained is

$$2\cdot\frac{\partial(F)-1}{4}+1=\frac{\partial(F)-1}{2}+1=\frac{\partial(F)+1}{2}=\left\lceil\frac{\partial(F)}{2}\right\rceil \quad .$$

Now, when $\partial_a(F) = \partial_{b,c,d}(F)$ then the word is either of the same form as

$$*a*a\ldots a*a \quad \text{or} \quad a*a\ldots a*a*$$

and of even length. This length must be a multiple of 4 because otherwise $\partial_a(F)$ will be odd and the W.P.S.A. will fail. Therefore, it can then be broken into a product of quadruples $*a*a$ of total length $\partial(F)$. Under the action of $\phi_0^{(n)}$ and $\phi_1^{(n)}$ each quadruple goes into $*$, $*a$ or $a*$. There are $\frac{\partial(F)}{4}$ quadruples, and so the maximum possible length of the word obtained is

$$2\cdot\frac{\partial(F)}{4}=\frac{\partial(F)}{2} \quad .$$

Finally, when $\partial_a(F) > \partial_{b,c,d}(F)$ then the word is of the same form as

$$a*a\ldots a*a*a$$

and of odd length. It can then be broken into a product of quadruples $a*a*$ of total length $\partial(F)-3$, and a single triple $a*a$ at the end. Under the action of $\phi_0^{(n)}$ and $\phi_1^{(n)}$ each quadruple goes into $*$, $*a$ or $a*$ and the triple at the end goes to $*$, a or 1. There are $\frac{\partial(F)-3}{4}$ quadruples and so the maximum possible length of the word obtained is

$$2\cdot\frac{\partial(F)-3}{4}+1=\frac{\partial(F)-3}{2}+1=\frac{\partial(F)-1}{2}=\left\lfloor\frac{\partial(F)}{2}\right\rfloor \quad .$$

■

Theorem 3.1.2 *Given a sequence ω, for successful application of the W.P.S.A. to a word F in G_ω, at most only the first $\lceil \log_2 \partial(F) \rceil$ of its digits are needed. This number is also the maximum depth of the tree obtained when running the W.P.S.A..*

Proof. For the substitution of the symbols $\widetilde{u}_i$, $\widetilde{v}_i$ and $\widetilde{w}_i$ determining the action of $\phi_0^{(i)}$ and $\phi_1^{(i)}$ during the i^{th} round of the W.P.S.A., the i^{th} digit of the ternary sequence ω is needed. Therefore, for successful application of the W.P.S.A., the maximum number of digits needed is equal to the maximum number of rounds possible. Denote this number by n. This maximum is attained if the algorithm terminates because an end node has been reached for all obtained words on the n^{th} level.

Suppose that n rounds of the W.P.S.A. take place. By Theorem 3.1.1, after the first round of the W.P.S.A. the maximum length of the words obtained is $\left\lceil \frac{\partial(F)}{2} \right\rceil$. Note that these words will have an even number of occurrences of the symbol a, otherwise the algorithm will terminate before all n rounds take place. After step 2 of the second round of the algorithm, the words will also be in reduced form. So, again by Theorem 3.1.1, after the second round of the algorithm we will obtain words of length at most $\left\lceil \left\lceil \frac{\partial(F)}{2} \right\rceil \cdot \frac{1}{2} \right\rceil = \left\lceil \frac{\partial(F)}{2^2} \right\rceil$. Continuing this way we see that after the n^{th} round, the length of any words we obtain will not exceed $\left\lceil \frac{\partial(F)}{2^n} \right\rceil$.

By assumption, after the n^{th} round of the algorithm, end nodes have been reached for all words on the n^{th} level. Recall that each end node is a word of length 1. Thus, n is an integer such that $\left\lceil \frac{\partial(F)}{2^n} \right\rceil = 1$ or, equivalently, $\partial(F) < 2^n$. Taking logarithms we get $\log_2 \partial(F) < n$ which implies $n = \lceil \log_2 \partial(F) \rceil$.

Since each round corresponds to a different level of the tree obtained when running the algorithm, n is also the maximum depth of the tree. ∎

Denote *by M_F^ω the exact number of digits of the sequence ω needed to run the W.P.S.A. on a given word F in G_ω. Clearly, $M_F^\omega \leq \lceil \log_2 \partial(F) \rceil$.*

Corollary 3.1.3 *Suppose that a word F is a relator in the group G_ω defined by a sequence ω. Then it is also a relator in all groups defined by sequences that share the first M_F^ω digits with the sequence ω and differ in any of the rest.*

Proof. By definition, only the first M_F^ω digits of the sequence ω are needed to verify that the word F is a relator in G_ω. ∎

Denote *by* Ω_F *the set of all sequences* ω *such that a given word* F *is a relator in* G_ω *and by* M_F *the number* $\min\{M_F^\omega \mid \omega \in \Omega_F\}$.

Clearly, by Corollary 3.1.3, Ω_F contains sequences that share the first M_F^ω digits with a sequence $\omega \in \Omega_F$ and differ in any of the rest.

Remark 3.1.4 Ω_F *might also contain sequences that are different in at least one of the first* M_F *digits.*

One example is the following:

Example 3.1.5 *Let* $F = (ac_\omega ab_\omega)^8$. *It can be checked using the W.P.S.A. that* F *is a relator for all ternary sequences having 012 and 021 as their first three digits.*

Theorem 3.1.6 *There exists an algorithm which, given a word* F, *finds the first* M_F^ω *digits of all sequences* $\omega \in \Omega_F$, *if any.*

Proof. Let ω_* be an arbitrary infinite ternary sequence and let F be the given word. The algorithm is as follows:

Algorithm 2 Given a word F, find the first M_F^ω digits $\forall \omega \in \Omega_F$.

1: Set $\omega_1 = 0$.

2: In case $\omega_1 = 3$, **terminate** the algorithm. Run the first round of the W.P.S.A. on F. **If** it fails its first step **or** if the word F is equal to 1_{G_ω} $(\omega = \omega_1\omega_*)$, **then** increase ω_1 by one and repeat step. In the latter case print out ω_1. **Else**, set $\omega_2 = 0$ and proceed to step **3**.

i: $(i \geq 3)$. **If** $\omega_{i-1} = 3$, delete it, increase ω_{i-2} by one and return to step ***i*-1**. Run the i^{th} round of the W.P.S.A. on F. **If** it fails its first step **or** if the word F is equal to 1_{G_ω} $(\omega = \omega_1 \ldots \omega_{i-1}\omega_*, i-1 \leq n)$, **then** increase ω_{i-1} by one and repeat step. In the latter case, print out $\omega_1 \ldots \omega_{i-1}$. **Else**, set $\omega_i = 0$ and proceed to step ***i*+1**. The algorithm terminates when it eventually returns to step 2 with $\omega_1 = 3$.

Each output of Algorithm 2 is obtained immediately after a successful application of the W.P.S.A.. Thus only the first M_F^ω digits of all sequences $\omega \in \Omega_F$ are printed. Clearly, if no sequences are printed, Ω_F is empty. ■

An upper bound for the number of outputs of the algorithm, though never attained, is $3^{\lceil \log_2 \partial(F) \rceil}$, which corresponds to all ternary sequences of length $\lceil \log_2 \partial(F) \rceil$. Note that $3^{\lceil \log_2 \partial(F) \rceil} \leq \partial(F)^{\lceil \log_2 3 \rceil} = \partial(F)^2$. Thus, the maximum running time of the algorithm is bounded by $O\left(\partial(F)^3 \log \partial(F)\right)$. It is therefore computationally feasible to run the algorithm, even for large $\partial(F)$.

3.2 Cryptanalysis

We begin with an important observation, omitted by Garzon and Zalcstein:

Remark 3.2.1 *It was not mentioned in [8] that the public key words w_0 and w_1 must share the following property, otherwise comparing them with the ciphertext becomes trivial: If $\partial_a(w_0)$ is even (or odd) then so should $\partial_a(w_1)$ be.*

The reason for this is that relators have an even number of letters a and so, if for example $\partial_a(w_0)$ is even, addition of relators will result to a ciphertext word with an even number of a's. If $\partial_a(w_1)$ is odd then we can immediately tell that $w_i^* = w_0$.

In the sequel we will exploit the information provided by each relational word in the public key of the cryptosystem. Suppose there are n relators in the public key, namely $r_1, \ldots, r_n$. Remark 3.1.4 suggests that each one of these relators could be a relator in groups defined by other sequences as well, not just by the private key. What we want to find are sufficiently many digits of a sequence ω such that all of $r_1, \ldots, r_n$ are indeed relators in G_ω, but $w_0 \neq w_1$ (or equivalently $w_0^{-1} w_1 \neq 1_{G_\omega}$). Then, by Corollary 3.1.3, we can just complete the sequence with random digits to obtain a key enabling us to decrypt messages.

Using Algorithm 2 on $w_k = w_0^{-1} w_1$, we firstly obtain the first $M_{w_k}^\omega$ digits of every sequence $\omega \in \Omega_{w_k}$. This will be a list of unacceptable initial segments of sequences. Note that Ω_{w_k} could be empty in which case the list would be empty as

well. The sequences in Ω_{w_k} are unacceptable because for them $w_j^{-1} w_i^* = 1_{G_\omega}$, where $i, j \in \{0, 1\}$, and hence we cannot distinguish between w_0 and w_1, as required for decryption.

We then apply the same algorithm on every r_i to obtain the first $M_{r_i}^{\omega}$ digits of all sequences $\omega \in \Omega_{r_i}$. In this way we get n lists of initial segments of sequences.

After obtaining the n lists, we compare them in order to find all initial segments which are common. This will be our list of candidates for a key. Notice that this list cannot be empty since it must contain the initial segment of Alice's private key. The number of digits of these common initial segments will be $\max\{M_{r_1}^{\omega}, \ldots, M_{r_n}^{\omega}\}$.

In case we have more than one candidates, we decide the suitability of the first candidate to be used for decryption by looking at the list of unacceptable initial segments of sequences. Let p denote the number of digits of the candidate. There are three possible cases:

Case 1: The unacceptable initial segments list contains at least one member with more digits than the candidate, with its first p digits being the candidate itself. If more than one such members exist, we consider the one with the fewest digits. Suppose this initial segment has k digits, $k > p$. We can then make the candidate into a suitable sequence by adding arbitrary digits. However, we must ensure that the first $k - p$ arbitrary digits we add will not coincide with the last $k - p$ digits of this particular unacceptable initial segment or any others sharing the first p digits with it. If this is impossible to perform, the candidate is unsuitable and we should test the next candidate for suitability.

Case 2: An unacceptable initial segment has k digits, $k \leq p$, and these digits coincide with the first k digits of the candidate. Then, this candidate is unsuitable and we should test the next candidate for suitability.

Case 3: The unacceptable initial segments list is empty or none of the previous two cases occur. In this case the candidate is suitable.

As mentioned above, there is at least one suitable candidate, namely the initial segment of Alice's original private key. We can make this into an infinite sequence

by adding random digits.

This is all we need for decryption since, on applying the W.P.S.A. to the word $w_j^{-1}w_i^*$ where $j,\, i \in \{0,1\}$, the relators would vanish and we would get 1_{G_ω} only if $j = i$.

3.3 Example

In this section we give a worked example of our cryptanalysis. For the sake of brevity, we will omit the subscript ω from the generators b, c and d.

