Algebraic Cryptanalysis of Stream Ciphers with Non-linear Update

By

Mehreen Afzal

A thesis submitted to the Faculty of Information Security Department, Military College of Signals, National University of Science and Technology, Pakistan in partial fulfillment of the requirements for the degree of PhD in Information Security

March, 2010

I, Mehreen Afzal, confirm that the work presented in this thesis is absolutely original. I also confirm that the sources of all the information, where it was necessary to derive, have been indicated and mentioned.

Dedication

To my loving parents and sisters

Abstract

Stream ciphers are quite well known for providing security in comunication. Due to their efficient implementation they have received attention of many cipher designers in previous years. Many new designs have been proposed and extensively analyzed in the form of NESSIE and eSTREAM projects. In general a new proposed design has to ensure, at least, that it is resistant to the existing attacks. Algebraic attack is now quite a familiar threat for stream ciphers. Therefore, to make out the design components, that can strengthen a cipher, against algebraic cryptanalysis must also be of interest to stream cipher designers. Algebraic cryptanalysis, in its general form, aims at recovering the internal secret state bits of the registers of the cipher by solving non-linear algebraic equations. That is why it is considered, not to be applicable on stream ciphers, where registers are updated non-linearly. Since, in this case, degree of algebraic equations, which relate internal states with key-stream bits, increase with each clock. However, different designs with nonlinear update may offer disparate levels of resistance.

In this thesis, we analyze some structures of stream ciphers with non-linear update and identify the level of resistance their design shows against the reocvery of secret internal states. Our objective is to analyze and compare the design of the key generating mechanism and not the cipher along with its initialization mechanism. Thus, we concentrate on the key generating part and compare the ciphers on the basis that how many of their internal state bits can be recovered by solving non-linear algebraic equations, using guess and determine approach. Caused by a rise in the degree

of equations with each clock, some of the internal state bits have to be guessed to recover the remaining. Our analysis reveals, that due to some thoughtful guessing, more internal state bits can be recovered which are not possible otherwise. However, some structures are resistent to give secret state bits by solving algebraic equations, even after guessing large number of bits. Aim of this thesis is to identify such structures.

Ciphers considered for this work are A5/1, A5/2, Trivium, Grain and Mickey. Significance of this work also lies in the fact that we have analyzed those ciphers which have been selected for the final portfolio of completed eSTREAM project. Based on our analysis, we also propose some modifications in the design of Grain-v1 to strengthen it against intial state recovery attack, without any increase in the secret state bits. Some modifications in the design of Trivium are also suggested therefore, the same structure can be used with larger key bit space.

Acknowledgements

Praise to Allah, the Almighty for blessing me with strength and patience to go through this difficult part of my career. Without His will and help I would never have managed to even start this journey.

Throughout the years of my PhD, numerous people have supported me in different ways. I would like to take this opportunity to gratefully acknowledge their essential contribution. First of all I wish to thank my supervisor Dr. Ashraf Masood for his continuous support and encouragement. I am indebted to him not only for all kind of guidance and advice in my work but also for his help in some difficult times during these years. I would, also like to thank Dr. Akbar, Dr. Noman Jafri and Dr. Shamim Baig, the members of PhD guidance committee, for their valuable time.

This research was financially supported by Higher Education Commission of Pakistan (HEC) and I am grateful to HEC for giving me this opportunity.

I am also grateful to all those researchers of Cryptography, whose works have inspired me and guided me throughout my thesis.

I would like to thank National University of Sciences and Technology to give me opportunity for the PhD program. I would also like to give credit, to all the helpful people, faculty and staff at NUST especially at Information security department, for their administrative and practical support. I also wish to thank all my friends and colleagues in department especially Firdous Kausar, Liaqat, Nazir, Ahmad Cheema, Imtiaz Ali Khokhar and Dr. Arif Wahla for their help and support.

Throughout these years there was a life besides my studies also. I would like to extend my gratitude to my friends Rubeena, Shahida and Rabia for giving me encouragement as well as enjoyable time. Finally many thanks to my ammi and my sisters for giving me unending support and blessings. There were times, when I really doubted whether my work would ever be finished. In all such moments my family and my friends were a constant source of comfort and encouragement for me. I would also like to say thank you to, my father, for so many things in my life, although he is no more around me, but my achievement would comfort and solace his soul.

Contents

List of Figures

List of Tables

Chapter 1

INTRODUCTION

1.1 Introduction

The history of cryptology is the story of the centuries-old battle between cryptographers and cryptanalysts. Cryptography strives to construct ways and means for defending communications while cryptanalysis aims at designing powerful methods to attack a secured system. A diverse range of disciplines and technologies have been developed during this combat. Past few decades have witnessed a revolution of digital and network communications. Now this area is more relevant than ever before. Today, secured data storage and transmission is not only the need of military and commercial sector, but it is a common man's problem also, because he uses mobile phones, internet, banking systems and other services.

Thus Cryptology is the science that aims at providing information security in the digital world and its constituent fields are cryptography and cryptanalysis. Cryptography, studies the design of algorithms and protocols for information security and cryptanalysis is concerned with the study of Mathematical techniques, that attempt to break cryptographic primitives. Although the ideal situation would be to develop algorithms, which are provably secure, but generally practical systems are not provably secure and on the other hand provably secure systems are not practical. Therefore, the security of a cryptographic algorithm or protocol is usually assessed by testing all possible known attacks on it. Thus, analysis of the strength of ciphers becomes a very important objective of the cryptanalytic techniques.

Besides, the conventional role of cryptography of providing confidentiality of communication and data it also aims at providing services like authentication, data integrity and non-repudiation. To provide these services, three categories of cryptographic primitives are used. These three categories are as follows: unkeyed primitives which involve hash functions, symmetric key primitives includes block ciphers and stream ciphers and asymmetric key or public key primitives comprise public key ciphers and signatures. This thesis deals with stream ciphers mainly, which make an important part of symmetric key primitives.

Following sections of this chapter are devoted to the general introduction to the topic. Consequently the general frame work and outline of this thesis will be presented.

1.2 Overview of Symmetric Ciphers

The conventional way of encrypting is to use symmetric-key cryptography, where the word "Symmetric" means that same key is used for encryption and decryption or more generally the decryption key can be easily $computationally$ derived from the encryption key. Two standard design approaches of symmetric key ciphers are block ciphers and stream ciphers.

Block ciphers, as the name shows is a type of symmetric key encryption which functions fixed-length blocks of bits with an unvarying transformation. When encrypting, a block cipher takes an n-bit block of plaintext and secret key as input, and gives a corresponding n-bit block of ciphertext as output. The first known block cipher was developed according to the design methodology given by Horst Feistel is Lucifer Feistel (1970, 1973) that was a contribution of IBM in the 1970s. Data Encryption Standard (DES) is actually a revised version of the Lucifer that was adopted as a US government FIPS standard. DES was publicly released in 1976 and has been widely used. Block ciphers are mainly based on the concept of confusion and diffusion as laid down by by Shannon

in his paper, ”Communication Theory of Secrecy Systems” published in 1949 Shannon (1949). Advanced Encryption Standard (AES) however, has replaced DES now as a United States Federal Standard. It was adopted by National Institute of Standards and Technology (NIST) in 2001 after a 5-year standardization process. AES has a block size of 128 bits and three possible key sizes, 128, 192 and 256 bits. This cipher is widely accepted, and it has strong resistance against various kinds of attacks.

Another form of symmetric encryption is known as stream ciphers. Here plaintext bits are combined with a pseudorandom bit stream (keystream), of the length of pliantext, typically by bit wise XOR operation Rueppel (1986). In a stream cipher the plaintext digits are encrypted, one at a time, also the transformation of successive digits varies during the encryption. Stream ciphers is an important class of cipher systems. Designing high-speed stream ciphers has been an important cryptographic topic for the last few years. It is widely believed that a stream cipher can be considerably faster than block ciphers in software, and also simpler to implement in hardware. Since each bit of the palintext is encrypted individually, therefore error propagation is limited. This makes them most suitable for providing security in telecommunication systems especially wireless communications.

The security of stream cipher depends on the randomness of the bits of this secret sequence. In information theoretic sense, a random sequence provides unconditional security if the secret sequence, once used for encryption, is never used again. The concept of unconditional or perfect secrecy was established by Claude Shannon some years after the introduction of one-time pad (OTP). The first one-time pad system was electrical and invented by Gilbert Vernam in 1917. One time pad is perfectly secure against ciphertext-only cryptanalysis because if the key is truly random, an attacker cannot compute the plaintext from the ciphertext without knowlege of the

key, even via a brute force search of the space of all keys, because all possible plaintexts are equally likely decryptions of the ciphertext. Although OTP provides perfect secrecy still it is not feasible to apply it to practical systems. Stream ciphers follow this basic concept of OTP. Stream ciphers have a different design structure from block ciphers but in some cases a block cipher can be made to act effectively as a stream cipher.

All the modern ciphers are designed to meet theoretical security of OTP but they also have to be practical. Thus their design goal is to efficiently generate pseudo-random bits, which are practically indistinguishable from the truly random bits. A stream cipher and a Psuedo-random number generator(PRNG) are not identical, but since mostly cryptanalysis of the steam ciphers is done in the assumed known plaintext scenario, therefore breaking the PRNG would mean breaking of the stream cipher.

Unlike block ciphers, stream ciphers do not have a standard model. A variety of constructions of stream ciphers can be found in literature in comparison with basic model of SPN and Feistal-ciphers in Block cipher design. Till 2001, the only generic standards for stream ciphers were the block ciphers in different modes including output feedback (OFB), ciphertex feedback (CFB) and Counter (CTR) mode etc. However some application-specific standards exists which include: A5/1, A5/2 and A5/3 for GSM, RC4 adopted by IEEE 802.11 as part of WEP, etc. In general many people are convinced that there is no need for separate or different model for stream cipher, and using any block cipher in some stream cipher mode can easily serve the purpose. Now that we have AES, whose security has been thoroughly checked and it is also efficient in both hardware (HW) and software (SW). But the argument against this approach is that many such stream cipher structures can be designed which are much faster in SW and smaller in HW than Rijndael and their security is also better than the block ciphers in counter mode. Since in multi-

media applications like mobile phones, music and video, where, extremely high throughput is needed and low-complexity HW is also required. These requirements can be fulfilled by dedicated stream ciphers only. Also a stream cipher can be designed to be highly efficient on one platform only, so that technology specific requirements may be met.

1.3 Overview of Cryptanalysis

The first cryptanalysis known in the history, exploited the variation in the frequencies of lettes. This eventually gave birth to the science of code breaking. The earliest known description of the technique is by an Arab philosopher Abu Yusuf Ya'qub ibn Is-haq ibn as-Sabbah ibn 'omran ibn Ismail al-Kindl. His work entitled ***A Manuscript on Deciphering Cryptographic Messages;*** was rediscovered in 1987 in the Sulaimaniyyah Ottoman Archive in Istanbul. This manuscript describes the oldest cryptanalysis method frequency analysis. Cryptanalysis was thus emerged as a result of scholarly efforts of Muslim scientists of ninth century who had gained a sufficiently higher and sophisticated level of knowledge in several disciplines including Mathematics, Statistics and linguistics.

During the course of history, techniques of cryptanalysis grew with the development of techniques of cryptography. For different cryptographic primitives, many attacks have been developed so far. Our focus in this thesis is on the Stream ciphers. Now we will mainly discuss cryptanalytic attacks on stream ciphers. Initially it was generally thought that cipher algorithm itself is a part of the secret. But this approach failed in many cases and today's cryptanalysis is based on the ***Kerchoff's*** principle which states that: ***The security of the encryption scheme must depend only on the secrecy of the key, and not on the secrecy of the algorithm.*** .

For modern cipher systems, following attack senarios are considered conventionally. However,

each analysis is performed under Kerchoff's principle.

- Ciphertext only attack. In this scenario adversary has only the ciphertext and tries to analyse it to receive some information from it. It could be a proof or a disproof for some hypothesis stated, or even the secret key recovering.

- Known plaintext attack. Adversary knows both the ciphertext and the corresponding plaintext. This is the most common scenario in modern cryptanalysis.

- Chosen plaintext attack. In this case adversary chooses the messages to be sent, and then gets the corresponding ciphertext.

- Chosen ciphertext attack. In this scenario adversary has temporary access to the decryption device, and chooses herself ciphertext to be decrypted receiving the corresponding plaintext.

The results of cryptanalysis could be one of the following:

- ***Total break*** when the secret key is recovered, and then the rest of the message is known as well. This is the most desirable result.

- ***Partial break*** which helps to reduce the search space for the secret key.

- ***Information deduction*** which allows the adversary to get some partial information about the plaintext.

- ***A distinguishing algorithm*** means that we can show that the corresponding cipher has a significant bias, when comparing with a perfect cipher. When applied to stream ciphers, the keystream is analysed. In this case we say a distinguisher can recognize whether the keystream comes from a cipher or a random source.

Since our focus in this thesis is on the cryptanalysis of stream ciphers, therefore details of the cryptanalytic techniques of symmetric ciphers in general, are not given here. For a comprehensive discussion on the techniques of cryptanalysis one can refer to Maximov (June, 2006). In consequent section we give an overview of the cryptanalysis of stream ciphers.

1.3.1 Cryptanalysis of Stream Ciphers

Stream ciphers are more often analyzed in a known-plaintext situation. For synchronous stream ciphers, this means that the designed attack on the key-stream generator needs a known segment of the generated sequence. Attacks on key-stream generators can be divided into key-recovery attacks, where the secret key or internal state is completely or partially recovered and distinguishing attacks, where the goal is to distinguish the key stream from a purely random sequence.

Unlike block ciphers, stream ciphers are known to have some known components like LFSR, NLFSR, Boolean functions, clock control mechanism etc. By virtue of these common design strategies there are also some generic attacks on stream ciphers. Generally strength of a stream cipher is evaluated on the basis of the results of the known cryptanalytic attacks. One of the most efficient classes of key-recovery attacks that can be applied to all synchronous ciphers are (fast) correlation attacks. Correlation attack that was introduced by Siegenthaler Siegenthaler (1984, 1985) and later on refined in Gunter (1988); Meier and Staffelbach (1989) can now be found with many improvements Wiener (1999); Johansson and Jonsson (1999); Bellare (2000); Preneel (2000); Davies (2000); Johansson and Jonsson (2000). Other key-recovery attacks that exploit the inherent linearity of LFSR's underlying a key-stream generator, are usually referred to as algebraic attacks. Algebraic attack was first applied to block ciphers and public key cryptosystems Courtois and Pieprzyk (2002); Courtois (2001). Its first successful application to stream ciphers was done on Toyocrypt Courtois (2002), and its application to LILI 128 Courtois and Meier (2003)

made them very popular. Because of the interest of the thesis, we will give a detailed account of algebraic cryptanalysis on stream ciphers in next chapter.

Attacks that exploit statistical irregularities in a key stream are called statistical attacks. Ideally an attacker should not be able to differentiate between a truly random sequence and the sequence generated by the keystream generator. An attacker can get some knowledge about the guessed plaintext if the sequence derived from guessed plaintext, and observed ciphertex has any statistical deviation from a truly random sequence.

1.4 Background for this Thesis

Stream ciphers are an important class of cipher systems. Designing high-speed stream ciphers has been widely studied during the last few years. Owing to their good cryptographic properties, as well as, ease of implementation, they have vast applications. However, their security is a question which is being addressed in many of the researches. The security evaluation includes the main general characteristics of a stream cipher and the attacks to which it should be resistant. As discussed earlier, there is no standard model for stream ciphers unlike the block ciphers standard DES and AES. However, the only standards covering stream ciphers (apart from the CTR, OFB and CFB block cipher modes of operation) were those produced for specific applications. Most notable amongst these are the stream ciphers A5/1, A5/2 and A5/3 designed for use with GSM; the RC4 stream cipher, standardised for use with IEEE 802.11 wireless networks as part of a system called Wired Equivalent Privacy (WEP), and E_0 for Bluetooth.

For last few years security of stream ciphers has received increasing attention. Two European projects: NESSIE snd ECRYPT, have great influence on the evolution of stream ciphers. NESSIE NESSIE (1999) stands for New European Schemes for Signatures, Integrity, and Encryption. It

was a project within the Information Society Technologies (IST) Programme of the European Commission which started in 2000 and ended in 2004. The main objective of NESSIE has been to put forward a portfolio of strong cryptographic primitives obtained after an open call and evaluated using a transparent and open process. None of the stream ciphers submitted in the NESSIE project were selected for the final portfolio, attributed to the weaknesses found in them. NESSIE project was followed by eSTREAM project by ECRYPTeSTREAM (2005). ECRYPT is a Network of Excellence within the Information Societies Technology (IST) Programme of the European Commission; it was launched in February 2004. In November 2004, ECRYPT, gave a call for Stream Cipher Primitive. This provides an opportunity to the cryptographic community to settle on a new encryption standard. 34 stream ciphers were submitted against this call till April 2005. The eSTREAM call encompassed two different profiles, for which ECRYPT believed stream ciphers may offer significant advantages compared to block ciphers. eSTREAM project has been concluded and a report published online on April 2008 gave the final portfolio of the eSTREAM candidates Steve Babbage and Robshaw (2008). Final portfolio has selected eight stream ciphers for both profile out of 34 submissions of diverse structures. Organizers of eSTREAM, expect that with the time as the analysis matures for different structures, ciphers for specific deployment may be selected accordingly. Focus of this thesis involves analysis of hardware oriented stream ciphers with non-linear update, especially those included in eSTRAEM project.

Most of the cryptanalysis on stream ciphers is performed under a known plaintext assumption. Algebraic attacks on stream ciphers with linear feedback are quite practical and have received a lot of attention. Algebraic Cryptanalysis has been proved to be quite a successful technique for the security evaluation of stream ciphers as well as a threat to the structures which are resistant to other types of attacks. Some pioneering works include Courtois (2002); Courtois and Meier (2003); Courtois (2005a); Armknecht and Karuse (2003); Courtois (2003); Armknecht (2004a).

Originally algebraic attacks are proved to be successful against ciphers having combining or filtering Boolean function along with the linear part. Thus it is generally assumed that the structures with non-linear feed back are not vulnerable to algebraic cryptanalysis. One of the aims of this research is to uncover the level of difficulty of mounting a practical algebraic attack on some recent stream ciphers, which do not have the conventional structure vulnerable to algebraic cryptanalysis.

Every new proposed design ensures at least that it is resistant to the existing attacks. As mentioned earlier, algebraic attack is now quite a familiar threat for stream ciphers. Moreover, to make out the design components that can strengthen a cipher against algebraic cryptanalysis, must also be of interest to stream cipher designers. Algebraic cryptanalysis, in its general form, aims to recover the internal secret state bits of the cipher by solving non-linear algebraic equations. That is why it is considered to be non-applicable on stream ciphers, where registers are updated non-linearly. Since in this case, degree of algebraic equations, which relate internal states with key-stream bits, increase with each clock. However, different designs with nonlinear update may offer disparate levels of resistance. This research compares some eSTREAM proposed structures of stream ciphers and identify the level of resistance their design shows against recovering the secret internal states. eSTREAM proposed stream ciphers: Grain, Trivium and Mickey M. Hell and Meier (2005a,b); Cannière (2006); Cannière and Preneel (2005b); Babbage and Dodd (n.d.b,n) are analyzed and their key generating structures are compared. Moreover, some modifications in the design of Trivium and Grain-v1 are also suggested to enhance their security level without an increase in the number of secret state bits.

1.5 Organization of the Thesis

This thesis consists of three main parts. The first part including Chapter 1 and Chapter 2 is a general introduction to the main topic of the thesis. Chapter 1 gives a brief overview of the stream ciphers, their importance and an introduction to the general approach of cryptanalysis of stream ciphers. Whereas Chapter 2 gives brief details of the algebraic cryptanalysis in general. Chapter 3 through 6 gives experimental results on algebraic analysis of stream ciphers with non-linear update. In the final part, Chapter 7 and 8 some modifications in the stream ciphers Grain and Trivium are suggested based on the analysis made in previous chapters. Chapter 9 will summarize and conclude the thesis. Next, a detail of all chapters is given. The work given in this thesis is based on following works Afzal and Masood (2007, 2008d); Afzal et al. (2006); Afzal and Masood (2008b); Afzal et al. (2008); Afzal and Masood (2008c,a,e, 2009, 2010) that have been already published. For experimental results PC environment is used with CPU at 1.73 GHz and 1 GB RAM. Whereas Maple 10 Maple (n.d.) has been used for generation of equations of the ciphers and non-linear equations are solved using F_4 algorithm of finding Groebner basis Faugère (1999) implemented in Magma version 2.13-5 System (n.d.)

In Chapter 2, we give details of the algebraic cryptanalysis technique for stream ciphers. Algebraic attack is a vital development in cryptanalytic techniques. This attack falls under the known and chosen plain-text attack and algebraic in nature rather than statistical. The efficiency of the attack depends on the efficiency of the algorithm to generate the algebraic equations and to solve the generated large set of multivariate equations. The generation of algebraic equations is algorithm specific, while solving the system of multivariate algebraic equations is a much studied problem in the field of computational algebraic geometry and commutative algebra. In Chapter 2, we will also discuss the techniques used efficiently for solving algebraic equations of Cryptographic

algorithms.

In Chapter 3, algebraic cryptanalysis of A5/1 and A5/2 is discussed. Both of these versions of GSM security system have clock controlled structures. A5/1 uses mutually controlled registers therefore an efficient algebraic attack to recover all the internal state bits is not possible. We have tried to figure out the number of state bits that can be practically found using Groebner basis technique for solving equations. Most successful attack on A5/2, the weaker version of GSM encryption algorithm, is algebraic in nature. We also give or experimental results using an implementation of Groebner basis algorithm. Our results show that state bits of the cipher can be recovered in fewer number of data frames than required in a previous efficient attack against GSM communication.

Chapter 4 gives Algebraic analysis of Grain family of ciphers. Two versions Grain-v1 and Grain-128 have been analyzed and experimental results on solving their algebraic equations are given. Both of these versions have a NLFSR along with a LFSR and that is why registers of the cipher are non-linearly updated. We have also analyzed their structures to obtain low degree equations. Results on finding internal state bits are given using guess and determine approach.

In Chapter 5, we have considered Trivium, an eSTREAM candidate which has been included in their final portfolio also. Trivium has an internal state of 288 bits and three of its states are updated non-linearly at each clock. Therefore, in this case also, degrees of equations relating internal state bits with output bits do not remain constant. We have shown in this chapter, that with some bits guessed we can practically recover rest of the internal state bits. We have also analyzed a tweaked version Trivium/128, that has been proposed in literarture to improve upon the security of Trivium to 128 bits. Our results show that tweaked version is not suitable to offer 128 bit security.

In Chapter 6, the structure of Mickey is considered for Algebraic analysis. Registers of Mickey

are also updated non-linearly. But in this case due to the interdependent clocking of registers guess and determine approach fails to work. Therefore, our experiments could not recover internal state bits of the registers with some guessed bits for Mickey unlike Grain and Trivium.

Chapter 7 and Chapter 8 are meant to propose some changes in the structure of Grain-v1 and Trivium respectively. We have tried to make them better resistant against recovery of internal state bits by solving algebraic system of equations with guess and determine approach. We aim to improve upon the algebraic structure of the structures so that recovery of internal state bits can be avoided. Our proposed modification does not at all affect the main design philosophy of ciphers in both cases. Rather we attempt to make very small changes which can make the recovery of internal state bits difficult with the approach used in previous chapters.

Chapter 9 will conclude this thesis.

1.6 Conclusion

In this chapter, we have given an introduction of the thesis. Background and importance of this work is explained and layout of the following chapters has also been presented.

Chapter 2

Algebraic Cryptanalysis of Stream Ciphers

2.1 Introduction

Most of the cryptanalysis on stream ciphers is performed under a known plaintext assumption. For the additive stream cipher this means that the attacker has direct access to the key-stream output from the generator. Algebraic attack is a development in the cryptanalysis of symmetric encryption techniques that has received much attention in past few years. Algebraic attacks are more applicable on stream ciphers than any other primitive. Due to successful applicability of algebraic attack on stream ciphers, especially on those which have linear feed back, this technique is considered an important threat for stream ciphers. In this chapter, a study on algebraic attacks mounted on stream ciphers in general is presented. We will also discuss briefly the possible application of this class of attacks on stream ciphers with somewhat different structures, like clock-controlled ciphers, and ciphers with non-linear update.

2.2 Algebraic Cryptanalysis of LFSR Based Stream Ciphers

Stream ciphers are generally very suitable for the applications which require fast encryption rate or extremely low implementation cost in hardware. A linear transition functions along with a non-linear filtering function to break the inherent linearity, therefore, are very common components of stream ciphers. Amongst all linear components, LFSRs are very popular because of their low cost implementation. To destroy the linear and periodic sequence produced by one or more LFSRs, a

non-linear combiner or filtering function is used. General structure is shown in Figure 2.1

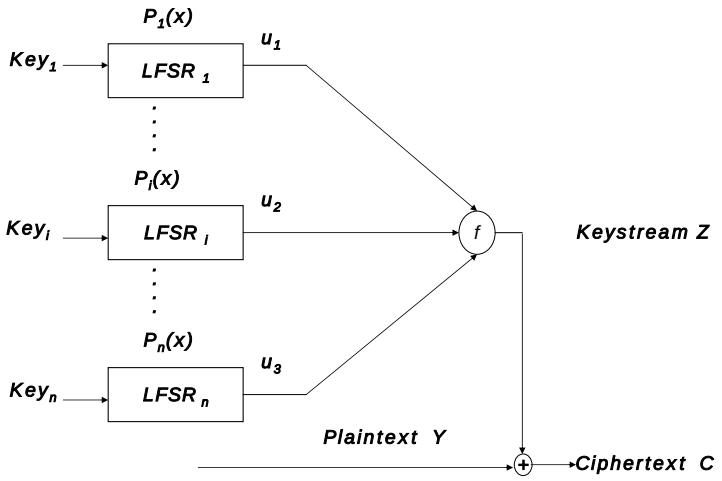

Figure 2.1: An additive stream cipher, where $(cipher - text)c_i = (plain - text)y_i \oplus (keystream)z_i$ and f is the non linear function

LFSR based stream ciphers are suitable for many cryptanalytic attacks as well. Algebraic cryptanalysis is also now one of the well known threats to such ciphers. The basic principle of algebraic attacks is based on the Shannons concept. That is to express the whole cipher as a large system of multivariate algebraic equations and then to solve it to recover the secret key. The complexity of this kind of attack will depend largely on the degree of the underlying algebraic system. Thus, Algebraic attacks can be viewed as a two step process:

First step: Find a system of algebraic equations that relates the bits of the initial state K and bits of the key-stream Z, this step differs for each cipher and depends mainly on its internal structure. However, it is a pre-computation step and once equations are formed, these can be used for attacking multiple key-streams.