Suppose **Alice** publishes the following **public key**:

$r_1 = (ab)^4$, $r_2 = (acab)^8$, $r_3 = (bada)^8$,

$w_0 = (bacabacacaca)^2 bacab$,

$w_1 = acacacabacabacacaca$.

Note that $\partial_a(w_0)$ and $\partial_a(w_1)$ are both even, in accordance to Remark 3.2.1.

Applying the Algorithm 2 on $w_k = w_0^{-1}\, w_1$ we get the list of **unacceptable** initial segments of sequences:

1. 01
2. 1
3. 21

Applying the algorithm on r_1, r_2 and r_3 we obtain 3 lists of initial segments of all sequences for which r_1, r_2 and r_3 are relators:

r_1	2							
r_2	012	021	101	11	121	202	212	22
r_3	00	010	020	102	120	202	212	22

We compare these in order to obtain the **candidates** list:

1. 202

2. 212

3. 22

Finally, by comparing the candidates and unacceptable lists, we deduce that by completing 202 (or 22) arbitrarily (e.g. with zeroes to get 202000...), we will be able to successfully decrypt all messages sent by **Bob**.

3.4 Conclusion

The Public Key Cryptosystem proposed in [8] is unsuitable for any practical implementations due to lack of security. This is because, as demonstrated in Section 3.2, the public key provides enough information to easily obtain keys behaving like the private key and which are suitable for successful decryption of ciphertexts.

Chapter 4

A Block Cipher Based on the Grigorchuk Groups

In this chapter we present a new block cipher based on the Grigorchuk groups. Throughout this chapter, we make use of the material discussed in Chapter 2. We begin by viewing the Grigorchuk groups from a different angle in Section 4.1. In the same section we will describe a function used in the encryption/decryption process of our cipher before presenting the cipher itself in Section 4.2. The security issues of the cipher are discussed theoretically in Section 4.3 and we finish with our conclusions in Section 4.4.

4.1 Auxiliary Results

4.1.1 Elements of G_ω acting as permutations

On the n^{th} level of the infinite binary tree we have 2^n nodes which we can label from 1 to 2^n. Each of these nodes consists of a binary sequence of length n . By definition of the generators of G_ω, they can act on each one of the nodes and therefore induce permutations of 2^n elements. The permutation group induced by G_ω when restricted on the n^{th} level of the infinite binary tree, is a subgroup of the Symmetric Group S_{2^n}. We shall denote it by $S^n_{G_\omega}$.

For example, on the fourth level of the infinite binary tree we have $2^4 = 16$ nodes. We can label these from 1 to 16:

$$\left|\begin{array}{cccccccccccccccc} 0000 & 0001 & 0010 & 0011 & 0100 & 0101 & 0110 & 0111 & 1000 & 1001 & 1010 & 1011 & 1100 & 1101 & 1110 & 1111 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 \end{array}\right|$$

If we apply the generator a of G_ω on each node we will get the following:

$$\left|\begin{array}{cccccccccccccccc} 1000 & 1001 & 1010 & 1011 & 1100 & 1101 & 1110 & 1111 & 0000 & 0001 & 0010 & 0011 & 0100 & 0101 & 0110 & 0111 \\ 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array}\right|$$

Therefore the generator a represents the following permutation of 16 elements:

$$a = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 \\ 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{pmatrix} .$$

In a similar way, the other three generators b_ω, c_ω and d_ω induce permutations as well, depending on the first three digits of the sequence ω (since the length of the binary sequence in each node is 4). The first digit ω_0 affects the first half of the permutation, ω_1 the third quarter and ω_2 the rest.

If the action of a generator under ω_0 is S, applying it on the first 8 nodes gives:

$$\left|\begin{array}{cccccccc} 0000 & 0001 & 0010 & 0011 & 0100 & 0101 & 0110 & 0111 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array}\right|$$

$$\downarrow$$

$$\left|\begin{array}{cccccccc} 0100 & 0101 & 0110 & 0111 & 0000 & 0001 & 0010 & 0011 \\ 5 & 6 & 7 & 8 & 1 & 2 & 3 & 4 \end{array}\right|$$

Similarly, if the action of a generator under ω_1 is S then its application on nodes 9 to 12 gives:

$$\left|\begin{array}{cccc} 1000 & 1001 & 1010 & 1011 \\ 9 & 10 & 11 & 12 \end{array}\right| \rightarrow \left|\begin{array}{cccc} 1010 & 1011 & 1000 & 1001 \\ 11 & 12 & 9 & 10 \end{array}\right|$$

Finally, if the action of a generator under ω_2 is S then, if we apply it on nodes 13 and 14, we get:

Table 4.1: The permutations induced by generators b_ω, c_ω and d_ω

	$\omega_0 = 0$	$\omega_1 = 0$	$\omega_2 = 0$	
b_ω	5 6 7 8 1 2 3 4	11 12 9 10	14 13	15 16
c_ω	5 6 7 8 1 2 3 4	11 12 9 10	14 13	15 16
d_ω	1 2 3 4 5 6 7 8	9 10 11 12	13 14	15 16
	$\omega_0 = 1$	$\omega_1 = 1$	$\omega_2 = 1$	
b_ω	5 6 7 8 1 2 3 4	11 12 9 10	14 13	15 16
c_ω	1 2 3 4 5 6 7 8	9 10 11 12	13 14	15 16
d_ω	5 6 7 8 1 2 3 4	11 12 9 10	14 13	15 16
	$\omega_0 = 2$	$\omega_1 = 2$	$\omega_2 = 2$	
b_ω	1 2 3 4 5 6 7 8	9 10 11 12	13 14	15 16
c_ω	5 6 7 8 1 2 3 4	11 12 9 10	14 13	15 16
d_ω	5 6 7 8 1 2 3 4	11 12 9 10	14 13	15 16

$$\begin{vmatrix} 1100 & 1101 \\ 13 & 14 \end{vmatrix} \rightarrow \begin{vmatrix} 1101 & 1100 \\ 14 & 13 \end{vmatrix}$$

The last 2 nodes remain unaffected.

Table 4.1 summarises all possible permutations induced by the three generators b_ω, c_ω and d_ω.

Each column has only two different entries, one being the identity permutation of the corresponding digits. We can therefore have $2^3 = 8$ different permutations per generator. This also means that $S^4_{G_\omega}$ can be easily implemented by keeping in memory only 3 arrays: one of length 8, one of length 4 and one of length 2.

Example 4.1.1 *If* $\omega_0 = 1$, $\omega_1 = 0$ *and* $\omega_2 = 2$ *then*

$$c_\omega = \begin{pmatrix} 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10\ 11\ 12\ 13\ 14\ 15\ 16 \\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 11\ 12\ 9\ 10\ 14\ 13\ 15\ 16 \end{pmatrix} .$$

Clearly, any word $g \in G_\omega$ induces a permutation equal to the composition of the permutations induced by each of its letters. In the sequel, we will denote this as σ_g.

Proposition 4.1.2 ([11]) $|S^n_{G_\omega}| = 2^{5\cdot 2^{n-3}+2},\ \forall n \geq 4$, *where* $\omega = 012012...$.

4.1.2 A Recursive Function Based on Permutations

Definition 4.1.3 *Given a pair* (a, b), *we define* $\overline{(a, b)} = a$.

Definition 4.1.4 *Given an element* $\sigma \in S_n$ *where* $n \geq 2$, *field* F *and a set* V *of* n k*-dimensional vectors* $v_i,\ 1 \leq i \leq n$, *with entries in* F *such that their order of appearance in* V *is random but fixed and determined by the subscript* $i \in \mathbb{Z}_n$, *we define function* $\tau_\sigma : F^k \times \mathbb{Z}_n \to F^k \times \mathbb{Z}_n$ *recursively as follows:*

$$\tau_\sigma(v_i, i) = \begin{cases} \left(v_i + v_{\sigma(i)} \ , \ i \right) & \text{if } \ i < \sigma(i) \\ \left(v_i \ , \ i \right) & \text{if } \ i = \sigma(i) \\ \left(v_i + \overline{\tau_\sigma\left(v_{\sigma(i)}, \sigma(i)\right)} \ , \ i \right) & \text{if } \ i > \sigma(i) \end{cases} .$$

Proposition 4.1.5 *Function* τ_σ *is invertible with* $\tau_\sigma^{-1}(\tau_\sigma(v_i, i)) = (v_i, i)$, *where* $\tau_\sigma^{-1} : F^k \times \mathbb{Z}_n \to F^k \times \mathbb{Z}_n$ *is recursively defined as follows:*

$$\tau_\sigma^{-1}(v_i, i) = \begin{cases} \left(v_i - v_{\sigma(i)} \ , \ i \right) & \text{if } \ i < \sigma(i) \\ \left(v_i \ , \ i \right) & \text{if } \ i = \sigma(i) \\ \left(v_i - \overline{\tau_\sigma\left(v_{\sigma(i)}, \sigma(i)\right)} \ , \ i \right) & \text{if } \ i > \sigma(i) \end{cases} .$$

Proof. We prove that the function τ_σ is bijective, and hence invertible.

One-to-one: The function τ_σ is defined on pairs (v_i, i), where $v_i \in F^k$ and $i \in \mathbb{Z}_n$ denotes the position of appearance of v_i in the set V. $\tau_\sigma(v_i, i) = (v, i)$, where $v \in F^k$, and $i \in \mathbb{Z}_n$ again denotes the position of appearance of v_i in the set V. Therefore, if $\tau_\sigma(v_i, i) = (v, i) = \tau_\sigma(v_j, j)$, then we must have $i = j$ and hence the function is one to one.

Onto: We have seen that the output of the function τ_σ is a pair (v, i), where $v \in F^k$ and $i \in \mathbb{Z}_n$. Therefore its range is $F^k \times \mathbb{Z}_n$.