Second step: Second step is performed after some key stream bits are obtained. The observed key-stream bits are substituted into the algebraic equations already formed and then these equations are solved using different methods like Linearization, XL, and Groebner bases. This solution is efficient if degree of equations is low, and a sufficient number of equations can be formed from the observed key-stream. Hence, to find low degree equations is a desirable goal in algebraic attacks.

2.2.1 Generation of Algebraic Equations

This section is devoted to give an outline on generating low degree equations for stream ciphers. First ever research on algebraic attacks on stream cipher is found on simple combiners Courtois and Meier (2003); Courtois (2002). In Courtois (2002), correlation properties of non-linear function are exploited. Further, an over defined system of multivariate equations of low degree is solved using XL algorithm. The complexity of the proposed attack was much lower than any other known attack. Whereas in Courtois and Meier (2003) no such low degree approximation is required, rather a new method for generating low degree equations is proposed. A brief detail of the approach will be given in the next section. Later, fast algebraic attacks were given by Courtois Courtois (2003), in which complexity of the attack was reduced by reducing the degree of equations using pre-computation algorithm. His results were further enhanced by Armknecht in Armknecht (2004a). Another research in Armknecht and Karuse (2003), showed an improved method for fast algebraic attacks using Fast Fourier Transform (FFT) and made the argument of algebraic attack on LFSR based stream ciphers being the most effective one.

2.2.1.1 Generating Equations for Combiners Without Memory

Simple combiners without memory can be represented as in Figure 2.2

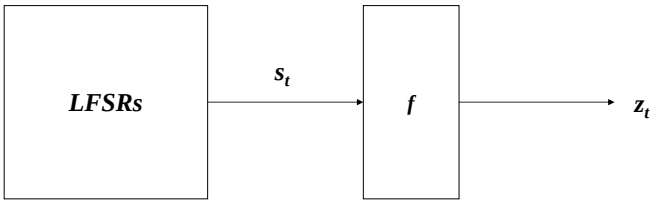

Figure 2.2: Simple Combiner without Memory

As described above, first step for an algebraic attack is to set up a system of equations in the unknowns s_t and output bits z_t. Suppose that LFSRs have k-bit state, and initial state is $S^0 = (s_0, s_1, ..., s_{k-1})$. Output bits and initial state bits can be algebraically related as follows:

$f(s_0, ...s_{k-1}) = z_0$

$f(L(s_0, ...s_{k-1})) = z_1$

$f(L^2(s_0, ...s_{k-1})) = z_2$

.....

The following equation always holds as a relation between initial states of LFSRs and output bits z_t: $f(s_t) \oplus z_t = 0, \forall t$, where $s_{t+1} = L^t(S_0) = L^t(s_0, ..., s_{k-1})$

$$\Rightarrow f(L^t(s_0, ...s_{k-1})) \oplus z_t = 0, \forall t \tag{2.1}$$

Above equation can be used to set up a system of equations while f and L are known. Here, degree of equations depends on the degree of f .

2.2.1.2 Combiners With Memory

A combiner with memory can generally be represented as in Figure 2.3:

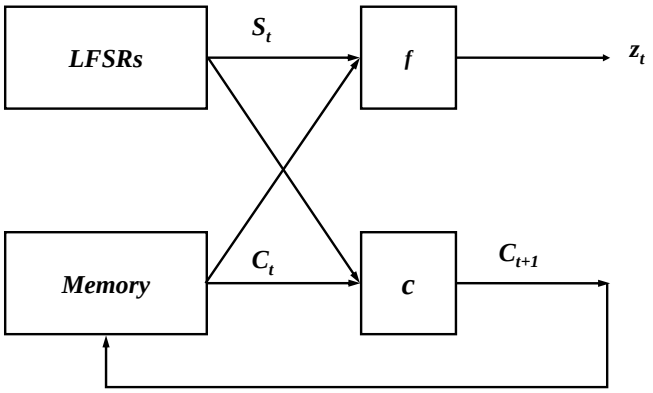

Figure 2.3: A Combiner with Memory

Stream ciphers, comprising of combiners with memory were primarily considered resistant to algebraic attacks but subsequent researches Armknecht and Karuse (2003); Courtois (2005a) proved it wrong. In this case, due to the role of memory bits, initial state bits and output bits may be related as follows:

$$z_t = f(s_t, c_t) where, s_{t+1} = L(s_t) = L^t(S_0) c_{t+1} = C(s_t, c_t) \tag{2.2}$$

. Acoording to equation 2.1, the following equation is followed:

$$f(L^t(s_0, ...s_{k-1}, c_t)) \oplus z_t = 0, \forall t. \tag{2.3}$$

It has been proved theoretically Armknecht and Karuse (2003) that for any combiner with k inputs and l bits of memory, an algebraic attack always exists.

2.2.1.3 Criteria For The Existence Of Low-Degree Equations

Following criteria have been given by Courtois for the algebraic attack on stream ciphers with simple combiners Courtois and Meier (2003):

- Boolean function f has a low algebraic degree
- f can be approximated by such a function with a probability close to 1, this probability is usually denoted as $1 - \epsilon$ for small ϵ
- The multivariate polynomial f has some multiple g such that $f.g$ is of low degree.
- $f.g$ can be approximated by a function of low degree with some probability $1 - \epsilon$

These criteria were extended further in Armknecht and Karuse (2003) and it has been proved that, for any combiner with k inputs and l bits of memory, required multivariate relations always exist with degree at most $\lceil k(l+1)/2 \rceil$. Moreovor a general criterion for the existence of low-degree equations for all the three cases: simple combiners, combiners with memory and for S-boxes were presented in Armknecht (2004b). Following definitions are required to state the general criterion.

- ***Annihilator:*** *Let $f, g \in B_n$ be Boolean function then g is annihilator of f if $f.g \equiv 0$ and set of annihilators is defined as: $An(f) = \{g \in B_n | f.g \equiv 0\}$*
- ***Characteristic function for simple combiners:****Let $f, g \in B_n$ be the output function of a simple combiner. For $z \in \{0,1\}$, Characteristic function $C_z \in B_n$ is defined as: $C_z(X) = 1 \Leftrightarrow f(X) = z, \forall X \in \{0,1\}^n$. For simple combiners it is easy to see that: $C_0 = f and C_1 = f \oplus 1$*

- ***Z-function*** *For a combiner with memory, for* $Z \in \{0,1\}^r, r \in I, r \geq 1$*, a function* $F_z : \{0,1\}^{k.r} \rightarrow \{0,1\}$ *is a z-function if the following holds for all clocks* t*:*

  $(z_t, ..., z_{t+r-1}) = Z \Rightarrow F_z(X) = 0, \forall X = (X_1, ...X_r) \in \{0,1\}^{k.r}$

- ***Characteristic function for combiners with memory*** *For a given* (k,l) *combiner, for* $Z \in \{0,1\}^r$*, then the charecteristic function* $C_Z : \{0,1\}^{k.r} \rightarrow \{0,1\}$ *is defined as:*

  $C_z(X_1, ...X_r) = 1 \Leftrightarrow \exists M \in \{0,1\}^l : f^r(M, X_1, ...X_r) = Z$

Thus the criterion for existence of low degree equations is given as:

Let $C \in B_n$ *be a charecteristic function. Then an equation of degree* $\leq d$ *exists iff* C *has an annihilator of degree* $\leq d$*.*

Thus, the existence of low degree equations and of low-degree annihilators can be considered equivalent.

2.2.2 Solving Algebraic Equations

The problem of solving systems of multivariate polynomial equations is NP-complete even for the quadratic equations in the binary field (GF(2)). As an example, the case of AES-128 can be considered, for which key can be recovered with a high probability from one AES-128 plaintext-ciphertext pair if a system of 8000 quadratic equations over GF(2) with 1600 variables can be solved Courtois and Pieprzyk (2002). The classical algorithm for solving such a system is Buchbergers algorithm for constructing Groebner bases and its many variants. In this section, a brief description of the techniques for solving the system of multivariate algebraic equations, like linearization, XL, XSL, Buchberger algorithm, F4 algorithm is given.

2.2.2.1 Linearization

Linearization is the simplest technique for solving the system of multivariate polynomial equations. It solves an over defined system of non-linear equations in polynomial time if enough linearly independent equations are given. Principle of linearization is to first replace each monomial with a new variable and then solve a linear system of a large number of indeterminates.

- **Complexity of Linearization:** Linearization method is easy and simple to implement, but its major drawback is its requirement of the knowledge of large amount of key-stream bits. There are about $T \approx \sum_{i=1}^{d} \binom{n}{i}$ monomials of degree$\leq d$ in the n variables. Considering each of these monomials as a new variable, if there are $T + M$ key-stream bits, a system of $R \geq T$ linear equations is obtained with T variables that can be solved by Gaussian Elimination on a linear system of size T. In theory, time taken by Guassian elimination is T^{ω}, where $\omega \leq 2.376$. Based on the Strassen's algorithm that requires about $7.T^{log_2 7}$ operations and the assumption that a modern CPU can handle 64 such operations in one single CPU clock, Courtois gave an estimation of the number of CPU clocks necessary in the attack as: $7/64.T^{log_2 7}$. However, this complexity estimate has been revised as: $E(T, R) = 0.76TR^{1.8} + 0.023R^{2.8}$) based on the fact that Strassen's algorithm has a large probability of failure in GF(2).

2.2.2.2 Relinearization

While linearization does not always lead to a unique solution, another technique Relinearization can solve many systems of equations which could not be solved by linearization Kipnis and Shamir (1999). Its complexity and success rate are however not well understood. The general idea of this method is to use linearization technique to solve m equations in $n(n+1)/2$ variables

(That is number of 2 or less degree monomials on n variables).

2.2.2.3 Extended Linearization

Extended Linearization(XL algorithm) was first proposed to solve quardratic equations Kipnis and Shamir (1999). Although XL is simple, the number of equations and variables for which it terminates successfully is not clear and also its asymptotic complexity is not known. XL can be further improved by techniques like XSL, FXL, XL, and XL_2 Kipnis and Shamir (1999). Still, a number of issues are to be settled regarding the success and implementation of XL and its variants.

2.2.2.4 Groebner Bases Algorithms

A brief description is given here that how a system of multivariate equations can be solved by using Groebner bases for the ideal generated by the multivariate polynomials. For this, consider the following problem: ***Problem:*** Let the base field is $GF(q) = F_q$ and $e_1, e_2, ..., e_m$ are m multivariate polynomials with n variables over the field F_q with

$X_i^q = X_i$ for $1 \leq n$, i.e., $e_j \in F_q[X_1, X_2, ...X_n] \langle X_1 q = X_1, ..., X_n = X_n^q \rangle$ for $1 \leq j \leq m$

The problem is, thus, to find the solution(s): $(x_1, x_2, ..., x_n) \in F_q^n$ such that $e_i(x_1, x_2, ..., x_n) = 0$ for $1 \leq i \leq m$

In abstract algebra, the above problem is studied in the following way:

Let $I \subseteq F_q[X_1, ...X_n]$ be the ideal generated by the polynomials $e_1, ...e_m$. The set of solutions $V(e_1, ...e_m) = (x_1, ..., x_n) \in F_q^n | e_i(x_1, ...x_n) = 0, \forall 1 \leq i \leq m$, is known as affine variety defined by $e_1, ...e_m$. So, the problem is the same as asking for the points in the affine variety: $V(e_1, ...e_m)$. This problem can be solved using Groebner basis of the ideal I. To describe the meaning of Groebner basis of an ideal, following must be considered:

Let $\prec$ be a total monomial order on the set of monomials X^α, where $\alpha \in F_q^n$. As the terms of each

polynomial can be uniquely ordered with respect to $\prec$, the notion of leading coefficient (LC(f)), leading monomial (LM(f)) and leading terms (LT(f)) are well defined.

Let $I \subseteq F_q[X_1, ..., X_n]$ be an ideal and $LT(I) = LT(f)|f \in I$. A finite subset $G = g_1, ..., g_s$ of I is said to be a Groebner basis if $\langle LT(g_1), ..., LT(g_s)\rangle = \langle LT(I)\rangle$.

It can be shown that every non-trivial ideal has a Groebner basis fixing a monomial order $\prec$. It may not be unique, but it is possible to get uniquely reduced Groebner basis. As the structure of the polynomials in Groebner is very simple, it is easy to exploit G for finding $V(G)$. Since $V(G) = V(I) = V(e_1, ..., e_m)$, finding $V(G)$ is enough. So, the remaining task is to generate the Groebner basis.