We now prove that τ_σ^{-1} is the inverse. For the case $i = \sigma(i)$ the proof is trivial. When $i < \sigma(i)$ we have

$$\begin{aligned}\tau_\sigma^{-1}\left(\tau_\sigma\left(v_i,i\right)\right) &= \tau_\sigma^{-1}\left(\ v_i+v_{\sigma(i)}\ ,\ i\ \right)\\ &= \left(\ v_i+v_{\sigma(i)}-v_{\sigma(i)}\ ,\ i\ \right)\\ &= \left(v_i,i\right)\ .\end{aligned}$$

When $i>\sigma\left(i\right)$ we have

$$\begin{aligned}\tau_\sigma^{-1}\left(\tau_\sigma\left(v_i,i\right)\right) &= \tau_\sigma^{-1}\left(\ v_i+\overline{\tau_\sigma\left(v_{\sigma(i)},\sigma\left(i\right)\right)}\ ,\ i\ \right)\\ &= \left(v_i+\overline{\tau_\sigma\left(v_{\sigma(i)},\sigma\left(i\right)\right)}-\overline{\tau_\sigma\left(v_{\sigma(i)},\sigma\left(i\right)\right)},i\right)\\ &= \left(v_i,i\right)\ .\end{aligned}$$

■

Example 4.1.6 *Consider* $\sigma=\begin{pmatrix}1&2&3&4&5&6\\4&6&2&1&5&3\end{pmatrix}$ *together with 6 vectors* $v_1,\ldots,v_6\in F^k$. *Then,*

$$\begin{aligned}&\tau\left(v_1,1\right)=\left(v_1+v_4,1\right)\\&\tau\left(v_2,2\right)=\left(v_2+v_6,2\right)\\&\tau\left(v_3,3\right)=\left(v_3+\overline{\tau\left(v_2,2\right)},3\right)=\left(v_3+v_2+v_6,3\right)\\&\tau\left(v_4,4\right)=\left(v_4+\overline{\tau\left(v_1,1\right)},4\right)=\left(v_1+2v_4,4\right)\\&\tau\left(v_5,5\right)=\left(v_5,5\right)\\&\tau\left(v_6,6\right)=\left(v_6+\overline{\tau\left(v_3,3\right)},6\right)=\left(2v_6+v_2+v_3,6\right)\end{aligned}$$

The inverse is obtained recursively starting from $i=6$ *down to* $i=1$:

$$\begin{aligned}&\tau^{-1}\left(\tau\left(v_6,6\right)\right)=\left(2v_6+v_2+v_3-\overline{\tau\left(v_3,3\right)},6\right)=\left(v_6,6\right)\\&\tau^{-1}\left(\tau\left(v_5,5\right)\right)=\left(v_5,5\right)\\&\tau^{-1}\left(\tau\left(v_4,4\right)\right)=\left(v_1+2v_4-\overline{\tau\left(v_1,1\right)},4\right)=\left(v_4,4\right)\\&\tau^{-1}\left(\tau\left(v_3,3\right)\right)=\left(v_3+v_2+v_6-\overline{\tau\left(v_2,2\right)},3\right)=\left(v_3,3\right)\\&\tau^{-1}\left(\tau\left(v_2,2\right)\right)=\left(v_2+v_6-v_{\sigma(2)},2\right)=\left(v_2,2\right)\\&\tau^{-1}\left(\tau\left(v_1,1\right)\right)=\left(v_1+v_4-v_{\sigma(1)},1\right)=\left(v_1,1\right)\end{aligned}$$

It can be seen from the above that, if we know σ and $\tau\left(v_i,i\right)$ for all $0\leq i\leq n$ (since the definition of τ is recursive), we can determine each v_i.

It is also easy to see that in the case where $F = \mathbb{F}_2$, the field of two elements, we have $\tau = \tau^{-1}$, since subtraction is equal to addition modulo 2.

4.2 Description of the Block Cipher

In this section we describe our block cipher. The notation used has been carried over from Section 4.1.

4.2.1 Key Schedule

The key is given as a 128-bits binary sequence k_0 which we will use to construct another fifteen sequences $k_1 \dots k_{15}$, called round keys, of the same length to be used in the encryption/decryption process. Each sequence k_{i+1} is dependent on the sequence $k_i = b_0^i b_1^i \dots b_k^i \dots b_{127}^i$, where the exponent denotes the *number* of the sequence (that is its order of construction) and the subscript the number of the bit, $0 \leq i \leq 14$:

First, we set

$$w_r = \sum_{m=0}^{15} b_{7m+r+i}^i \; mod \; 3, \; 0 \leq r \leq 7 \; .$$

We then translate sequence k_i into a reduced word g_i of length 256 or 257 in the Grigorchuk group G_ω, where $\omega = \omega_0\omega_1 \dots \omega_7\omega_*$ with ω_* an arbitrary infinite ternary sequence, as follows ($0 \leq l \leq 127$):

- if $b_l^i = 0 \neq b_{(l+1 \bmod 128)}^i$ we write ab
- if $b_l^i = 1 \neq b_{(l+1 \bmod 128)}^i$ we write ac
- if $b_l^i = 0 = b_{(l+1 \bmod 128)}^i$ we write ad
- if $b_l^i = 1 = b_{(l+1 \bmod 128)}^i$ we write ab the first time, ac the second time and ad the third time.
- if $l = 127$ and $i = 1 \bmod 2$ we write a.

Next, we divide sequence k_i into 16 8-bit blocks, $k_0^i \dots k_{15}^i$, where the exponent denotes the number of the sequence and the subscript the number of the sub-block. Sequence k_{i+1} is obtained by applying the permutation induced by σ_{g_i} (see subsection 4.1.1) on these blocks.

In this way we obtain $k_0 \dots k_{15}$, $g_0 \dots g_{15}$ and $\sigma_{g_0} \dots \sigma_{g_{15}}$. For ease of notation, in the sequel we will write σ_i for σ_{g_i}.

Remark 4.2.1

- *All sequences must be pre-computed before encryption or decryption.*
- *There are* 3^{128} *words of the form* $a*a \dots a*a*$ *and length 256, with* $* \in \{b, c, d\}$. *Since the number of keys is* 2^{128}*, we have that* Keyspace $\subset$ Wordspace.
- *The choice of* ω_0 *to* ω_7 *is such that they depend on the whole of the round key and also on its number, thus having more probability for the sequence* ω *being significantly different in each round. The fact that only seven digits need to be defined is explained in Subsection 2.1.*
- *The translation of* k_i *into* g_i *has been chosen to provide as even a distribution of* b*'s,* c*'s and* d*'s as possible, and so that we obtain* 2^{128} *different words: two different round keys cannot yield the same words.*
- *The additional letter a, concatenated at the end of words every other round, has a purpose connected to the security of the cipher, as explained in Subsection 4.3.2.*

4.2.2 Encryption

The plaintext block P is 128-bits, which we split into 16 8-bit sub-blocks p_0 to p_{15}. Encryption consists of 16 rounds, starting from round 0. In each round, four operations take place. The resultant 128-bit block C after round 15 is the ciphertext.

Round j, $0 \leq j \leq 15$.

1. The first operation is the application of g_0 to g_{15}, as computed during key generation, to the 16 sub-blocks, each seen as one of the nodes on the 8^{th} level of the binary tree. This is done by applying g_i to sub-block p_i, $0 \leq i \leq 15$. This nonlinear operation can be regarded as a *key-dependent S-box*.

2. We permute the 16 sub-blocks according to σ_j.

3. We XOR key sub-block $k^i_{j+i \bmod 16}$ with sub-block p_i, where $0 \leq i \leq 15$.

4. Starting from $i = 0$, if $\sigma_j(i) \neq i$, we XOR sub-block p_i to sub-block $p_{\sigma_j(i)}$.

Remark 4.2.2 *The 4^{th} operation is equivalent to setting $p_i = \tau_{\sigma_j}(p_i, i)$, $0 \leq i \leq 15$ (Section 4.1.2).*

4.2.3 Decryption

We divide the 128-bit ciphertext block C into 16 8-bit sub-blocks c_0 to c_{15}. Decryption is the inverse of encryption so it will also require 16 rounds. In each round the same four operations as in encryption take place, but in reverse order.

Round j, $0 \leq j \leq 15$.

1. Starting from $i = 15$, if $\sigma_{15-j}(i) \neq i$, we XOR sub-block c_i to sub-block $c_{\sigma_{15-j}(i)}$.

2. We XOR key sub-block $k^i_{15-j+i \bmod 16}$ with sub-block c_i, where $0 \leq i \leq 15$.

3. We permute the 16 sub-blocks according to σ^{-1}_{15-j}.

4. Application of g_0^{-1} to g_{15}^{-1} to the 16 sub-blocks, done by applying g_i^{-1} to sub-block c_i, $0 \leq i \leq 15$.

Remark 4.2.3 *The 1^{st} operation is equivalent to setting $c_i = \tau^{-1}_{\sigma_j}(c_i, i)$, $15 \geq i \geq 0$ (Section 4.1.2).*

By looking at how the key schedule and operations of the cipher are defined, we can see that the key and block size can be increased at will. The only constraint is that they must be the same and also a power of two. The reason for this is that encryption and decryption are associated with the level nodes of the binary tree, the number of which is also a power of two.

4.3 Security of the Cipher

Since the invention of the first block cipher, cryptanalysts have devised various different methods of attack. The most important and powerful of these are differential [3] and linear [20] cryptanalysis.

The natural thing to do when designing a block cipher is to make it immune to all kinds of known cryptanalytic attacks. However, one must be careful that emphasis on resistance against these attacks does not result in other weaknesses.

Nevertheless, since our purpose is mainly the theoretical survey of the application of groups into the construction of block ciphers, we will, for the moment, omit the analysis of our cipher's resistance against the many existing cryptanalytic attacks.

We will therefore restrict ourselves in looking for flaws in our cipher's group theoretic properties. First we show that all round operations are necessary to achieve security. Then we discuss the key schedule and weak keys and explain the "merits" of our cipher against differential cryptanalysis. Similar S-box analysis based attacks are treated similarly and so for the moment are omitted.

4.3.1 Necessity of the Round Operations

As seen in Section 4.2, each round of encryption consists of four operations. Here, we will discuss the importance of each.

Evaluation of 1^{st} operation:

This is the key-dependent S-box, providing confusion (see Subsection 2.2.1) and resistance to cryptanalytic attacks requiring knowledge of the structure of the S-boxes.

Evaluation of 2^{nd} operation:

This operation allows the shuffling of the text blocks. It is important due to the particularity every other word g_i has, namely the additional letter a. In effect, this extra letter results in complementing the first digit of the text block during the first operation. Therefore the second operation makes sure this happens in all text blocks and not constantly in the same.

Evaluation of 3^{rd} operation:

This operation is the application of the user defined secret parameter on the text blocks. It provides protection even when the other operations fail. It makes, for example, the effect of "weak" keys negligible, as seen in subsection 4.3.2.

Evaluation of 4^{th} operation:

This is the diffusion layer (see Subsection 2.2.1) of the cipher. Without it, since the plaintext sub-blocks will not be mixed together, the cipher is vulnerable to a chosen-plaintext attack (see Subsection 2.2.3). For example, one can choose a plaintext consisting of 15 identical sub-blocks and 1 that is different. After encryption, the ciphertext will also contain 15 identical sub-blocks and 1 different, the position of which will reveal part of the total permutation induced by the cipher. Changing each time the plaintext position of the sub-block that is different, 2^4 chosen plaintexts can yield the total permutation induced by the cipher. Hence, we will know how each 8-bit block in a certain position (out of 16) encrypts under the key. Therefore, with 2^8 possible 8-bit blocks in each position and 2^4 different positions, we can create

a codebook with 2^{12} entries. This will enable us to decode all ciphertext without knowing the secret key.

4.3.2 Key Schedule & Weak Keys

The only suspicion for what might be termed a "weak key" arises in the case when g_i is equal to the identity in the corresponding Grigorchuk group, for some i, where $0 \leq i < 15$. Recall that identity elements have an even number of letters a, and by the key schedule algorithm, i has to be odd.

In such case $k_i = k_{i+1}$ but, due to the round constant in the choice of the digits of the sequence ω for round $i+1$, most probably we will be considering a different Grigorchuk group. In the unlikely case $g_i = 1_{G_\omega}$ in the new group as well or if the sequence ω is the same and we are considering the same Grigorchuk group, we will have $g_{i+1} = g_i a = a \neq 1_{G_\omega}$. As a result, k_{i+2} will be equal to k_i with the two halves swapped. If these two halves are the same, then we have that $k_i = k_{i+2} = 1_{G_\omega}$.

We now consider the effect such a key will have on encryption. In the first operation, only the plaintext sub-block p_i will remain unchanged. The second operation will have no effect only in the i^{th} round. However, even when this happens, the third operation, simple XORing of key sub-blocks, will not cancel out that of the previous round, due to the round constant in the choice of key sub-blocks to be XORed. Finally, since the fourth operation depends on the second, it will have no effect only in the i^{th} round.