The concept of Groebner Bases was introduced by Buchberger (1965) in the context of his work on performing algorithmic computations in residue classes of polynomial rings. Buchbergers algorithm Buchberger (1985) the classical algorithm for computing the Groebner bases is given as follows:

Buchberger's algorithm uses the generalized Euclidean division algorithm to find out the remainder polynomial by dividing a polynomial by a set of polynomials. As given in the algorithm for an ideal $\langle e_1, ..., e_m\rangle$, all S-polynomials are calculated and are reduced with respect to $(e_1, ..., e_m)$. If $rem(S(e_i, e_j), (e_1, ..., e_m)) \neq 0$ then $(e_1, ...e_m)$ is to be extended with $rem(S(e_i, e_j), (e_1, ..., e_m))$. Next, all S-polynomials are calculated which already have not been calculated and are reduced similarly. If $rem(S(e_i, e_j), (e_1, ..., e_m)) \neq 0$ then $lm(rem(S(e_i, e_j), (e_1, ..., e_m))) \notin \langle lm(f_1, ..., lm(f_k))\rangle$. So, each time the generating set is extended, the monomial ideal generated by leading monomial of the generator is also extended. The termination is guaranteed by Dicksons lemma. And result is the Groebner bases. For details one may refer to Froeberg (1999). The complexity of the algorithm is closely related to the total degree of the intermediate polynomials that

Algorithm 1 The Buchberger's Algorithm

Require: $E = (e_1, ..., e_m)$

Ensure: A Groebner basis $G = (g_1, ..., g_s)$ for $I = \langle e_1, ..., e_m \rangle$

1: Initialize $G = E$

2: **repeat**

3: $\acute{G} = G$

4: **for** each pair p, q, $p \neq q$ in $\acute{G}$ **do**

5: Combine each p, q by cancelling the leading terms to get $S(p, q)$

6: Compute the remainders of $S(p, q)$ by G

7: Augment the non-zero remainders with G

8: **end for**

9: **until** $G = \acute{G}$

are generated during the execution of the algorithm. In the worst case it is known to run in double exponential time and on average its running time seems to be single exponential. A number of modifications of the algorithm exist to improve its performance regarding implementation. These include especially F_4 Faugère (1999) and F_5 Faugère (2002). Algorithms proposed by Faugere which are based on the idea of viewing Euclidean division algorithm in terms of matrix reduction algorithm. These are the fastest implementations of algorithm for finding Groebner bases so far. These fast implementations made them suitable for algebraic attacks also Faugere and Joux (2003). Some researches have also compared the efficiency of Groebner basis algorithms and XL algorithm. Groebner basis algorithms have been established to be more efficient Gwenole Ars and Sugita (2004); Makoto Sugita and Imai (n.d.).

2.3 Algebraic Cryptanalysis of Stream Ciphers With Non-linear Update

As discussed in previous sections, algebraic cryptanalysis has been proved successful for stream ciphers where linear update of states is being used with some non-linear component. The most common design is that of an LFSR with non-linear Boolean function s. In this case, degree of the non-linear equations is an important parameter to test the complexity of algebraic attack. The main efforts in this situation are directed towards formulating any method to reduce the degrees of equations. In this context researches on developing efficient algorithms for finding annihilators of Boolean functions are important. However, for stream ciphers in which internal state is updated non-linearly, degrees of the equations which relate internal states with output bits, increase with each clock. This creates a difficulty in applying conventional method of algebraic cryptanalysis on stream ciphers with non-linear update. That is why, algebraic attack is generally considered difficult for stream ciphers with non-linear update.

2.4 Conclusion

Algebraic cryptanalysis of stream ciphers are reviewed in this chapter. A description of the equation generation for stream ciphers and different techniques for lowering the degrees of equations is given. We have also briefly discussed different methods used for solution of non-linear equations.

Chapter 3

Algebraic Analysis of A5/1 and A5/2

3.1 Introduction

Global System for Mobile communication (GSM) is a global standard for digital cellular communication. GSM is the name of a standardization group that was established in 1982. Today GSM mobile phones are used world wide for communication, via voice and short-message-service (SMS) text. A5 is a collective name for encryption algorithms used in GSM communication. A5/0 means no encryption. A5/1 is the "standard" encryption algorithm, while A5/2 is the "export", algorithm. A5/3 is a new algorithm based on the UMTS/WCDMA algorithm Kasumi that will make the future standard for GSM.

A5/1 and A5/2 possess a simple clock controlled structure. Clock control is one of the mechanisms employed to introduce non-linearity into key stream generator built from linear feedback shift registers. Many cryptanalytic attacks against both A5/1 and A5/2 can be found in literature. Soon after that the structures of A5/1 and A5/2 were unveiled Briceno et al. (1999), these two have been extensively analyzed. First ever attack against A5/1 on its alleged structure is carried out by Golic Golic (1997). Successive are many time-memory trade off, search algorithms, correlation and hardware based attacks Biham and Dunkelman (2000); J.D.Golic and Conner (1994); Krause (2002); Ekdahl and Johansson (2003); T A. Biryukov and Wagner (2001); Pornin and Stern (2000). In this chapter, we will consider the algebraic analysis of A5/1 and A5/2 ciphers. Algebraic attack against clock controlled stream ciphers has been studied in Al-Hinai et al. (2006)

where mainly those ciphers are considered in which one generator controls the clocking of other generators. Due to mutual clocking of registers of A5/1, algebraic equations which relate its internal states with output bits obtain very high and varying in time degrees. However, due to weak structure of A5/2 most of the attacks against it are algebraic in nature. Generally, flaws in the GSM protocol are exploited in these attacks and opportunely they are extended to cipher-text only attacks I. Goldberg and Green (1999); Petrovic and Fuster-Sabater (2000); Barkan et al. (2003); A. Bogodanov and Rupp (2003).

Organization of this chapter is as follows: Section 3.2 and Section 3.3 gives algebraic analysis of A5/1 and A5/2 respectively. Chapter is concluded in Section 3.4.

3.2 Algebraic Analysis of A5/1

3.2.1 Design Structure of A5/1

The design of A5/1 was initially kept secret from public, but its design was reverse engineered in 1999 M. Blaze and Roe (1999). A5/1 is a binary additive stream cipher based on irregular clocking of three maximal length linear feedback shift registers. The three registers are of lengths 19, 22 and 23, thus, the key size is 64 bits and the key-stream is produced by XOR-ing the output from the three registers. The LFSRs are clocked in an irregular fashion. It is a type of stop/go clocking which is based on the majority rule. Each register has a single clocking bit. A majority function is calculated from the clocking bit of each register which are input bits of the function. This function output is 1 or 0 if two or more of the input bits are 1 or 0 respectively. Based on the output of majority function, a register is clocked if its clock bit contains a value equal to this majority value. The binary clock-controlled bits are derived from each register by using stop/ go rule described above and key-stream is then produced as the XOR of the output bits of the three

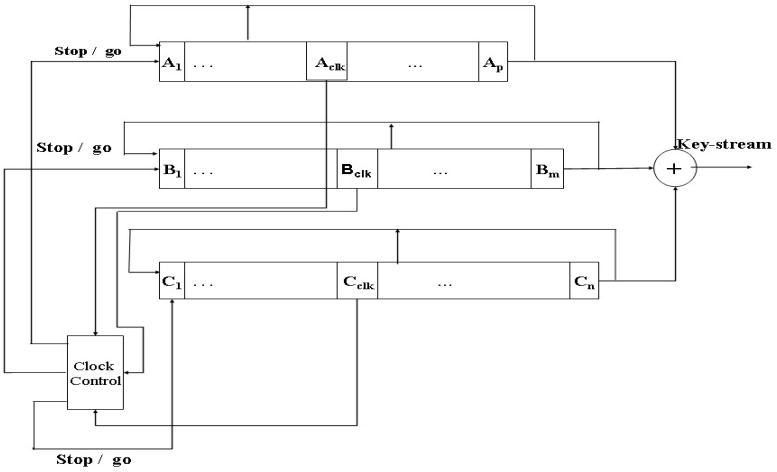

Figure 3.1: The Structure of A5 Key-Stream Generator

registers.

The structure of A5/1 algorithm is shown in Figure 3.1. It has three primitive linear feedback shift registers R_1, R_2 and R_3 of lengths p, m and n, so they produce maximum length sequences when clocked regularly. Their initial states are represented as $A_1...A_p, B_1, ...B_m$ and $C_1, ...C_n$.

Each register has a single clocking bit represented as A_{clk}, B_{clk} and C_{clk}. The binary clock-controlled bits are derived from each register by using stop/ go rule as described in the previous section and key-stream is then produced as the XOR of the output bits of the three registers. In the case of A5/1 algorithm of GSM, the lengths are respectively 19, 22 and 23. R_1, R_2 and R_3 are clocked respectively iff $maj(A_{clk}, B_{clk}, C_{clk}) = A_{clk}$, $maj(A_{clk}, B_{clk}, C_{clk}) = B_{clk}$, and $maj(A_{clk}, B_{clk}, C_{clk}) = A_{clk}$. where $maj(X, Y, Z) = XY \oplus XZ \oplus YZ$.

The relation of output bit of LFSR A with its state bits can be written as:

$$A_p^t = A_p^{t-1}(A_{clk}^{t-1} \oplus A_{clk}^{t-1}B_{clk}^{t-1} \oplus A_{clk}^{t-1}C_{clk}^{t-1} \oplus B_{clk}^{t-1}C_{clk}^{t-1})$$

$$\oplus A_{p-1}^{t-1}(1 \oplus A_{clk}^{t-1} \oplus A_{clk}^{t-1}B_{clk}^{t-1} \oplus A_{clk}^{t-1}C_{clk}^{t-1} \oplus B_{clk}^{t-1}C_{clk}^{t-1}) \quad (3.1)$$

If we represent $f_{t-1} = A_{clk}^{t-1}B_{clk}^{t-1} \oplus A_{clk}^{t-1}C_{clk}^{t-1} \oplus B_{clk}^{t-1}B_{clk}^{t-1}$ then equation 3.1 can be written as:

$$A_p^t = A_p^{t-1}(A_{clk}^{t-1} \oplus f_{t-1}) \oplus A_{p-1}^{t-1}(1 \oplus A_{clk}^{t-1} \oplus f_{t-1}) \tag{3.2}$$

Or, $A_p^t = A_{p-1}^{t-1} \oplus (A_{p-1}^{t-1} \oplus A_p^{t-1})(A_{clk}^{t-1} \oplus f_{t-1})$; Similarly:

$B_m^t = B_{m-1}^{t-1} \oplus (B_{m-1}^{t-1} \oplus B_m^{t-1})(B_{clk}^{t-1} \oplus f_{t-1})$; And $C_n^t = C_{n-1}^{t-1} \oplus (C_{n-1}^{t-1} \oplus C_n^{t-1})(C_{clk}^{t-1} \oplus f_{t-1})$

Since, key-stream bit at time t is found by taking addition modulo 2 of the output bits at time t of the three LFSRs, therefore, key-stream bit at time t can be represented as:

$z^t = A_p^t \oplus B_m^t \oplus C_n^t \Rightarrow z^t = (A_{p-1}^{t-1} \oplus B_{m-1}^{t-1} \oplus C_{n-1}^{t-1}) \oplus (A_{p-1}^{t-1} \oplus A_p^{t-1})(A_{clk}^{t-1} \oplus f_{t-1})$

$$\oplus (B_{m-1}^{t-1} \oplus B_m^{t-1})(m_{clk}^{t-1} \oplus f_{t-1}) \oplus C_{n-1}^{t-1} \oplus C_n^{t-1})(C_{clk}^{t-1} \oplus f_{t-1}) \tag{3.3}$$

This system of equations represents A5/1 cipher, but it can be seen that degree of this system of equations will increase at each step.

Reducing the degree of the equations is primary in the algebraic attack. Taking binary derivative to simplify the degree of equations. From equation 3.3 we can obtain:

$$A_p^{t+1} = A_p^t(A_{clk}^t \oplus f_t) \oplus A_{p-1}^t(1 \oplus A_{clk}^t \oplus f_t) \tag{3.4}$$

From equation 3.1 and 3.3 we have:

$\Rightarrow A_p^{t+1} \oplus A_p^t = A_p^t(1 \oplus A_{clk}^t \oplus f_t) \oplus A_{p-1}^t(1 \oplus A_{clk}^t \oplus f_t) = (1 \oplus A_{clk}^t \oplus f_t)(A_p^t \oplus A_{p-1}^t)$

Thus we obtain:

$\Rightarrow z^t \oplus z^{t+1} = (A_p^t \oplus A_{p-1}^t \oplus B_m^t \oplus B_{m-1}^t \oplus C_n^t \oplus C_{n-1}^t)(1 \oplus f_t)$

$$\oplus A_{clk}^t(A_p^t \oplus A_{p-1}^t) \oplus B_{clk}^t(B_m^t \oplus B_{m-1}^t) \oplus C_{clk}^t(C_n^t \oplus C_{n-1}^t) \tag{3.5}$$

$$\Rightarrow (z^t \oplus z^{t+1}).f_t = (A^t_{clk}(A^t_p \oplus A^t_{p-1}) \oplus B^t_{clk}(B^t_m \oplus B^t_{m-1}) \oplus C^t_{clk}(C^t_n \oplus C^t_{n-1})).f_t \quad (3.6)$$

As it is known that f_t is of degree 2, therfore, after first iteration degree of the developed equations will increase at each step. Yet another approach to reduce the degree of equations can be considered as follows: from equation 3.5:

$$(z^t \oplus z^{t+1})(1 \oplus A^t_p \oplus A^t_{p-1} \oplus B^t_m \oplus B^t_{m-1} \oplus C^t_n \oplus C^t_{n-1}) = (A^t_{clk}(A^t_p \oplus A^t_{p-1}) \oplus B^t_{clk}(B^t_m \oplus B^t_{m-1}) \oplus A^t_{clk}(C^t_n \oplus C^t_{n-1}))(1 \oplus A^t_p \oplus A^t_{p-1} \oplus B^t_m \oplus B^t_{m-1} \oplus C^t_n \oplus C^t_{n-1})$$

Here, if we consider the case of consecutive same bits then the equations will become:

$$(A^t_{clk}(A^t_p \oplus A^t_{p-1}) \oplus B^t_{clk}(B^t_m \oplus B^t_{m-1}) \oplus ACt_{clk}(C^t_n \oplus C^t_{n-1}))(1 \oplus A^t_p \oplus A^t_{p-1} \oplus B^t_m \oplus B^t_{m-1} \oplus C^t_n \oplus C^t_{n-1}) = 0 \Rightarrow (A^t_p \oplus A^t_{p-1})(B^t_m \oplus B^t_{m-1})(A^t_{clk} \oplus B^t_{clk}) \oplus (A^t_p \oplus A^t_{p-1})(C^t_n \oplus C^t_{n-1})(A^t_{clk} \oplus C^t_{clk}) \oplus (B^t_m \oplus B^t_{m-1})(C^t_n \oplus C^t_{n-1})(B^t_{clk} \oplus C^t_{clk}) = 0$$

or

$$A^t_{clk}(A^t_p \oplus A^t_{p-1})(B^t_m \oplus B^t_{m-1} \oplus C^t_n \oplus C^t_{n-1}) \oplus B^t_{clk}(B^t_m \oplus B^t_{m-1})(A^t_p \oplus A^t_{p-1} \oplus C^t_n \oplus C^t_{n-1}) \oplus C^t_{clk}(C^t_n \oplus C^t_{n-1})(A^t_p \oplus A^t_{p-1} \oplus B^t_m \oplus B^t_{m-1}) = 0 \quad (3.7)$$

equation 3.7 also gives product of three variables like previous equations. But here, since we are looking for consecutive same bits, therefore, some equations obtained from the initial clocking will be discarded, and this will lead to even more rapid increase in the degree of equations. In this case however, increase in the degrees varies with different initial state bits.

For solving non-linear equations, we adopted the method of guess and determine and used F_4 algorithm implemented in Magma. Since, clocking of registers is interdependent, therefore if one register or even two complete registers are guessed, third register cannot be recovered. Since, middle bits control the clocking of LFSRs, therefore, degrees of the algebraic equations increase

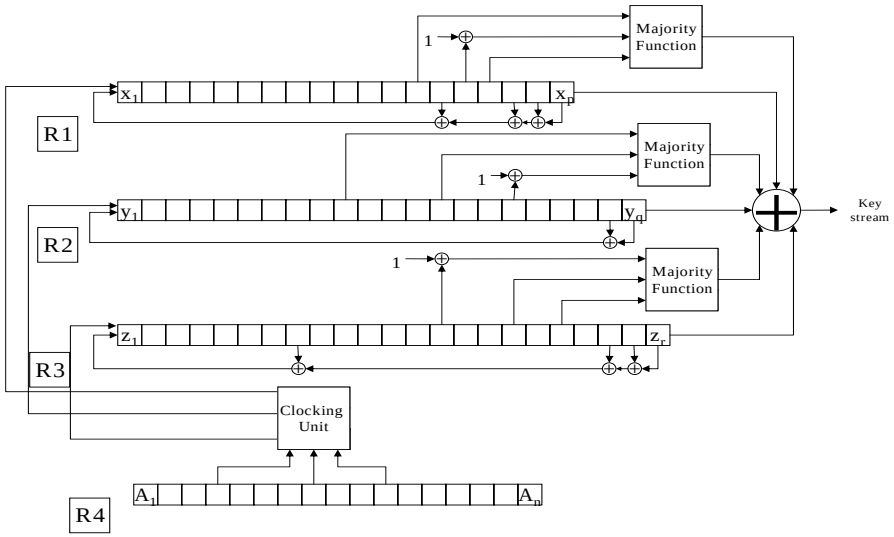

Figure 3.2: The Structure of A5/2 Key-Stream Generator

immediately after first clocking. Therefore, to obtain some linear equations, guessed bits must be from the clocking bits to starting bits of each LFSR. Guessing bits in this way, we can obtain nineteen out of 64 internal state bits in a fraction of a second on a PC with CPU at 1.73 GHz and I GB RAM.

3.3 Algebraic Cryptanalysis of A5/2

3.3.1 System of Algebraic Equations for A5/2 Key-Stream Generator

The generic structure of A5/2 cipher is given in Figure 3.2. It shows the algebraic relation of internal states of A5/2 with output bits. The cipher has four LFSRs represented as R_1, R_2, R_3 and R_4 of lengths p, q, r and n respectively. The connection polynomials of all the registers are primitive, that is when regularly clocked they are maximal length registers.

The initial states of the four registers are represented as $x_1...x_p, y_1, ...y_q$, $z_1, ...z_r$ and $A_1, ...A_n$. For A5/2 these lengths p, q, r, and n are 19, 22, 23 and 17 respectively. Initialization procedure is

linear in 64 bits key and 22 bits publicly known frame. For details of initialization procedure one may refer to E. Barkan and Keller (2003); Krause (2002). After initialization, stop/go clocking of R_1, R_2, and R_3 is started according to the feedback given in Figure 3.2. R_4 controls the irregular clocking of the rest of three registers. R_4 is regularly clocked and at each clock R_1, R_2 and R_3 are clocked according to the following rule: R_1, R_2 and R_3 are clocked respectively iff $maj(A_4, A_8, A_{11}) = A_{11}$, $maj(A_4, A_8, A_{11}) = A_4$, and $maj(A_4, A_8, A_{11}) = A_8$. where $maj(X, Y, Z) = XY \oplus XZ \oplus YZ$. Thus, in each cycle at least two of the three registers are clocked. Middle bits of the three registers therefore, contribute in deciding of their clocking or not. At each cycle, one output bit at time t is produced as:

$$output - bit^t = x_{19}^t \oplus maj(x_{13}^t, 1 \oplus x_{15}^t, x_{16}^t) \oplus y_{22}^t \oplus maj(y_{10}^t, y_{14}^t, 1$$

$$\oplus y_{17}^t) \oplus z_{23}^t \oplus maj(1 \oplus z_{14}^t, z_{17}^t, z_{19}^t) \tag{3.8}$$

Due to the irregular clocking algebraic relation of state bits of register R_1 with previous states can be written as follows:

$x_p^t = x_p^{t-1} \cdot (A_4^{t-1} \oplus maj(A_4^{t-1}, A_8^{t-1}, A_{11}^{t-1})) \oplus x_{p-1}^{t-1} \cdot (1 \oplus A_4^{t-1} \oplus maj(A_4^{t-1}, A_8^{t-1}, A_{11}^{t-1}))$ where $p = 2..19$

and

$$x_1^t = x_1^{t-1} \cdot (A_4^{t-1} \oplus maj(A_4^{t-1}, A_8^{t-1}, A_{11}^{t-1})) \oplus feedbackR1^{t-1} \cdot (1 \oplus A_4^{t-1} \oplus maj(A_4^{t-1}, A_8^{t-1}, A_{11}^{t-1}))$$

where $feedbackR1^{t-1} = x_{14}^{t-1} \oplus x_{17}^{t-1} \oplus x_{18}^{t-1} \oplus x_{19}^{t-1}$

From above expression, we can find the contribution of R_1 register into the XOR of output bit at time t as:

$$(x_{19}^{t-1} \oplus maj(x_{13}^{t-1}, 1 \oplus x_{15}^{t-1}, x_{16}^{t-1})) \cdot (A_{11}^{t-1} \oplus maj(A_4^{t-1}, A_8^{t-1}, A_{11}^{t-1})) \oplus (x_{19}^{t} \oplus$$

$$maj(x_{13}^{t}, 1 \oplus x_{15}^{t}, x_{16}^{t})) \cdot (1 \oplus A_{11}^{t-1} \oplus maj(A_4^{t-1}, A_8^{t-1}, A_{11}^{t-1})) \tag{3.9}$$

Similarly outputs from R2 and R3 can be obtained as follows:

$$(y_{22}^{t-1} \oplus maj(y_{10}^{t-1}, y_{14}^{t-1}, 1 \oplus y_{17}^{t-1})) \cdot (A_4^{t-1} \oplus maj(A_4^{t-1}, A_8^{t-1}, A_{11}^{t-1})) \oplus (y_{22}^{t} \oplus$$

$$maj(y_{10}^{t}, 1 \oplus y_{14}^{t}, y_{17}^{t})) \cdot (1 \oplus A_4^{t-1} \oplus maj(A_4^{t-1}, A_8^{t-1}, A_{11}^{t-1})) \tag{3.10}$$

and

$$(z_{23}^{t-1} \oplus maj(z_{14}^{t-1}, z_{17}^{t-1}, 1 \oplus z_{19}^{t-1})) \cdot (A_8^{t-1} \oplus maj(A_4^{t-1}, A_8^{t-1}, A_{11}^{t-1})) \oplus (z_{23}^{t} \oplus$$

$$maj(z_{14}^{t}, 1 \oplus z_{17}^{t}, z_{19}^{t})) \cdot (1 \oplus A_8^{t-1} \oplus maj(A_4^{t-1}, A_8^{t-1}, A_{11}^{t-1})) \tag{3.11}$$

From above expressions, it can be seen that degrees of algebraic equations, which relate internal states of the registers with output bits, increase with time. But this increase is only because of the bits from R_4 which is the smallest register. This situation invites guessing of R_4 bits as is adopted in major attacks against A5/2 E. Barkan and Keller (2003); Krause (2002). And the failure of this approach in case of A5/1 is attributed to the mutual dependency of registers for clocking, guessing one register or few bits of all registers does not help in reducing the degrees of all algebraic equations. Our emphasis here, is on the fact that apart from cryptanalysis friendly protocol of GSM, the structure of A5/2 itself is vulnerable to very efficient algebraic attack.

3.3.2 Experimental Results on Algebraic Cryptanalysis of A5/2 Using Groebner Basis

Our implementation is software based and approach is similar to E. Barkan and Keller (2003). The difference lies in the application of Groebner basis to solve algebraic equations thus formed by relating internal states of R_1, R_2, and R_3 with the output bits, while R_4 is guessed.

Groebner basis can be efficiently used for solving algebraic equations and so, their use for algebraic cryptanalysis is quite well known. Though, exact complexity of algorithm for finding Groebner basis is not yet known. Therefore, it is important to find experimentally the data and time complexity of solving equations with Groebner basis. Our simulation environment is same as described in Chapter 1 and thus, system of non-linear equations is solved using F_4 algorithm implemented in Magma version 2.13-5 System (n.d.)

It can be observed easily that protocol independent A5/2 cipher is also an easy prey to algebraic cryptanalysis. Using the same approach as has been used in previous attacks, bits from R_4 are guessed and for each guess, system of quardratic equations is developed. Our experiments show that second degree equations can be solved in 0.42 sec to recover 64 secret internal state bits with a few more than 500 equations of one cycle.

When A5/2 runs in GSM protocol, after one initialization only 114 equations can be obtained from each frame. For detail of the GSM protocol E. Barkan and Keller (2003) is referred. Also, one bit of each register is known, after loading 64 keybits and 22 frame bits. So, we have to guess 16 bits of R_4 to recover 61 bits of the other three registers. An important consideration is that for each register R_i, XOR difference between Ri_f and Ri_{f+1} is known, where f and $f + 1$ represent two different frames. Initial states of three registers of one frame (just before 99 clocking) can be taken as 61 variables and initial states of next frames can be written linearly in these variables. Thus, we can make a system of quadratic equations in 61 variables, 114 equations from each frame. Then the number of frames to obtain enough equations to solve the system is to be decided. Linearization of quadratic equations of 61 variables renders a linear system of 655 variables. However, it is found experimentally E. Barkan and Keller (2003) that 450 equations from four frames are enough to solve the equations of A5/2 cipher due to the presence of a number

of linear variables of system and other variables expressed as products of those for each frame.

We have tested experimentally that algebraic equations of A5/2 of any three frames can be solved with Magma, in 2 to 3 seconds. It should be noted here that we need to solve this system of equations $2^{16} - 1$ times that is for each non-zero guess of R_4. For each wrong guess, the system of equations becomes inconsistent, and Groebner basis of such a system is found to be one. The inconsistency of the system is decided in a fraction of a second. Thus, overall time complexity of our approach is comparable to E. Barkan and Keller (2003), but we need less data.