After this discussion, it is safe to assume that the effect of weak keys is negligible with respect to the security of the cipher.

4.3.3 Differential Cryptanalysis

We first describe the basic principles of differential cryptanalysis as was discovered by Shamir and Biham in 1990 [3] to attack the Data Encryption Standard (DES).

Differential cryptanalysis is a chosen plaintext attack which studies the evolution

of the difference between two plaintexts through the cipher. Differences are normally regarded with respect to XOR but can be adjusted to the needs of the cipher under attack.

The only components of the cipher affecting these differences are the S-boxes. Each plaintext difference can result in various ciphertext differences depending on the key used. The probability that a given ciphertext difference occurs given a plaintext difference can be found by examining the properties of the S-boxes and computing a table (called difference distribution table) for each one, before starting the attack.

What the attacker wants to find is such a plaintext/ciphertext difference pair (called differential) with a high probability for each S-box. The encryption key does not affect the differences since XORing makes the key-bits used disappear, and so these tables can be pre-computed. The attacker then multiplies the probabilities of each differential to obtain the probability of the differential characteristic.

As seen above, the S-boxes of a cipher play a crucial role in differential cryptanalysis. A cryptanalyst needs to know the structure of the S-boxes in order to formulate the attack. One way to offer protection against differential cryptanalysis is through careful design of the S-boxes so that there are no differential characteristics with high probability. A method to achieve this is the Wide Trail Design Strategy, used in the design of Rijndael [4].

It is also believed that key-dependent S-boxes, that is S-boxes that change according to the encryption key, make a cipher immune to differential cryptanalysis. The cryptanalyst doesn't know where to start: he cannot even construct difference distribution tables since he doesn't know the structure of the S-boxes.

Another cipher, apart from ours, using key-dependent S-boxes is Twofish [28], one of the five AES finalists. In [23], an attempt to attack Twofish using differential cryptanalysis was made. The authors question the intuition that key-dependent S-boxes are more secure than carefully constructed S-boxes. They propose choosing the S-box to fit the attack, instead of choosing the attack to fit the S-box. They

have attacked a reduced-round (6-round) Twofish by choosing characteristics that have a high probability of appearing for a fraction of the keyspace.

The reason their approach was successful is that Twofish uses 16 of the key bits to construct the S-boxes from a given set of four fixed S-boxes. As a result, there are only 2^{16} possible S-boxes, that are feasible to analyse off line (that is before the attack is mount). Our cipher has 16 S-boxes with 2^{128} possibilities for each, depending on the key, making off line analysis infeasible.

4.4 Conclusions

In this chapter we have proposed a new block cipher whose internal operations are taken from group theory. In particular it is based on the Grigorchuk groups. We have seen that one can easily change the key and block size, as long as they remain the same and a power of two, and it has key-dependent S-boxes making certain types of cryptanalysis non applicable.

However, study of reactions against the aggregation of existing cryptanalytic attacks and efficiency of implementation was omitted and left for future work. Focus was given on theoretically investigating possible weaknesses springing from the underlying group theoretic properties.

In general, a lot more work is needed before this cipher can even be compared to the state-of-art ciphers in existence.

Chapter 5

On the Classification of Periodic Binary Sequences into Nonlinear Complexity Classes

In this chapter we begin classification of periodic binary sequences into nonlinear complexity classes and present an algorithm that checks for short cycles in large nonlinear feedback shift registers (NLFSRs, see Section 2.2 in Chapter 2 for definition) using this classification. The results of this chapter will appear in [25].

We begin in Section 5.1 with the definition of nonlinear complexity (a name preferred to maximum order complexity as it is more easily seen as the counterpart to linear complexity). In Section 5.2 we attempt to classify the sequences into nonlinear complexity classes, before presenting the algorithm checking for short cycles in Section 5.3. The majority of results in the latter section are due to Johannes Mykkeltveit, the co-author of [25]. We finish with our conclusions in Section 5.4.

5.1 Preliminaries

In this section we give the definition of nonlinear complexity and a brief discussion of how to proceed in the next section.

Definition 5.1.1 *A sequence s is periodic if there exists a positive integer r such that $s_{i+r} = s_i$, for $i = 0, 1, \ldots$, and aperiodic otherwise. The smallest such positive integer r is called the period of s and denoted by $p(s)$.*

Definition 5.1.2 *The nonlinear complexity $\mathrm{C}(s)$ of a periodic sequence s is the least integer k such that all k-vectors $(s_q, s_{q+1}, \ldots, s_{q+k-1})$, $q = 0, 1, \ldots, p(s) - 1$ are different. Indices are reduced modulo $p(s)$. $\mathrm{C}(s)$ is defined to be 1 if $p(s) = 1$.*

Example 5.1.3 *Let $s = (000101)^\infty$. Then $\mathrm{C}(s)= 4$, since all 4-vectors are different and the 3-vector (010), 2-vector (00) and 1-vector (0) are repeated.*

We will denote by $nlin(k, e)$ the number of binary sequences of nonlinear complexity k and period e.

Definition 5.1.4 *A binary necklace of length l is an equivalence class of binary strings of length l under rotation. It is periodic if the strings it contains are periodic, and aperiodic otherwise. In the periodic case we have that the period e of the strings divides l and $e/l > 1$.*

In order to classify the binary sequences of period e into nonlinear complexity classes we have to consider a representative from each binary aperiodic necklace of length e. This can be deduced from the fact that all members of a binary necklace have the same nonlinear complexity [15]. Therefore, throughout this chapter, s will denote the repeating part of the binary periodic sequence $(s_0 s_1 s_2 \ldots s_{e-1})^\infty$ of period $p(s) = e$. Also, all indices will be reduced modulo e.

Proposition 5.1.5 ([15]) *For any integer* e*, we have that $nlin(k, e) = 0$, where $1 \leq k < \lceil \log_2(e) \rceil$.*

Proof. Suppose there was a sequence of such complexity. That would mean that all $k - vectors$ would be different. There are e such k-vectors but only $2^k < e$ distinct binary k-vectors. ■

The total number of binary aperiodic necklaces of length e is well known [2, 30] to be equal to the number of irreducible binary polynomials of degree e. Hence

$$\sum_{k=\lceil \log_2(e) \rceil}^{e} nlin(k,e) = \frac{1}{e} \sum_{d|e} \mu\left(\frac{e}{d}\right) 2^d \ , \tag{5.1}$$

where μ denotes the Möbius function.

Table 5.1 tabulates the first values of $nlin(e-\gamma, e)$, as found by exhaustive search. The sum of each row is described by (5.1). Our aim is to find a general formula for each individual entry of the table.

5.2 Determining $nlin(e-\gamma, e)$

As discussed in the previous section, consider $s = s_0 s_1 s_2 \dots s_{e-1}$. We remind the reader that throughout the chapter, all subscripts are reduced modulo e. For s to have complexity $\mathrm{C}(s) = e - \gamma$, where $0 \leq \gamma \leq e - \lceil \log_2(e) \rceil$, it would mean, by definition, that all $(e-\gamma)$-vectors $(s_q, s_{q+1}, \dots, s_{q-1-\gamma})$, where $0 \leq q \leq e-1$, are different, and also that at least one pair of the $(e-1-\gamma)$-vectors

$$\begin{array}{rcll} s_0 s_1 & \dots & s_{e-2-\gamma} & (\mathcal{S}_0) \\ s_1 s_2 & \dots & s_{e-1-\gamma} & (\mathcal{S}_1) \\ \vdots & & & \vdots \\ s_i s_{i+1} & \dots & s_{i-2-\gamma} & (\mathcal{S}_i) \\ \vdots & & & \vdots \\ s_{e-1} s_0 & \dots & s_{e-3-\gamma} & (\mathcal{S}_{e-1}) \end{array}$$

is the same (otherwise, if none are equal we would have $\mathrm{C}(s) \leq e-1-\gamma$, and if a triplet or more are equal, we would have that at least two of the $(e-\gamma)$-vectors are equal and so $\mathrm{C}(s) \geq e+1-\gamma$).

Without loss of generality, we consider the $e-1$ cases of $(\mathcal{S}_0)$ being the same as

Table 5.1: The first values of $nlin(e-\gamma, e)$

2	1																		
3	2	0																	
4	2	1	0																
5	4	2	0	0															
6	2	4	3	0	0														
7	6	4	4	4	0	0													
8	4	6	8	10	2	0	0												
9	6	4	14	18	14	0	0	0											
10	4	10	12	16	40	17	0	0	0										
11	10	8	14	30	46	64	14	0	0	0									
12	4	6	16	38	54	104	100	13	0	0	0								
13	12	10	18	38	70	134	194	142	12	0	0	0							
14	6	16	20	36	74	132	303	366	188	20	0	0	0						
15	8	6	26	40	106	170	324	558	644	268	32	0	0	0					
16	8	14	20	52	92	176	348	732	1110	1122	390	16	0	0	0				
17	16	14	26	54	104	214	410	806	1448	2162	1918	538	0	0	0	0			
18	6	16	28	46	94	236	416	816	1803	2966	4105	3297	703	0	0	0	0		
19	18	16	30	62	120	246	482	958	1880	3560	5960	7914	5506	842	0	0	0	0	
20	8	14	20	68	140	248	494	952	1916	4176	7340	12070	14800	9046	1085	0	0	0	0
21	12	10	42	64	122	242	618	1096	2158	4312	8302	15132	23836	27942	14660	1310	0	0	0
22	10	28	36	68	140	274	550	1094	2188	4330	9583	17380	31069	47571	51611	23160	1465	0	0
23	22	20	38	78	152	310	614	1230	2450	4890	9700	19056	35894	63504	94132	94660	36428	1544	0
24	8	14	36	70	126	306	558	1354	2452	4894	9774	21236	39740	74422	128876	185014	172188	56232	1570
25	20	18	34	70	200	344	680	1364	2720	5404	10816	21560	42510	82526	153554	261740	361356	310652	84640
26	12	34	44	84	172	338	680	1356	2716	5408	10826	21596	47102	88711	171375	315939	528631	701641	555775
27	18	16	54	88	170	338	682	1364	3220	5938	11860	23686	47302	94080	184402	354656	647200	1064850	1358056
e/γ	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19

$(\mathcal{S}_i),\ 1 \leq i \leq e-1$:

$$\begin{array}{lllllll}
(\mathcal{S}_0) = (\mathcal{S}_1) & \Rightarrow & s_0 = s_1, & s_1 = s_2, & \ldots & , & s_{e-2-\gamma} = s_{e-1-\gamma} \\
(\mathcal{S}_0) = (\mathcal{S}_2) & \Rightarrow & s_0 = s_2, & s_1 = s_3, & \ldots & , & s_{e-2-\gamma} = s_{e-\gamma} \\
\vdots & & & & \vdots & & \\
(\mathcal{S}_0) = (\mathcal{S}_i) & \Rightarrow & s_0 = s_i, & s_1 = s_{i+1}, & \ldots & , & s_{e-2-\gamma} = s_{i-2-\gamma} \\
\vdots & & & & \vdots & & \\
(\mathcal{S}_0) = (\mathcal{S}_{e-1}) & \Rightarrow & s_0 = s_{e-1}, & s_1 = s_0, & \ldots & , & s_{e-2-\gamma} = s_{e-3-\gamma}
\end{array}$$

Now, suppose $(\mathcal{S}_0) = (\mathcal{S}_i)$ for some $1 \leq i \leq e-1$. To ensure that the $(e-\gamma)$-vectors $(s_0 s_1 \ldots s_{e-1-\gamma})$ and $(s_i s_{i+1} \ldots s_{i-1-\gamma})$ are different, we must also have that $s_{e-1-\gamma} \neq s_{i-1-\gamma}$.

Similarly, the $(e-\gamma)$-vectors $(s_{e-1}, s_0, s_1, \ldots, s_{e-2-\gamma})$ and $(s_{i-1}, s_i, s_{i+1}, \ldots, s_{i-2-\gamma})$ will be be different only if $s_{e-1} \neq s_{i-1}$.

When $\gamma = 0$, there is only one inequality, namely $s_{e-1} \neq s_{i-1}$. However, as $s_j = s_{j+i}$, where $0 \leq j \leq e-2$, and $i-1 < e-1$ we have

$$s_{i-1} = s_{i+i-1} = s_{2i-1} = \ldots = s_{ki-1} \ ,$$

for some $k \in \mathbb{N}$. Now, if we let $gcd(e,i) = d$ then, when $k = \frac{e}{d}$ we have

$$s_{i-1} = s_{\frac{e}{d}i-1} = s_{\frac{i}{d}e-1} = s_{e-1} \ ,$$

a contradiction. We have thus provided an alternative proof of the following proposition.

Proposition 5.2.1 ([15]) $nlin(e,e) = 0$.

In the sequel we will consider fixed i and γ such that $1 \leq i \leq e-1$ and $1 \leq \gamma \leq e - \lceil \log_2(e) \rceil$.

For $\gamma > 1$, no relations between s_{e-t} and $s_{i-t},\ \gamma \geq t \geq 2$, are imposed and thus we can have all possible combinations of equalities and inequalities. The only constraint is that we have to have an even number of inequalities, otherwise we would reach a contradiction, as in the case $\gamma = 0$. However, as will be seen in Proposition

5.2.21, this constraint is a necessary but not sufficient condition to avoid such a contradiction.

The total number of possibilities for the choice of relations is

$$\sum_{k=0}^{\lfloor \frac{\gamma-1}{2} \rfloor} \begin{pmatrix} \gamma-1 \\ 2k \end{pmatrix} = 2^{\gamma-2} \ .$$

Our approach will be to view the relations between s_t and $s_{i+t},\ 0 \leq t \leq e-1$ in each recursion as a binary vector v of length e, where a 0 would denote an equality and a 1 an inequality. It is obvious that these vectors are of even parity (that is they have an even number of 1's) and we have

$$s_{i+t} = s_t \oplus v_t \ .$$

We will call such a vector v the *relating* vector of the recursion $R_v(e,i)$. The set of all $2^{\gamma-2}$ possible relating vectors for a given e and γ will be denoted by $\mathcal{V}_{e,\gamma}$. If v is of the form

$$v = 0 \ldots 01 \ \underbrace{0 \ldots 0}_{\gamma-1} \ 1 \ ,$$

which means we only have equalities, then $R_v(e,i)$ is the same as $R(e,i,\gamma)$ of Definition 5.2.2. Note that with this notation the case $\gamma = 1$ is now also covered.

Definition 5.2.2 *By recursion $R(e,i,\gamma)$ we mean the following list of relations between the digits of a sequence s of length e:*

$$s_j = s_{i+j},\ s_{e-1-\gamma} \neq s_{i-1-\gamma},\ s_{e-1} \neq s_{i-1} \ ,$$

where $j \in \{0, \ldots, e-2\} \setminus \{e-1-\gamma\}$.

Definition 5.2.3 *Two sequences of length e are cyclically inequivalent if they do not belong to the same binary necklace.*

Definition 5.2.4 *$S(R_v(e,i))$ is the set of cyclically inequivalent sequences which satisfy recursion $R_v(e,i)$ and $|S(R_v(e,i))|$ denotes its cardinality.*

Definition 5.2.5 *$S(R_v(e,i),\delta)$ is the set of cyclically inequivalent sequences of nonlinear complexity δ which satisfy recursion $R_v(e,i)$ and $|S(R_v(e,i),\delta)|$ denotes its cardinality.*

Example 5.2.6 *Consider $v = 00000101$. We have that $R_v(8,2) = R(8,2,2)$, defined by the following list of relations between the digits of a lenght-8 sequence s:*

$$s_0 = s_2,\ s_1 = s_3,\ s_2 = s_4,\ s_3 = s_5,\ s_4 = s_6,\ s_5 \neq s_7,\ s_6 = s_0 \text{ and } s_7 \neq s_1\ .$$

Therefore, we have $s_0 = s_2 = s_4 = s_6$ and $s_1 = s_3 = s_5 \neq s_7$, giving 4 possible sequences:

$$S(R(8,2,2)) = \{00000001,\ 11111110,\ 01010100,\ 10101011\}\ .$$

Their corresponding nonlinear complexities are 7, 7, 6 and 6 and so $|S(R(8,2,2),7)| = |S(R(8,2,2),6)| = 2$.

Using the definition above, we can now relate $nlin(e-\gamma,e)$ and recursions $R_v(e,i)$ in the following obvious theorem.

Theorem 5.2.7 $nlin(e-\gamma,e) = \sum_{v\in\mathcal{V}_{e,\gamma}} \sum_{i=0}^{e-1} |S(R_v(e,i),e-\gamma)|$.

Proposition 5.2.8 *Let $v_1,\ldots,v_\gamma$ be the relating vectors of recursions $R(e,i,1)$, $\ldots, R(e,i,\gamma)$ respectively. Then, the relating vectors in $\mathcal{V}_{e,\gamma}$ are obtained from all possible sums of v_γ with an even number of vectors v_z, where $z \in \{1\ldots\gamma-1\}$.*

Proof. All vectors $v_1,\ldots,v_\gamma$ have a 1 in the $(e-1)^{th}$ position, and since there is an odd number of them in each sum, the resulting vector will also have a 1 in the $(e-1)^{th}$ position. Also, the $(e-\gamma-1)^{th}$ position will also be a 1 due to v_γ. The remaining positions will depend on which v_z, where $z \in \{1\ldots\gamma-1\}$, are in the sum: the $(e-z-1)^{th}$ positions will be a 1 and the rest will be 0. ∎

Example 5.2.9 *For $e = 9, \gamma = 4$, there are $2^{4-2} = 4$ possible relating vectors:*

1. (000010001) *obtained from recursion $R(9,i,4)$*

2. $(000011011) = (000010001) \oplus (000001001) \oplus (000000011)$ *obtained from recursions* $R(9,i,4)$, $R(9,i,3)$ *and* $R(9,i,1)$ *respectively.*

3. $(000011101) = (000010001) \oplus (000001001) \oplus (000000101)$ *obtained from recursions* $R(9,i,4)$, $R(9,i,3)$ *and* $R(9,i,2)$ *respectively.*

4. $(000010111) = (000010001) \oplus (000000101) \oplus (000000011)$ *obtained from recursions* $R(9,i,4)$, $R(9,i,2)$ *and* $R(9,i,1)$ *respectively.*

Theorem 5.2.10 *Consider the recursion* $R_v(e,i)$ *and suppose that* $v = v^0 \oplus v^1 \oplus \ldots \oplus v^l \in \mathcal{V}_{e,\gamma}$, *where* v^0 *and* v^k *are the relating vectors of the recursions* $R(e,i,\gamma_0 = \gamma)$ *and* $R(e,i,\gamma_k)$ *respectively,* $1 \leq k \leq l$, l *is even and* $\gamma_k < \gamma \ \forall k$. *Let* s^0, s^k *be sequences satisfying these recursions respectively. Then the sequence* $s = s^0 \oplus s^1 \oplus \ldots \oplus s^l \in S(R_v(e,i))$.

Proof. We have that $s^m = s_0^m \ldots s_{e-1}^m$, $v^m = v_0^m \ldots v_{e-1}^m$ and $s_{n+i}^m = s_n^m \oplus v_n^m$, where $0 \leq m \leq l$, $0 \leq n \leq e-1$. Therefore,

$$s_{n+i} = \bigoplus_{m=0}^{l} s_{n+i}^m = \bigoplus_{m=0}^{l} (s_n^m \oplus v_n^m) = \bigoplus_{m=0}^{l} s_n^m \bigoplus_{m=0}^{l} v_n^m = s_n \oplus v_n \ .$$

■

Proposition 5.2.8 together with Theorem 5.2.10 above suggests that we only need to investigate recursions $R(e,i,\gamma)$ and sums of sequences satisfying them.

5.2.1 The Recursions $R(e,i,\gamma)$

In this subsection we study the properties of the recursions $R(e,i,\gamma)$. In the way we defined them, we have not imposed any conditions to exclude the case of a third $(e-\gamma-1)$-vector being equal to the other two. In that case, since we are working in binary, at least two of the $(e-\gamma)$-vectors would be the same and therefore the nonlinear complexity of the sequence greater than $(e-\gamma)$. Hence, the following two results are immediate.

Proposition 5.2.11 $S(R(e,i,\gamma), e-\gamma) \subseteq S(R(e,i,\gamma))$.

Corollary 5.2.12 $|S(R(e,i,\gamma))| = \sum_{d=1}^{\gamma} |S(R(e,i,\gamma), e-d)|$.

Definition 5.2.13 *Given m such that $0 \leq m \leq e-1$, the recursion $R^m(e,i,\gamma)$ is the following list of relations between the digits of a sequence s of length e:*

$$s_j = s_{i+j},\ s_{m-1-\gamma} \neq s_{i-1-\gamma+m},\ s_{m-1} \neq s_{i-1+m}\ ,$$

where $j \in \{0,\ldots,e-1\} \setminus \{m-1-\gamma, m-1\}$.

The recursion $R^m(e,i,\gamma)$ corresponds to the case $(\mathcal{S}_m) = (\mathcal{S}_{m+i})$ (as compared to $(\mathcal{S}_0) = (\mathcal{S}_i)$ for recursion $R(e,i,\gamma)$). It has as relating vector the m-digits cyclic shift of the relating vector for $R(e,i,\gamma)$.

Proposition 5.2.14 *Let sequence $s = s_0 s_1 \ldots s_{e-2} s_{e-1} \in S(R(e,i,\gamma))$. Then its m-digits right cyclic shift $s' = s_{e-m} s_{e-m+1} \ldots s_{e-m-2} s_{e-m-1} \in S(R^m(e,i,\gamma))$.*

Proof. Obvious. ∎

Example 5.2.15 *Consider $R^3(8,2,2)$. It is defined by the following list of relations between the digits of a lenght-8 sequence s:*

$$s_0 \neq s_2,\ s_1 = s_3,\ s_2 \neq s_4,\ s_3 = s_5,\ s_4 = s_6,\ s_5 = s_7,\ s_6 = s_0 \text{ and } s_7 = s_1\ .$$

Therefore, we have $s_1 = s_3 = s_5 = s_7$ and $s_0 = s_4 = s_6 \neq s_2$, giving 4 possible sequences:

$$S(R^3(8,2,2)) = \{00100000,\ 11011111,\ 10001010,\ 01110101\}\ .$$

It can be seen that the members of $S(R^3(8,2,2))$ are 3-digits right cyclic shifts of the members of $S(R(8,2,2))$ found in Example 5.2.6.

It is a consequence of Proposition 5.2.14 that recursions $R(e,i,\gamma)$ and $R^m(e,i,\gamma)$, where $0 \leq m \leq e-1$, share the same properties.