Ciphertext-only attack described in E. Barkan and Keller (2003), exploits the structured redundancy in the message which is introduced into the message because of employment of error correction code prior to encryption. However, the technical difference between ciphertext-only and known-plaintext attacks is the requirement of data frames. In ciphertext-only attack, only 272 equations can be extracted from 456 bits of cipher text (4 data frames). Thus, the ciphertext-only attack of E. Barkan and Keller (2003), requires eight frames to obtain enough equations for linearization. But when we solve equations using Groebner basis, 272 equations are actually enough to solve the equation to obtain 61 unknowns. Time required, however, to solve 272 equations is nearly 30 sec. But in this case also the inconsistency is decided in less than a second time. Thus, if same equations are solved with Groebner basis algorithm, only four data frames will be needed instead of eight. Data requirements for our and previous (E. Barkan and Keller (2003)) experimental results are summarized in Table 3.1, time complexity is comparable in both cases:

3.4 Conclusion

Mostly those ciphers can be successfully algebraically attacked in which relation of output bits can be easily developed in terms of state bits of LFSRs. If this is not the case, then mounting

Table 3.1: Experimental results of solving equations of A5/2 with groebner basis

Att.	Required Data Frames
Known Plain-Text attack E. Barkan and Keller (2003)	4
Known plain-text with Groebner basis	3
Cipher text only attack E. Barkan and Keller (2003)	8
Cipher text only with Groebner basis	4

an algebraic attack becomes difficult. Successful Algebraic attacks on clock-controlled stream ciphers show that security of clock-controlling must also be checked against algebraic attack along with correlation and other divide and conquer type of attacks. In this context the analysis A5/1 key-stream generator against algebraic attack is important and is achieved in this research. It is found that this type of clocking shows more resistance to algebraic attack rather than the irregular clocking used in stop-and-go generator; alternating step generator, self-decimated and step1/step2 generator. Due to mutual dependence of registers in clocking, guess and determine approach is not very successful and very high degree equations are formed. We also found that at maximum, 19 state bits can be recovered for A5/1 generator if remaining bits are guessed.

An evidence of experiments using Groebner basis is also presented to enhance previous results on algebraic cryptanalysis of A5/2 cipher of GSM. Our results demonstrate that we need three data frames instead of four to mount a known plaintext attack with the same time complexity. And

in case of cipher-text only attack our requirement is of four data frames as compared to eight of previous attacks.

Chapter 4

Algebraic Analysis of Grain Family of Ciphers

4.1 Introduction

Stream ciphers with keystream generators are very commonly used for secure communication. However, it is important to analyze a keystream generator against existing cryptanalytic attacks. In this context cryptanalysis is generally considered a tool to measure the strength of a cipher in terms of its resistance against different types of attacks. Algebraic cyptanalysis is a very important development in this field and has received a lot of attention in many researches as discussed in previous chapters. In this chapter, we focus on the algebraic cryptanalysis of non-linear feedback shift register (NLFSR) based stream cipher. Grain M. Hell and Meier (2005a,b), one of the ECRYPT stream cipher project eSTREAM (2005) candidates, has a NLFSR along with a non-linear Boolean function and a linear feed back shift register (LFSR). Till third phase of eSTREAM project, two versions of Grain have emerged; and we have analyzed both of these versions. Existing attacks on Grain version 1 (Grain-v1), includes: a correlation attack based on linear approximation Berbain et al. (2006) and a distinguishing attack using linear circuit approximation S. Khazaei and Kiaei (n.d.). Whereas Grain version 2 (Grain-128) has not yet faced any serious attack.

In case of Grain family of ciphers, and other such possible structures in which NLFSRs are used, internal state of the cipher is updated nonlinearly and as a result algebraic equations possess continually rising degrees. In this work, algebraic equations for both versions of Grain are generated

and also analyzed thoroughly to obtain low degree equations. An approach of guess and determine by solving algebraic equations is also discussed. Further, the importance of guessing bits at different positions is also presented. Thus, we try to figure out the maximum number of bits that can be recovered while other bits are guessed.

Both versions of Grain have an initialization procedure with a secret key and an initialization vector. We, however, analyze the cipher after initialization. So, the target here is to recover the secret initial states of LFSR and NLFSR, as is the case generally in algebraic attacks, rather than to recover the secret key which is an input of initialization step.

Organization of this chapter is as follows: Section 4.2 presents a brief account of both versions of Grain. In Section 4.3 and 4.4, we describe algebraic analysis of equations generated for Grain-v1 and Grain-128. A study of variation in the degrees of algebraic equations with different state bits guessed is also presented. Section 4.5 wil summarize and conclude this chapter.

4.2 Design of Grain-v1 and Grain-128

Grain family of ciphers has an NLFSR, along with a non-linear Boolean function and an LFSR. Till third phase of eSTREAM project, two versions of Grain have emerged; and we have considered both of these versions for our experiments. The basic structure of Grain has an internal state of size 2n bits, with n being the secret key bits. The internal state is divided into NLFSR and LFSR of length n bits each. At each clock pulse, selecting some bits of the LFSR and some of the NLFSR and applying a Boolean function one key stream bit is produced. The general structure of Grain family of ciphers is given in Figure 4.1.

The design of Grain-v1 uses LFSR and NLFSR of 80 bits each and a Boolean function of 11 variables as the feedback function of the NLFSR and another Boolean function of five variables to

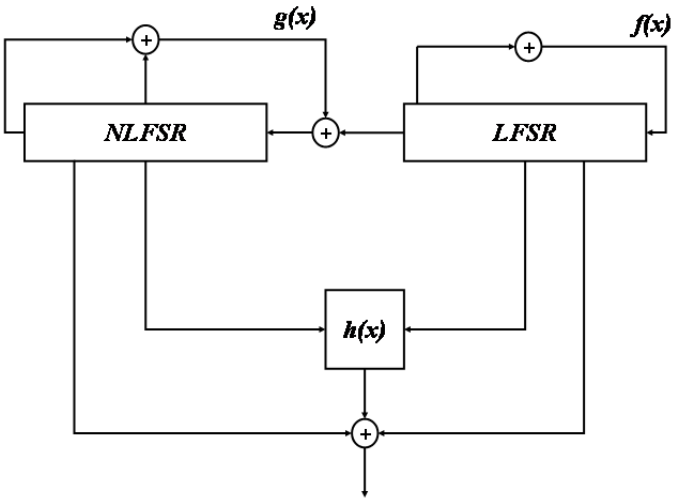

Figure 4.1: An Overview of Grain Cipher

filter the contents of five fixed bits of the internal states of LFSR and NLFSR. The key generation procedure for the Grain-v1 is given as the following code where the initial states of the LFSR and NLFSR are denoted as $(y_1, y_2, ...y_{80})$ and $(x_1, x_2, ...x_{80})$:

procedure Grain-v1$((y_1..y_{80}), (x_1, ..x_{80}))$

1: **for** $k = 1$ to n **do**

2: $keybit_k := x_1 + h(x_{64}, y_{65}, y_{47}, y_{26}, y_4)$

3: $feedbackLFSR := (y_1, y_{14}, y_{24}, y_{39}, y_{52}, y_{63})$

4: $feedbackNLFSR := (y_1 + g(x_{64}, x_{61}, x_{53}, x_{46}, x_{38}, x_{34}, x_{29}, x_{22}, x_{16}, x_{10}, x_1))$

5: **for** $i = 1$ to 79 **do**

6: $y_i := y_{i+1}$

7: $x_i := x_{i+1}$

8: $y_{80} := feedbackLFSR$

9: $x_{80} := feedbackNLFSR$

10: **end for**

11: **end for**

where function g and h are given as:

$g(x_{11}, x_{10}, x_9, x_8, x_7, x_6, x_5, x_4, x_3, x_2, x_1) = x_{11} + x_{10} + x_9 + x_8 + x_7 + x_6 + x_5 + x_4 + x_3 + x_2 + x_1 + x_{11}x_{10} + x_7x_6 + x_3x_2 + x_{10}x_9x_8 + x_6x_5x_4 + x_{11}x_8x_5x_2 + x_{10}x_9x_7x_6 + x_{11}x_{10}x_4x_3 + x_{11}x_{10}x_9x_8x_7 + x_6x_5x_4x_3x_2 + x_9x_8x_7x_6x_5x_4$

and

$h(x_5, x_4, x_3, x_2, x_1) = x_2+x_5+x_1x_4+x_3x_4+x_4x_5+x_1x_2x_3+x_1x_3x_4+x_1x_3x_5+x_2x_3x_5+x_3x_4x_5$

The main building blocks of Grain-128 are an LFSR and an NLFSR of length 128 each and an output Boolean function, similar to that of Grain-v1. With initial states of the LFSR and NLFSR denoted as: $(y_1, y_2, ..., y_{128})$ and $(x_1, x_2, ..., x_{128})$, the key bits are generated as follows:

procedure Grain-128 $((y_1, .., y_{128}), (x_1, .., x_{128}))$

1: **for** $k = 1$ to n **do**

2: $keybit_k := x_3+x_{16}+x_{37}+x_{46}+x_{65}+x_{74}+x_{90}+y_{94}+h(x_{13}, y_9, y_{14}, y_{21}, x_{96}, y_{43}, y_{61}, y_{80}, y_{96})$

3: $feedbackLFSR := (y_1, y_8, y_{39}, y_{71}, y_{82}, y_{97})$

4: $feedbackNLFSR := (y_1+g(x_1, x_{27}, x_{57}, x_{92}, x_{97}, x_4, x_{68}, x_{12}, x_{14}, x_{18}, x_{19}, x_{28}, x_{60}, x_{41}, x_{49}, x_{62}, x_{66}, x_{69}, x_{85}))$

5: **for** $i = 1$ to 127 **do**

6: $y_i := y_{i+1}$

7: $x_i := x_{i+1}$

8: $y_{128} := feedbackLFSR$

9: $x_{128} := feedbackNLFSR$

10: **end for**

11: **end for**

where g and h in this case are given by: $g(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}, x_{13}, x_{14},$

$x_{15}, x_{16}, x_{17}, x_{18}, x_{19}) = x_1 + x_2 + x_3 + x_4 + x_5 + x_6x_7 + x_8x_9 + x_{10}x_{11} +$

$x_{12}x_{13} + x_{14}x_{15} + x_{16}x_{17} + x_{18}x_{19}$

and

$h(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9) = x_1x_2 + x_3x_4 + x_5x_6 + x_7x_8 + x_1x_5x_9$

4.3 Alegbraic Analysis of Grain-v1

As mentioned earlier, the internal state of Grain Version 1 is of 160 bits, with 80 bits each of LFSR and NLFSR. If the initial state of NLFSR is labeled as: $x_1, x_2, .., x_{80}$, and those of LFSR as $y_1, y_2, .., y_{80}$. According to the structure of the cipher, given earlier, output bit stream of the cipher can be expressed in these variables representing the internal state bits. Consider for example, the following five output bit expressions. Equating these expressions with the known output bits, gives us a non-linear system of algebraic equations in 160 variables.

$x_1 + y_{26} + x_{64} + y_4y_{65} + y_{47}y_{65} + x_{64}y_{65} + y_4y_{26}y_{47} + y_4y_{47}y_{65} + y_4y_{47}x_{64} + y_{26}y_{47}x_{64} + y_{47}y_{65}x_{64},$

$x_2 + y_{27} + x_{65} + y_5y_{66} + y_{48}y_{66} + x_{65}y_{66} + y_5y_{27}y_{48} + y_5y_{48}y_{66} + y_5y_{48}x_{65} + y_{27}y_{48}x_{65} + y_{48}y_{66}x_{65},$

$x_3 + y_{28} + x_{66} + y_6y_{67} + y_{49}y_{67} + x_{66} * y_{67} + y_6y_{28}y_{49} + y_6y_{49}y_{67} + y_6y_{49}x_{66} + y_{28}y_{49}x_{66} + y_{49}y_{67}x_{66},$

$x_4 + y_{29} + x_{67} + y_7y_{68} + y_{50}y_{68} + x_{67}y_{68} + y_7y_{29}y_{50} + y_7y_{50}y_{68} + y_7y_{50}x_{67} + y_{29}y_{50}x_{67} + y_{50}y_{68}x_{67},$

$x_5 + y_{30} + x_{68} + y_8y_{69} + y_{51}y_{69} + x_{68}y_{69} + y_8y_{30}y_{51} + y_8y_{51}y_{69} + y_8y_{51}x_{68} + y_{30}y_{51}x_{68} + y_{51}y_{69}x_{68}$

After a few clocks, some of the internal states become non-linear. And consequently, the succeeding equations which involve those state bits have even more higher degree. So, here, after 17 equations of degree 3, we obtain a collection of equations with degree 8. In a similar manner, degrees of equations continue to increase in steps. Figure 4.2 demonstrates the degrees of some

equations of Grain-1. It can be seen that after almost 80 equations, the degree of the equations becomes as high as 160, which is the maximum possible degree here.