Definition 5.2.16 *We will call two recursions equivalent (denoted by $\sim$) if the sequences they define belong to the same binary necklace.*

Example 5.2.17 *We have already seen that* $S(R(8,2,2)) \sim S(R^3(8,2,2))$ *in the previous examples.*

Lemma 5.2.18 $R(e,i,\gamma) \sim R(e,i,e-\gamma)$.

Proof. Let sequence $s = s_0 s_1 \dots s_{e-2} s_{e-1} \in S(R(e,i,\gamma))$. By Proposition 5.2.14, its γ-digits right cyclic shift $s' = s_{e-\gamma} s_{e-\gamma+1} \dots s_{e-\gamma-2} s_{e-\gamma-1} \in S(R^\gamma(e,i,\gamma))$, where the recursion $R^\gamma(e,i,\gamma)$ is the following list of relations between the digits of a sequence s of length e:

$$s_j = s_{i+j},\ s_{\gamma-1-\gamma} = s_{e-1} \neq s_{i-1-\gamma+\gamma} = s_{i-1},\ s_{\gamma-1} \neq s_{i-1+\gamma}\ ,$$

where $j \in \{0,\dots,e-1\} \setminus \{e-1, \gamma-1\}$. This is exactly $R(e,i,e-\gamma)$. ∎

Lemma 5.2.19 $R(e,i,\gamma) \sim R(e,e-i,\gamma)$.

Proof. Let sequence $s = s_0 s_1 \dots s_{e-2} s_{e-1} \in S(R(e,i,\gamma))$. By Proposition 5.2.14, its $(e-i)$-digits right cyclic shift $s' = s_i s_{i+1} \dots s_{i-2} s_{i-1} \in S(R^{e-i}(e,i,\gamma))$, where the recursion $R^{e-i}(e,i,\gamma)$ is the following list of relations between the digits of a sequence s of length e:

$$s_j = s_{i+j},\ s_{i-1-\gamma+e-i} = s_{e-1-\gamma} \neq s_{e-i-1-\gamma},\ s_{i-1+e-i} = s_{e-1} \neq s_{e-i-1}\ ,$$

where $j \in \{0,\dots,e-1\} \setminus \{e-1-\gamma, e-1\}$. This is exactly $R(e,e-i,\gamma)$. ∎

Combining the results of the lemmas above we come to the following corollary:

Corollary 5.2.20

$$R(e,i,\gamma) = R^i(e,e-i,\gamma) = R^{e-\gamma}(e,i,e-\gamma) = R^{i-\gamma}(e,e-i,e-\gamma)\ .$$

By Proposition 5.2.1 and Corollary 5.2.20 we deduce that in the sequel, for each e we have to look only at the distinct recursions $R(e,i,\gamma)$ for $1 \leq i,\ \gamma \leq \left\lfloor \frac{e}{2} \right\rfloor$. Proposition 5.2.21 below limits them further to those such that $\gcd(e,i) \mid \gamma$.

Given a positive integer n, we will denote by $Div(n)$ the set of divisors of n and by $Div^*(n)$ the set $Div(n) \setminus \{n\}$.

Proposition 5.2.21 *If $gcd(e,i) \notin Div(\gamma)$, then $|S(R(e,i,\gamma))| = 0$.*

Proof. $R(e,i,\gamma)$ is given by

$$s_j = s_{i+j},\ s_{e-1-\gamma} \neq s_{i-1-\gamma},\ s_{e-1} \neq s_{i-1}\ ,$$

where $j \in \{0,\ldots,e-2\} \setminus \{e-1-\gamma\}$. Let $gcd(e,i) = g$. Then, there exist $x, y \in \mathbb{N}$ such that $i = xg$ and $e = yg$.

First we observe that for $i \neq \gamma$, by definition of $R(e,i,\gamma)$ we have

$$s_{i-\gamma-1} = s_{i+i-\gamma-1} = s_{2i-\gamma-1} = \ldots = s_{k_1 i-\gamma-1}\ ,$$

for some $k_1 \in \mathbb{N}$. In the case $i = \gamma$ we just have $s_{e-1} \neq s_{\gamma-1}$.

Also,

$$s_{i-1} = s_{i+i-1} = s_{2i-1} = \ldots = s_{k_2 i-1}\ ,$$

for some $k_2 \in \mathbb{N}$. Now, when $k_1 = \frac{e}{g}$ we obtain

$$s_{i-\gamma-1} = s_{\frac{e}{g}i-\gamma-1} = s_{\frac{i}{g}e-\gamma-1} = s_{e-\gamma-1}$$

and when $k_2 = \frac{e}{g}$

$$s_{i-1} = s_{\frac{e}{g}i-1} = s_{\frac{i}{g}e-1} = s_{e-1}\ ,$$

both contradicting the definition of $R(e,i,\gamma)$.

So, what we need to avoid the contradiction is to find $k_1, k_2 < \frac{e}{g}$ such that

$$k_1 i - \gamma - 1 \equiv e-1 \quad \text{and} \quad k_2 i - 1 \equiv e - \gamma - 1\ .$$

In that case we would have

$$s_{i-\gamma-1} = \ldots = s_{e-1} \neq s_{i-1} = \ldots = s_{e-\gamma-1}\ ,$$

satisfying both inequalities.

Now, if $k_1 i - \gamma - 1 \equiv e-1$, then we would have

$$\begin{aligned} k_1 i \equiv \gamma \bmod e \ &\Rightarrow\ k_1 i = k_3 e + \gamma \\ &\Rightarrow\ k_1 x g = k_3 y g + \gamma \\ &\Rightarrow\ g(k_1 x - k_3 y) = \gamma\ , \end{aligned}$$

for some $k_3 \in \mathbb{N}$.

Similarly, if $k_2 i - 1 \equiv e - \gamma - 1$, then

$$\begin{aligned} k_2 i \equiv e - \gamma \bmod e \quad &\Rightarrow \quad k_2 i = k_4 e - \gamma \\ &\Rightarrow \quad k_2 x g = k_4 y g - \gamma \\ &\Rightarrow \quad g(k_4 y - k_2 x) = \gamma \ , \end{aligned}$$

for some $k_4 \in \mathbb{N}$.

Since all of $g, k_1 x - k_3 y$ and $k_4 y - k_2 x \in \mathbb{N}$, the only possible way these equalities can hold is when g is a divisor of γ. Otherwise, for $g \notin Div(\gamma)$, we have that $s_{e-\gamma-1} = s_{i-\gamma-1}$ and $s_{e-1} = s_{i-1}$, contradicting the definition of $R(e, i, \gamma)$ and as a result $|S(R(e, i, \gamma)| = 0$. ∎

Proposition 5.2.22 *Let* $\gcd(e, i) = g$. *If* $g \in Div(\gamma)$, *then*

$$|S(R(e, i, \gamma))| = \begin{cases} 2^g & \text{if} \quad e \neq 2\gamma \\ 2^{g-1} & \text{if} \quad e = 2\gamma \end{cases} .$$

Proof. I. Let $\gcd(e, i) = g$. $R(e, i, \gamma)$ is given by

$$s_j = s_{i+j}, \ s_{e-1-\gamma} \neq s_{i-1-\gamma}, \ s_{e-1} \neq s_{i-1} \ ,$$

where $j \in \{0, \ldots, e-2\} \setminus \{e - 1 - \gamma\}$. This gives

$$\begin{array}{ccccccc} s_0 & = \ldots = & s_{ki} & = \ldots = & s_{e-i} \ , \\ s_1 & = \ldots = & s_{ki+1} & = \ldots = & s_{e-i+1} \ , \\ \vdots & & \vdots & & \vdots \\ s_{g-2} & = \ldots = & s_{ki+g-2} & = \ldots = & s_{e-i+g-2} \ , \end{array} \tag{5.2}$$

where $k \in \mathbb{Z}^*$, and also

$$s_{i-1} \quad = \ldots = \quad s_{e-\gamma-1} \neq s_{i-\gamma-1} \quad = \ldots = \quad s_{e-1} \ ,$$

as seen in the proof of Proposition 5.2.21. The above form of representing the recursion will be called its *structure*. There are 2 possibilities for each line and so $|S(R(e, i, \gamma))| = 2^g$.

II. $e = 2\gamma$ means that $1 \leq i \leq \gamma$. We use the same reasoning as in **I**. However, due to Corollary 5.2.20, for each sequence we obtain we will also obtain its γ-digits cyclic shift. Therefore, $|S(R(e, i, \gamma))| = 2^{g-1}$. ∎

Proposition 5.2.22 tells us that when $e = 2\gamma$ we have two repeating $(e-1-\gamma)$-vectors: $(\mathcal{S}_0) = (\mathcal{S}_i)$ and $(\mathcal{S}_\gamma) = (\mathcal{S}_{\gamma+i})$. In addition, when $g = 1$, $S(R(e,i,\gamma))$ contains the self-complementary sequences, that is to say sequences whose complements belong to the same binary necklace. The complement of a sequence s is a sequence s' such that $s \oplus s' = 11\ldots1$.

Example 5.2.23 *Consider* $R(8,3,4)$. *We have* $e = 8 = 2\gamma$ *and* $\gcd(8,3) = 1$. *It is given by the following list of relations between the digits of a length-8 sequence* s*:*

$$s_0 = s_3,\ s_1 = s_4,\ s_2 = s_5,\ s_3 \neq s_6,\ s_4 = s_7,\ s_5 = s_0,\ s_6 = s_1 \text{ and } s_7 \neq s_2\ .$$

Therefore, we have $s_2 = s_5 = s_0 = s_3 \neq s_6 = s_1 = s_4 = s_7$ *, giving 2 possible sequences, one being the complement of the other:* 01001011 *and* 10110100. *However, the first is a 4-digits right cyclic shift of the second or, in other words, they are self-complementary. Hence*

$$S(R(8,3,4)) = \{01001011\}\ .$$

Conjecture 5.2.24 *For all* γ, i *and* i' *such that* $0 \leq \gamma, i, i' \leq \left\lfloor \frac{e}{2} \right\rfloor$, $i \neq i'$ *and for all* m *such that* $0 \leq m \leq e-1$, *we have that*

$$S(R(e,i,\gamma)) \cap S(R^m(e,i',\gamma)) = \emptyset\ .$$

Theorem 5.2.25 *Let* $\gcd(e,i) = g$. *If* $g \in Div(\gamma)$, *then*

$$|S(R(e,i,\gamma), e-\gamma)| = \begin{cases} \displaystyle\sum_{d|g} \mu(d) 2^{g/d} & \text{if } g = i = \gamma < \frac{e}{2} \\ \displaystyle\sum_{d|g} \left(\mu(d) 2^{g/d}\right) - 2^{g-1} & \text{if } g = i = \gamma = \frac{e}{2} \\ 0 & \text{if } g < i = \gamma \\ 2^g & \text{if } g, \gamma < i \text{ and } \gamma < \frac{e}{2} \\ 2^{g-1} & \text{if } i \notin Div(\gamma) \text{ and } \gamma = \frac{e}{2} \end{cases},$$

where μ *denotes the Möbius function.*

Remark 5.2.26 *The present version of Theorem 5.2.25 does not cover the following cases, left for future work:* $g = i < \gamma$, $g < i \in Div^*(\gamma)$ *and* $g < i < \gamma$ *with* $i \notin Div^*(\gamma)$.

Proof. I. Let $e > 2\gamma$ and $\gcd(e, \gamma) = \gamma$. By Proposition 5.2.22 $R(e, \gamma, \gamma)$ gives

$$\begin{array}{lllll} s_0 & = \ldots = & s_{k_1\gamma} & = \ldots = & s_{e-\gamma} \ , \\ s_1 & = \ldots = & s_{k_1\gamma+1} & = \ldots = & s_{e-\gamma+1} \ , \\ \vdots & & \vdots & & \vdots \\ s_{\gamma-2} & = \ldots = & s_{k_1\gamma+\gamma-2} & = \ldots = & s_{e-2} \ , \\ s_{\gamma-1} & = \ldots = & s_{e-\gamma-1} & \neq & s_{e-1} \ , \quad \text{where } k_1 \in \mathbb{Z}^* \ . \end{array} \tag{5.3}$$

Now, consider $d \in Div^*(\gamma)$. We have that $\gcd(e, d) = d$ and $R(e, d, d)$ is given by

$$\begin{array}{lllllll} s_0 & = \ldots = & s_{k_2 d} & = \ldots = & s_{e-\gamma} & = \ldots = & s_{e-d} \ , \\ s_1 & = \ldots = & s_{k_2 d+1} & = \ldots = & s_{e-\gamma+1} & = \ldots = & s_{e-d+1} \ , \\ \vdots & & \vdots & & \vdots & & \vdots \\ s_{d-2} & = \ldots = & s_{k_2 d+d-2} & = \ldots = & s_{\gamma-2} & = \ldots = & s_{e-2} \ , \\ s_{d-1} & = \ldots = & s_{e-\gamma-1} & = \ldots = & s_{e-d-1} & \neq & s_{e-1} \ , \end{array} \tag{5.4}$$

where $k_2 \in \mathbb{Z}^*$.

We can see that the set $\{k_2 d + q\}$ contains the set $\{k_1\gamma + q + pd\}$, where $0 \leq q \leq d-1$, $0 \leq p \leq \frac{\gamma}{d} - 1$ and $k_1, k_2 \in \mathbb{Z}^*$. Therefore, the sequences satisfying $R(e, d, d)$ also satisfy $R(e, \gamma, \gamma)$ (splitting each line of (5.4) in $\frac{\gamma}{d}$ parts, we obtain (5.3)). Since $|S(R(e, d, d), e - \epsilon)| = 0$ when $\epsilon > d$ and $Div^*(d) \subset Div^*(\gamma)$, we are only interested in $S(R(e, d, d), e - d)$ for each $d \in Div^*(\gamma)$.

Hence, by Corollary 5.2.12 and Proposition 5.2.22,

$$|S(R(e, \gamma, \gamma), e - \gamma)| = 2^\gamma \ - \sum_{d \in Div^*(\gamma)} |S(R(e, d, d), e - d)| \ . \tag{5.5}$$

Rearranging gives

$$2^\gamma = \sum_{d \in Div(\gamma)} |S(R(e, d, d), e - d)| \ . \tag{5.6}$$

Now, applying the Möbius Inversion Formula (see Theorem 2.3.3) to (5.6), we obtain

$$|S(R(e, \gamma, \gamma), e - \gamma)| = \sum_{d \in Div(\gamma)} \mu(d) 2^{\gamma/d} \ , \tag{5.7}$$

where μ denotes the Möbius function.

II. Let $e = 2\gamma$. We start by following the same arguments as the proof above and so, by Proposition 5.2.22, we have

$$|S(R(e,\gamma,\gamma), e-\gamma)| = 2^{\gamma-1} \quad - \sum_{d \in Div^*(\gamma)} |S(R(e,d,d), e-d)| \ . \tag{5.8}$$

Now, since $d < \gamma = \frac{e}{2}$, we know from (5.7) that

$$|S(R(e,d,d), e-d)| = \sum_{d' \in Div(d)} \mu(d') 2^{d/d'} \ .$$

By the Möbius Inversion Formula

$$\sum_{d \in Div(\gamma)} \sum_{d' \in Div(d)} \mu(d') 2^{d/d'} = 2^{\gamma} \ ,$$

and therefore (5.8) becomes

$$\begin{aligned} |S(R(e,\gamma,\gamma), e-\gamma)| &= 2^{\gamma-1} - 2^{\gamma} \ + \sum_{d \in Div(\gamma)} \left(\mu(d) 2^{\gamma/d}\right) \\ &= \sum_{d \in Div(\gamma)} \left(\mu(d) 2^{\gamma/d}\right) - 2^{\gamma-1} \ . \end{aligned}$$

III. Let $\gcd(e,\gamma) = g \in Div^*(\gamma)$. $R(e,\gamma,\gamma)$ gives

$$\begin{array}{lllll} s_0 & = \ldots = & s_{k_1\gamma} & = \ldots = & s_{e-\gamma} \ , \\ s_1 & = \ldots = & s_{k_1\gamma+1} & = \ldots = & s_{e-\gamma+1} \ , \\ \vdots & & \vdots & & \vdots \\ s_{g-2} & = \ldots = & s_{k_1\gamma+g-2} & = \ldots = & s_{e-\gamma+g-2} \ , \\ s_{\gamma-1} & = \ldots = & s_{k_1\gamma+\gamma-1} & = \ldots = & s_{e-\gamma-1} \neq \quad s_{e-1} \ , \text{where } k_1 \in \mathbb{Z}^* \ . \end{array} \tag{5.9}$$

Since $\gamma > g$, we have that $\gamma = gx$, for some $x > 1$. Therefore, $s_{k_1\gamma+q} = s_{k_1 gx+q} = s_{k_2 g+q}$, where $0 \leq q \leq g-2$ and $k_1, k_2 \in \mathbb{Z}^*$. Similarly $s_{k_1\gamma+\gamma-1} = s_{k_2 g+gx-1}$. Hence, each line of (5.9) is just a rearrangement of a line obtained by recursion $R(e,g,g)$ for which $|S(R(e,g,g), e-\epsilon)| = 0$, when $\epsilon > g$.

IV. Let $\gcd(e,i) = g$. We have $g < i$ and $i > \gamma$. $R(e,i,\gamma)$ gives (5.2). None of the combinations yield a different recursion and so by Proposition 5.2.22, $|S(R(e,i,\gamma), e-\gamma)| = 2^g$.

V. Let $\gcd(e,i) = g < i$. The case we have is $\gamma > i \notin Div(\gamma)$. $R(e,i,\gamma)$ gives (5.2) and by Proposition 5.2.22 we get $|S(R(e,i,\gamma), e-\gamma)| = 2^{g-1}$. ∎

The next theorem follows from Theorems 5.2.7 and 5.2.25.

Theorem 5.2.27 *$nlin(e-1,e) = \phi(e)$, where $\phi(n)$ is Euler's totient function.*

5.2.2 The Recursions $R_v(e,i)$.

We have already studied recursions $R_v(e,i) = R(e,i,\gamma)$, that is when the relating vector v has only two $1's$, in the previous subsection. By Proposition 5.2.8 and Theorem 5.2.10, studying the rest of the cases is equivalent to investigating the properties of sums of sequences satisfying recursions $R(e,i,\gamma)$. Doing this is further work.

5.3 An Algorithm Checking for Short Cycles in Large Nonlinear Feedback Shift Registers

In this section we present our proposed algorithm that checks for short cycles in large NLFSRs. A *cycle* is the periodic part of a sequence generated by a NLFSR. Every NLFSR is described by its nonlinear recursion.

Definition 5.3.1 *A recursion $s_n = f(s_0, s_1, \ldots, s_{n-1})$ is irreducible if all sequences it generates have complexity n, and reducible otherwise. We say it is of order n.*

Jansen [15] defined the *maximum order complexity* of s as the length of the shortest Feedback Shift Register that generates s. We relate to this in the following:

Definition 5.3.2 *The minimum recursion* $\mathrm{MinR}(s)$ *for s is the recursion of order* $\mathrm{C}(s)$ *and with the fewest binary operations that generates s.*

Finally, we let $\tau(e)$ denote the total number of cycles of period $\leq e$.

5.3.1 Algorithm Check_Sh

The algorithm takes the following two items of input data:

1. An array L of all sequences of period $\leq e$, e a given parameter:

$$\begin{aligned} L = \quad & ((0, s_0), (1, s_0), (0, \overline{s_0}), (01, s_0 \oplus s_1), \ldots, \\ & \left(s_0^q s_1^q \ldots s_{C(s^q)-1}^q, \mathrm{MinR}\left(s^q\right)\right), \ldots, \\ & \left(s_0^{\tau(e)} s_1^{\tau(e)} \ldots s_{C\left(s^{\tau(e)}\right)-1}^{\tau(e)}, \mathrm{MinR}\left(s^{\tau(e)}\right)\right)) \ , \end{aligned}$$

where $\overline{s_a}$ is the complement of s_a, that is $\overline{s_a} = s_a \oplus 1$.

2. The recursion $s_n = f(s_0, s_1, \cdots, s_{n-1})$ to be checked for short cycles.

Each element in L is a pair, the left member being a binary vector of length $\mathrm{C}(s)$ contained in the sequence s, and the right member being the minimum recursion for s. Obviously such a pair defines s uniquely. For example, the first member in the list is the all zero sequence, satisfying $s_{m+1} = s_m$ with $s_0 = 0$. The second is the all one sequence, the third is $0101\ldots$, and the fourth $011011\ldots$, which satisfies $s_{m+2} = s_m \oplus s_{m+1}$ with $s_0 = 0$ and $s_1 = 1$.

The main function of the algorithm is the following:

Algorithm 3 Check_Sh

Set $q = 1$. **While** $q \leq \tau(e)$ **do**

1: Read the q^{th} member of L, $L(q) = \left(s_0^q s_1^q \ldots s_{C(s^q)-1}^q, \mathrm{MinR}\left(s^q\right)\right)$.

2: Generate the whole period of the sequence $s_0^q s_1^q \ldots s_{e(s^q)-1}^q$.

3: **If** $s_{n+h}^q = f\left(s_h^q, s_{h+1}^q, \ldots, s_{n+h-1}^q\right)$, $h = 0, 1, \ldots, e\left(s^q\right) - 1$, **then quit Check_Sh** with the result that $f(s_0, s_1, \ldots, s_{n-1})$ is reducible. It generates the q^{th} member in L (we assume that n is greater than $\mathrm{C}(s^q)$ for all (s^q) in L) . **Otherwise** increase q by one.

End Check_Sh with the result that $f(s_0, s_1, \ldots, s_{n-1})$ does not generate cycles of period $\leq e$.

5.3.2 Implementing the Algorithm

By definition, the size of L is $\tau(e)$. However, when checking recursion $s_n = f(s_0, s_1, \cdots, s_{n-1})$ for short cycles, we only want to go through those sequences s in L that have complexity $\mathrm{C}(s) < n$. Therefore the "effective" size of L is

$$|L| = \sum_{e=2}^{e_{max}} \sum_{k=\lceil \log_2(e) \rceil}^{n-1} nlin(k, e) \ ,$$

where e_{max} is the largest period considered in L. Since our purpose is to check for short cycles, we will be looking at a reasonable, from a computational complexity aspect, value for e_{max} (≈ 30).