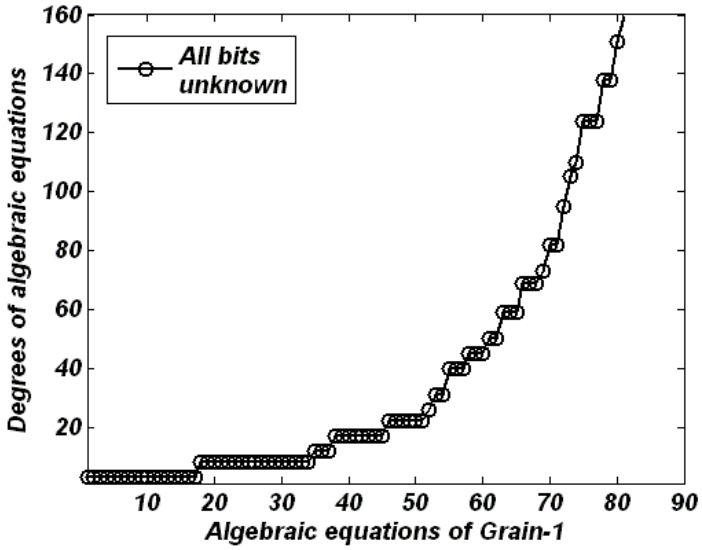

Figure 4.2: Variation in the degrees of algebraic equations of Grain-1, with all 160 unknown state bits

4.3.1 Lowering the Degrees of Equations

A major parameter which influences the complexity of an algebraic attack is the degree of the underlying algebraic system. When the transition is linear, any keystream bit can obviously be expressed as a function of the same degree as of the combining or filtering Boolean function f, in the initial state bits. It has been established, however, that the keystream generator may be vulnerable to algebraic attacks even if the degree of the algebraic function is high due to the existence of a low degree annihilator of f or of $(1 + f)$ Meier et al. (2004). The concept of annihilator cannot be used in the conventional way in the case of Grain due to the non-linear update of state bits. Next, we will discuss, however, the use of annihilator of to lower the degree of equations of Grain-v1.

The Boolean function h of five variables which is used to filter the contents of five fixed bits of the internal states of LFSR and NLFSR is of degree three. Second degree annihilator of h and $1 + h$ can be found. The second degree annihilator of h is $f_1(x_1, x_2, x_3, x_4, x_5) = 1 + x_1 + x_4 + x_5 + x_1.x_4 + x_1.x_5 + x_2.x_3 + x_3.x_5 + x_4.x_5$, whereas that of $1 + h$ is $f_2(x_1, x_2, x_3, x_4, x_5) = x_4 + x_1.x_3 + x_1.x_4 + x_1.x_5 + x_2.x_3 + x_4.x_5$. Since $keybit_k = x_1 + h(x_{64}, y_{65}, y_{47}, y_{26}, y_4)$, therefore the use of annihilator is dependent on the value of x_1, that is a state bit of NLFSR. Hence, with guessed bits of NLFSR, when the output bit is generated with the annihilator as given below the degree of the overall equations can be reduced by one:

if $outputbit + x_1 = 1$ **then**

$output = f_1(x_{64}, y_{65}, y_{47}, y_{26}, y_4)$

else

if $outputbit + x_1 = 0$ **then**

$output := f_2(x_{64}, y_{65}, y_{47}, y_{26}, y_4)$

end if

end if

It is also important to note here, that an equation with annihilator can only be generated if both the keystream and x_1 bits are known. Therefore, only eighty equations can be generated when all the eighty bits of NLFSR are guessed. Since, update function of NLFSR involves bits from LFSR also, therefore, further equations can only be generated when some bits of LFSR are also guessed. Therefore, if we have to reduce the number of guessing bits, we will obtain lesser equations. Thus, to make use of annihilator for lowering the degrees of equations, it is necessary to limit the number of equations to be genarated. Once equations are generated, next step of an algebraic attack is to solve them. Discussion given below will show how we can solve algebraic

equations when bits from LFSR are guessed. But still some other methods of solving equations like Raddum (n.d.) can be tried to solve a small set of equations that are generated with some bits of NLFSR guessed using annihilators for low degree equations.

Simulations are performed within the same resources as mentioned in Chapter 1. System of non-linear equations has been solved with F_4 algorithm implemented in Magma System (n.d.).

A number of experiments is performed to find out the maximum number of bits that can be recovered while others are guessed, by guessing bits at different positions. Our experiments show that out of all 160 bits, no more than 77 bits can be recovered, while remaining 83 are guessed. The next important thing is to know that which bits must be guessed to give best results.

If the guessed bits give some linear equations, then these equations along with some high degree equations can be solved to give unknown bits. To give a comparison between degrees of algebraic equations, where 83 bits of both LFSR and NLFSR cipher are guessed at different positions, we plot degrees of equations as they grow with generated equations. Figure 4.3 (a) demonstrates the rise in the degree of algebraic equations of Grain-1 when all state bits of LFSR are guessed, whereas those of NLFSR are unknown. Figure 4.3 (b) and (c) illustrate the degrees when three bits of NLFSR at its first and last positions are guessed respectively alongwith all guessed bits of LFSR. In a similar manner, Figure. 4.4 (a), (b) and (c) illustrate the degrees when all bits of NLFSR and 3 bits at different positions of LFSR are guessed. The fact that we cannot guess less than 83 bits is attributed to the memory requirements for solving equations thus formed, by F4 algorithm in Magma. Adding even one more unknown variable makes the memory requirements beyond the available.

Considering Figure 4.3, almost identical increase in the degrees of equations can be observed for all the three cases. Among some very high degree equations, we obtain some equations of degree

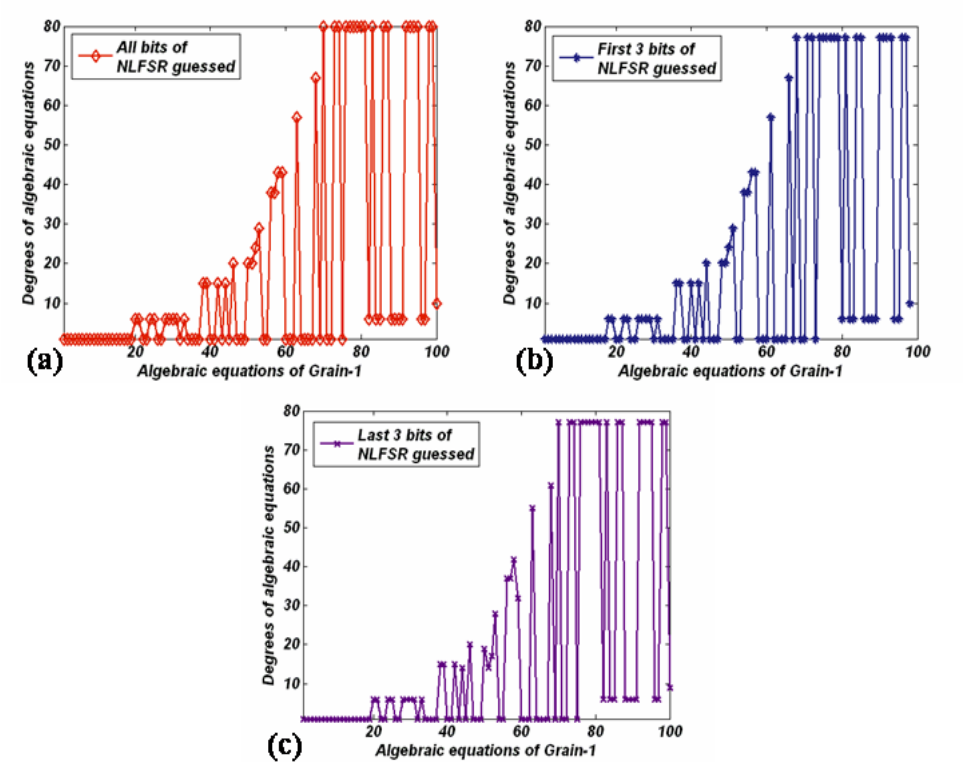

Figure 4.3: Variation in the degrees of algebraic equations of Grain-v1, with all 80 bits of LFSR, and 3 different internal state bits of NLFSR guessed.

as low as 3. In this situation, we choose the approach of solving equations in steps. That is, first, we solve some lower degree equations among them to obtain values of some of the unknowns. In the subsequent step, these obtained values replace the variables in equations. As a result, some other lower degree equations are obtained, which can next be solved to give the remaining unknowns. Table 4.1, explains our approach in solving equations of Grain-v1.

The First column of Table 4.1 gives the three different positions of the guessed bits as illustrated in Figure.4.3. Corresponding to all of these three cases considered here, we give the time and memory taken for each computation with the above mentioned resources. Only the situation when all bits of LFSR are guessed and 3 last bits of NLFSR are guessed, results in solution of the equations to give 77 unknown bits. In first step, 48 linear equations are chosen from 150 generated

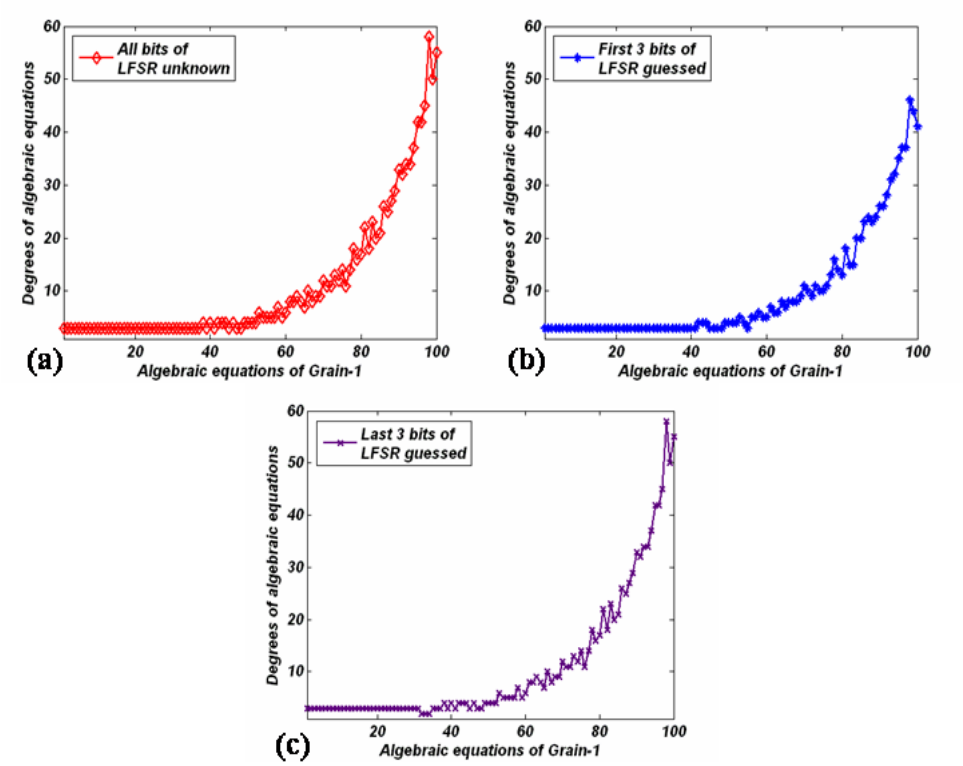

Figure 4.4: Variation in the degrees of algebraic equations of Grain-v1, with all 80 bits of NLFSR, and 3 different internal state bits of LFSR guessed.

equations. Values of 46 unknowns obtained in this step are substituted in all 150 equations. This in turn, reduces the overall degrees of equations. Amongst these equations, those with degree up to 7 are chosen for the next step. And other 31 unknowns are obtained by solving 37 equations of degree not more than seven. The situation given in Figure 4.4 however, does not have any equation of degree lower than 3. Our experiments do not succeed in solving equations in this case.

4.4 Algebraic Analysis of Grain-128

Grain-128, has an internal structure of 256 bits, half bits each for both LFSR and NLFSR. The goal here is to find out these 256 bits from the algebraic equations which are developed to relate internal states with output bits. As is the case with Grain-1, solving its equations to obtain all unknown

Table 4.1: Experimental results of solving equations of Grain-v1 in two steps when all bits of LFSR and 3 bits of NLFSR at different positions are guessed

Guessed positions	Number of variables found	Number of equations solved	Degrees of equations used	Time to find solution	Memory Usage
Fig 3(a)					
1st step	44	48	1	0.016 sec	5.8 MB
2nd step	none	37	7	-	Exceeds the available
Fig 3(b)					
1st step	41	48	1	0.016 sec	5.8 MB
2nd step	none	37	7	-	Exceeds the available
Fig 3(c)					
1st step	46	48	1	0.016 sec	5.8 MB
2nd step	31	37	7	0.188 sec	6.36 MB

bits is not possible. An analysis regarding the degrees of equations with different guessed bits is presented next, which will help in determining the number and position of guessed bits, to achieve the best possible results. Figure 4.5 illustrates the degrees of 200 equations of Grain-128, with all 256 unknown bits.