Definition 5.3.3 *The repeativity of a cycle s, denoted by $R(s)$, is defined as the number of $(\mathrm{C}(s)-1)$-vectors of s which are repeated.*

We will denote by $D(s)$ the number of different $(\mathrm{C}(s)-1)$-vectors occurring in s,

Example 5.3.4 *The cycle $s = 000101$ considered in Example 5.1.3 has nonlinear complexity $\mathrm{C}(s) = 4$, repeativity $R(s) = 2$ and $D(s) = 1$ since the 3-vector 010 appears twice.*

Theorem 5.3.5 *The probability that recursion $s_n = f(s) = F(s_1, s_2, \ldots, s_{n-1}) \oplus s_0$ generates a sequence of period $e(s)$, complexity $\mathrm{C}(s)$ and repeativity $R(s)$ is:*

$$\mathcal{P}(s) = \begin{cases} 2^{-e(s)} & if \quad C(s) < n \\ 2^{R(s)-e(s)} & if \quad C(s) = n \\ 0 & if \quad C(s) > n \end{cases} \ .$$

Proof. There are $2^{2^{n-1}}$ different F's. In the case $C(s) < n$, s determines $F(s_1, s_2, \ldots, s_{n-1})$ for e digits of $(s_1, s_2, \ldots, s_{n-1})$, leaving $2^{n-1} - e$ digits to be chosen arbitrarily. In other words, the fraction of all possible F's which generate s is $\frac{2^{2^{n-1}-e}}{2^{2^{n-1}}} = 2^{-e}$ as required.

In the case $C(s) = n$, $R(s)$ $(n-1)$-vectors occur twice in s. Therefore we have $e(s) = D(s) + R(s)$. This gives

$$\mathcal{P}(s) = \frac{2^{2^{n-1}-D(s)}}{2^{2^{n-1}}} = \frac{2^{2^{n-1}+R(s)-e(s)}}{2^{2^{n-1}}} = 2^{R(s)-e(s)} \ .$$

Finally, $f(s)$ cannot generate a sequence s with $C(s) > n$. ■

To optimise the algorithm, we should reorder the sequences in L in a way that $\mathcal{P}(s)$ does not increase. Furthermore, this version of the algorithm is not optimal if $f(s_0, s_1, \cdots, s_{n-1})$ has a symmetry. For example, if $s_n = f(s_0, s_1, \ldots, s_{n-1})$ is mapped onto itself if we replace s_q with $\overline{s_q}$ (self-complementary), if we replace s_q with s_{n-q} (reversible) or a combination of both, where $0 \leq q \leq n$. For example, $s_2 = \overline{s_1} \oplus s_0$ is reversible, and $s_2 = \overline{s_0}$ is both self-complementary and reversible. These symmetries define equivalence classes of sequences. In these cases a more optimal algorithm would be to include in L one representative for each equivalence class.

Nevertheless, the biggest difficulty in generating L lies in obtaining MinR(s) for each sequence s. The corresponding register synthesis with the fewest number of terms (not fewest binary operations as we want) proposed in [15] is of order $2e^2 \log_2(e)$, where e is the length of s.

It is a problem for further research to try to reformulate the algorithm in such a way that we do not need MinR. We may for instance let the algorithm which generates L be reentrant and let algorithm Check_Sh (Algorithm 3) call it each time it needs new bits of the short sequence it is working on.

5.4 Conclusions

In this chapter we have commenced the classification of periodic binary sequences into nonlinear complexity classes. Not all cases have been covered but our results and methodology suggest a way forward. Finally, we have presented an algorithm that performs checks for short cycles in large Nonlinear Feedback Shift Registers. Its implementation and efficiency depend on the above mentioned classification.

Chapter 6

Conclusions and Open Problems

The contributions of this thesis fall into three topics within cryptography, namely the cryptanalysis of public key cryptosystems , design of block ciphers and investigations of nonlinear feedback shift register sequences, mainly used in stream ciphers.

6.1 Overview of Chapter 3

In Chapter 3, through successful cryptanalysis, we demonstrated that the public key cryptosystem proposed in [8] is insecure. We believe that security cannot be improved by any modifications to the scheme since, as our results have shown, building a cryptosystem on the word problem of a Grigorchuk group given by a "secret" sequence is not safe.

6.2 Overview of Chapter 4

In Chapter 4, we made the innovation of designing a block cipher whose internal functions are based on the Grigorchuk groups. To the best of our knowledge, it is the first time such an attempt has been made. For this reason, emphasis was given in exploring the behaviour of the cipher with respect to its internal functions and the mathematics behind them.

Therefore, this topic offers rich grounds for further research, focused on the security and efficient implementation of the cipher. For example, it can be tested against the cryptanalytic attacks for block ciphers, listed in Subsection 2.2.3 of Chapter 2. Furthermore, the potential of a fast implementation using finite state automata (discussed in Subsection 2.1.1) can also be examined.

6.3 Overview of Chapter 5

In Chapter 5 we begun classification of periodic binary sequences into nonlinear complexity classes. An obvious direction for further research would be trying to obtain the missing results.

In particular, Conjecture 5.2.24 awaits a proof and so do the missing cases from Theorem 5.2.25 as explained in Remark 5.2.26. The latter problem, as shown by computer simulation in MAGMA, is very closely related to investigating the general recursions $R_v(e, i)$, also an open problem.

At the end of the chapter, we presented an algorithm that checks for short cycles in large nonlinear feedback shift registers. The design, however, is mainly theoretical and further work is needed before an efficient implementation can take place.

To begin with, the classification must be completed so that methods to generate MinR(s) can be found. Alternatively, we can try to reformulate the algorithm in such a way that we do not need MinR, as explained in the end of Subsection 5.3.2.

Finally, as also discussed in Subsection 5.3.2, L should be generated so that it contains only one representative from each equivalence class in the cases when the recursion under consideration has a symmetry.

Bibliography

[1] I. Anshel, M. Anshel and D. Goldfeld, An Algebraic Method for Public Key Cryptography. Mathematical Research Letters 6 (1999), 287–291.

[2] E.R. Berlekamp, Algebraic Coding Theory. McGraw-Hill Book Company (1968).

[3] E. Biham and A. Shamir, Differential Cryptanalysis of DES-like Cryptosystems. Journal of Cryptology Vol. 4 (1991) No. 1, 3–72.

[4] J. Daemen and V. Rijmen, The Design of Rijndael: AES - The Advanced Encryption Standard. Springer-Verlag (2002).

[5] D. Epstein, J. Cannon, D. Holt, S. Levy, M. Paterson and W. Thurston, Word Processing in Groups. Jones and Bartlett (1992).

[6] W. Diffie and M.E. Hellman, New Directions in Cryptography. IEEE Transactions on Information Theory Vol. IT-22 (1976), 644–654.

[7] D. Erdmann and S. Murphy, An Approximate Distribution for the Maximum Order Complexity. Designs, Codes and Cryptography Vol. 10 (1997), Springer-Verlag, 325–339.

[8] M. Garzon and Y. Zalcstein, The Complexity of Grigorchuk Groups with Application to Cryptography. Theoretical Computer Science Vol. 88 (1991), Elsevier, 83–98.

[9] R.I. Grigorchuk, Bernside's Problem on Periodic Groups. Functional Anal. Appl. Vol. 14 (1980), 41–43.

[10] R.I. Grigorchuk, Degrees of Growth of Finitely Generated Groups and the Theory of Invariant Means. Math. USSR Izvestiya Vol. 25 (1985), No. 2, 259–300.

[11] R.I. Grigorchuk, Personal Communication (2004).

[12] R.I. Grigorchuk, V.V. Nekrashevich and V.I. Sushchanskii, Automata, Dynamical Systems and Groups. Proceedings of the Steklov Institute of Mathematics Vol. 231 (2000).

[13] S.W. Golomb, Shift Register Sequences. Revised edition, Aegean Park Press (1982).

[14] P. de la Harpe, Topics in Geometric Group Theory. Chicago Lectures in Mathematics, The University of Chicago Press (2000).

[15] C.J.A. Jansen, Investigations on Nonlinear Streamcipher Systems: Construction and Evaluation Methods. PhD Thesis, Technical University of Delft (1989).

[16] K.H. Ko, S.J. Lee, J.H. Cheon, J.W. Han, J. Kang and C. Park, New Public Key Cryptosystem Using Braid Groups. Proceedings of Crypto '00, Lecture Notes in Computer Science Vol. 1880 (2000), Springer-Verlag, 166–183.

[17] L.V. Ly and W. Schindler, How to Embed Short Cycles into Large Nonlinear Feedback-Shift Registers. Revised Selected Papers of the 4th International Conference in Security in Communication Networks 2004, Lecture Notes in Computer Science Vol. 3352 (2004), Springer-Verlag, 367–380.

[18] W. Mao, Modern Cryptography Theory and Practice. Prentice Hall (2003).

[19] J.L. Massey, Shift Register Synthesis and BCH Decoding. IEEE Transactions in Information Theory Vol. IT-15 (1969), 122–127.

[20] M. Matsui, Linear Cryptanalysis Method for DES Cipher. Proceedings of Eurocrypt '93, Lecture Notes In Computer Science Vol. 765 (1993), Springer-Verlag, 386–397.

[21] A. Menezes, P. van Oorschot and S. Vanstone, Handbook of Applied Cryptography. CRC Press (1996).

[22] R.C. Merkle, Fast Software Encryption Functions. Proceedings of Crypto '90, Lecture Notes In Computer Science Vol. 537 (1991), Springer-Verlag, 476–501.

[23] S. Murphy and M.J.B. Robshaw, Differential Cryptanalysis, Key-Dependent S-Boxes and Twofish. Designs, Codes and Cryptography Vol. 27 (2002), No. 3, 229–255.

[24] G. Petrides, Cryptanalysis of the Public Key Cryptosystem Based on the Word Problem on the Grigorchuk Groups. Proceedings of the 9th IMA International Conference on Cryptography and Coding, Lecture Notes in Computer Science Vol. 2898 (2003), Springer-Verlag, 234–244.

[25] G. Petrides and J. Mykkeltveit, On the Classification of Periodic Binary Sequences into Nonlinear Complexity Classes. To appear in Proceedings of SETA'06 International Conference on Sequences and their Applications, Lecture Notes in Computer Science Vol. 4086 (2006), Springer-Verlag.

[26] B. Schneier, Applied Cryptography. Second Edition, Wiley (1996).

[27] B. Schneier, Description of a New Variable-Length Key, 64-Bit Block Cipher (Blowfish). Proceedings of Fast Software Encryption '93, Lecture Notes In Computer Science Vol. 809 (1994), Springer-Verlag, 191–204.

[28] B. Schneier, J. Kelsey, D. Whiting, D.Wagner, C. Hall and N. Ferguson, Twofish: A 128-Bit Block Cipher. http://www.schneier.com/paper-twofish-paper.pdf

[29] C. E. Shannon, Communication Theory of Secrecy Systems. Bell System Technical Journal, Vol. 28-4 (1949), 656–715.

[30] The On-Line Encyclopedia of Integer Sequences.
http://www.research.att.com/∼njas/sequences/?Anum=A001037.

[31] M.I Gonzalez Vasco, D. Hofheinz, C. Martinez, and R. Steinwandt, On the Security of Two Public Key Cryptosystems Using Non-Abelian Groups. Designs, Codes and Cryptography Vol. 32 (2004), 207–216.

[32] N. Wagner and M. Magyarik, A Public-Key Cryprosystem Based on the Word Problem. Proceedings of Crypto '84, Lecture Notes in Computer Science Vol. 196 (1985), Springer-Verlag, 19–36.