4.4.1 Lowering the Degrees of Equations

The output combining Boolean function of Grain-128 is of degree three. The overall degrees of output equations can be reduced, if this combining function h is replaced by some lower degree

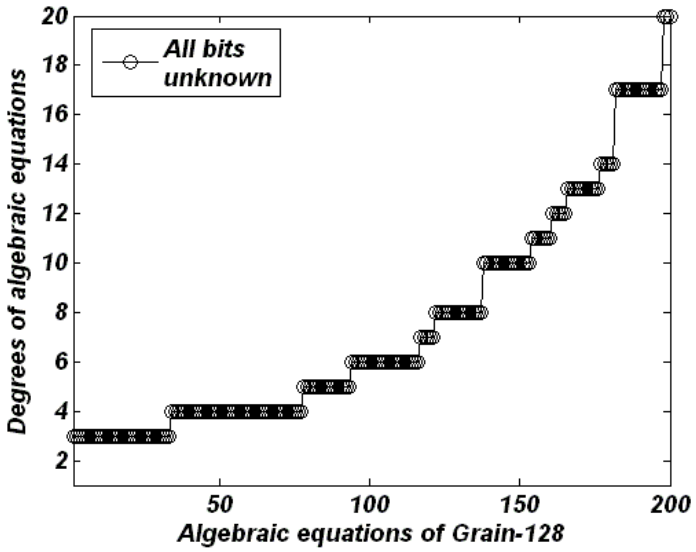

Figure 4.5: Variation in the degrees of equations of Grain-128, with all 256 unknown state bits

function. We have obtained second degree approximations of 9-variable Boolean function of Grain-128: $h(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9) = x_1x_2 + x_3x_4 + x_5x_6 + x_7x_8 + x_1x_5x_9$

Our results are summarized in Table. 4.2

Table 4.2: Second degree aproximations of Grain-128

Second Degree Approximation	Primitive Polynomial Used in Our Heuristic	Correlation Coefficient
$h_1 = x_5 + x_1x_2 + x_1x_5 + x_3x_4 + x_5x_6 + x_5x_9 + x_7x_8$	$1 + x^4 + x^5 + x^8 + x^9$	0.75
$h_2 = x_1x_2 + x_3x_4 + x_5x_6 + x_7x_8$	$1 + x^4 + x^5 + x^8 + x^9$	0.75

Two approximations of correlation coefficient of 0.75 can be seen in Table 4.2. Therefore, out of 512 output bits of Boolean function $h(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9)$ and its approximations, 448 bits are similar, whereas only 64 are different. Replacing original function with its second degree approximation, one can reduce the degrees of algebraic equations of the cipher, which

relate internal states of the cipher with output bits. A comparison of the degrees of equations with original h of degree three and those with second degree approximation is shown in Figure.4.6.

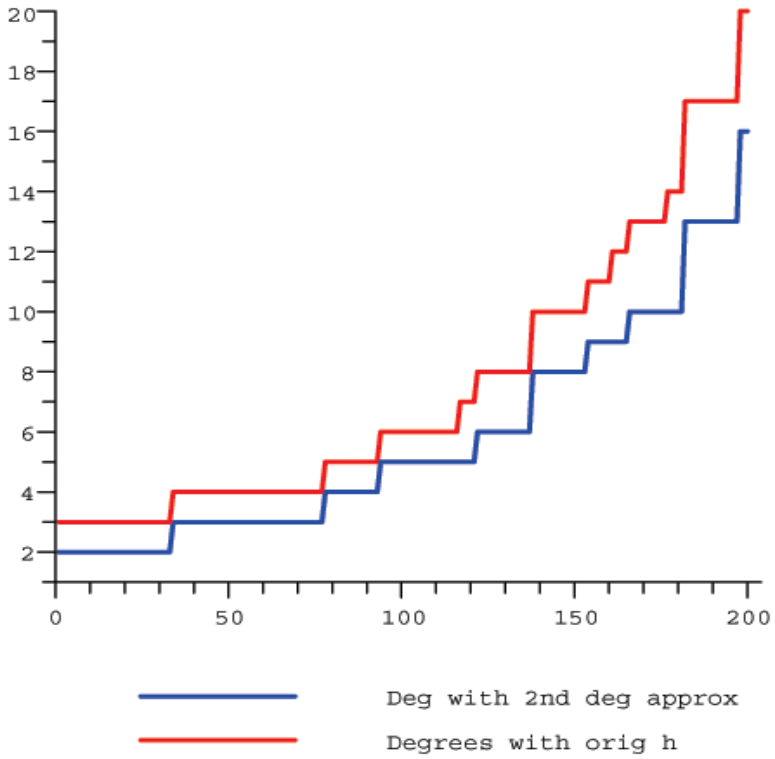

Figure 4.6: Comparison of the degrees of equations of the cipher with original h and its approximation.

An observation about these two obtained approximations is that all of the last 256 output bits of first approximation shown in Table. 4.2, whereas all first 256 output bits of the second approximation are similar to those of the original h. XOR of the actual Boolean function h and its two approximations is respectively given in Hexadecimal format as follows: $h \oplus h_1 =$

$0000AAAA0000AAAA0000AAAA0000AAAA0000AAAA0000AAAA0000$

$AAAA0000AAAA00000000000000000000000000000000000000000000$

000000000000000

$h \oplus h_2 = 00000000000000000000000000000000000000000000000000000000000000000$

$0000055550000555500005555000055550000555500005555000055550000555500005555$

It is evident that there is a symmetry in the correlation between the third degree Boolean function h of the Grain and its two second degree approximations h_1 and h_2. Due to this symmetry, lower degree algebraic equations are expected to be generated by replacing actual third degree function with second degree approximations.

XOR of the function h and h_1 has last 256 bits as zero which means that whenever bit x_1 of the input $(x_1, x_2...x_9)$ vector is one; the output of functions h and h_1 is same whereas when $x_1 = 0$ the output of h is similar to that of h_2. From the algorithm of Grain-128, we know that first bit of the input vector of function h is thirteenth bit of the NLFSR. Thus, based on the value of the thirteenth bit we can generate 2nd degree equations instead of 3rd degree due to the output function h.

To exploit second degree approximation in place of third degree output function h, we need to guess first input bit that is thirteenth bit of the NLFSR. Thus, we can generate as many equations as bits of NLFSR are guessed starting from its thirteenth bit. Since, the bits of NLFSR are guessed, the input bits x_1 and x_5 of function h are also among them. These two input bits of function h contribute in its third degree term. Therefore degrees of equations for h or second degree approximation of h are same. Therefore, the second degree approximations do not help in finding lower degree equations in case of Grain-128. It will be seen in the next paragraphs that only 64 bits of inner state of Grain-128 can be found when all the remaining bits are guessed.

It can be noticed from Figure. 4.2 and Figure. 4.5 that degrees of equations of Grain-128 do not rise as quickly as in the case of Grain-v1. But each equation of Grain-128 involves a large number of unknowns and that is why solving equations of Grain-128 is more computationally extensive. After a number of experiments with different number of guessed bits, it is concluded that not more than 64 bits can be recovered from the algebraic equations within the available resources. The

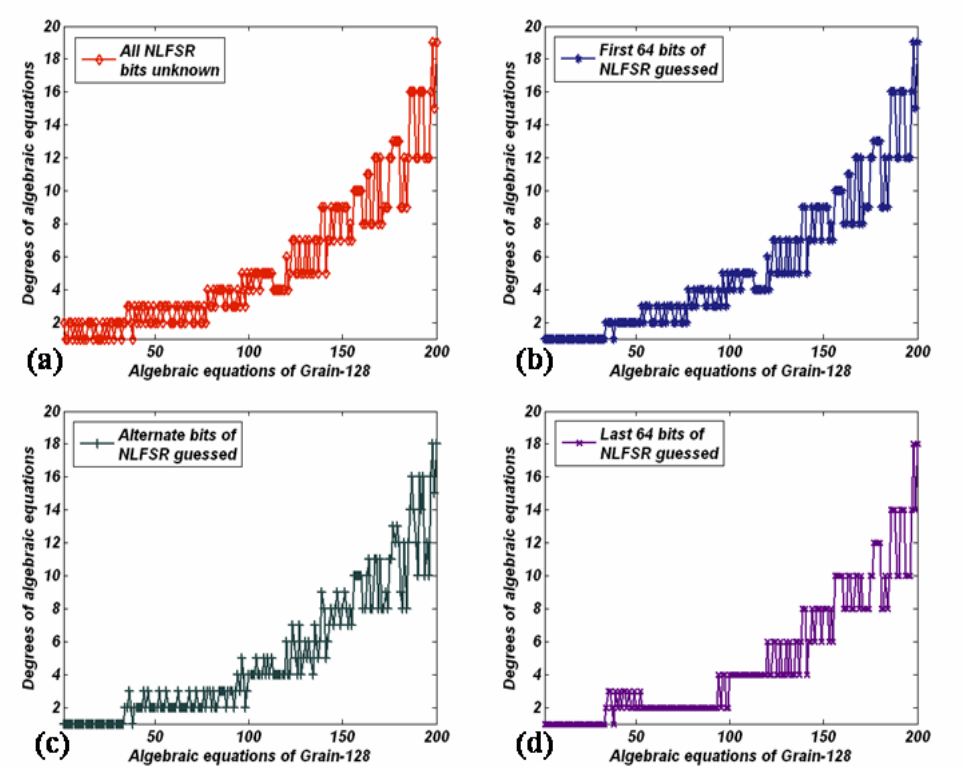

Figure 4.7: Variation in the degrees of algebraic equations of Grain-128, with all 128 bits of LFSR, and 64 different internal state bits of NLFSR guessed.

memory requirements for loading and solving the equations with variables more than the above mentioned limit exceeds the available resources. However, to decide what position of guessed bits gives best results, a comparison of the degrees of equations for different situations is presented in Figure. 4.7 and Figure.4.8.

Figure 4.7 (a) illustrates the rise in the degree of generated equations with all 128 guessed bits of LFSR while all 128 bits of NLFSR are unknown. Figure. 4.7 (b), (c) and (d) demonstrate the situations when along with 128 guessed bits of LFSR, 64 bits of NLFSR are guessed at first consecutive, alternate, and last consecutive positions respectively. Figure. 4.8 presents the otherwise situation, that is, when all bits of NLFSR are guessed, while half of the bits of LFSR are guessed at different positions.

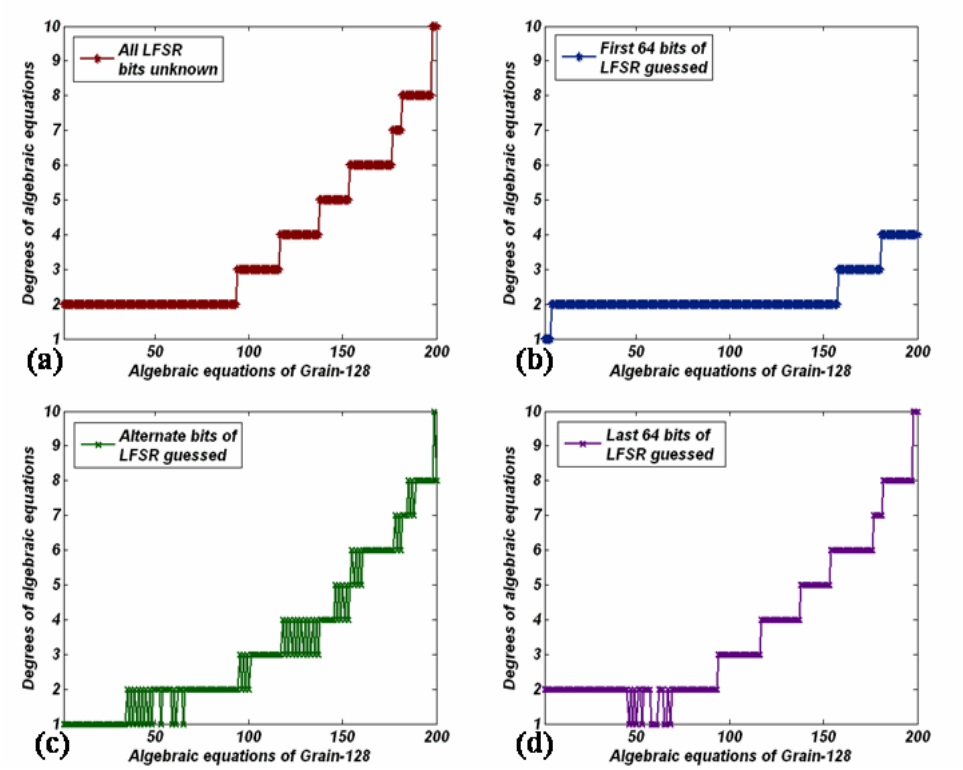

Figure 4.8: Variation in the degrees of algebraic equations of Grain-128, with all 128 bits of NLFSR, and 64 different internal state bits of LFSR guessed.

First we take into consideration the situation given in Figure 4.8, where all 128 bits of LFSR are guessed, and we have to recover 64 bits of NLFSR. Comparing the Figure. 4.8 (b), (c) and (d) it can be seen that almost same number of linear equations is obtained. With last 64 bits of NLFSR guessed, a large number of 2nd degree equations is also obtained. That is why for this case only; equations can be solved to give values of 64 unknowns. Likewise, we compare the degrees of equations shown in Figure 4.7 (b), (c) and (d), where all bits of NLFSR are guessed and the goal is to recover 64 bits of LFSR. Here, the most favorable situation is when alternate bits of LFSR are guessed, since a few linear equations are obtained. However, in case of Figure. 4.7 (b), a number of second degree equations is also obtained, therefore, a few unknown bits can be obtained by solving equations. Detail is given in Table 4.3.

Table 4.3: Experimental results of solving equations of Grain-128.

Guessed positions	Number of variables found	Number of equations solved	Time to find solution	Memory Usage
Figure.4.7(d)	60	110	5.422 sec	10.8 MB
Figure.4.7(d)	64	110	65.67 sec	25.4 MB
Figure.4.8(b)	58	150	11.87 sec	15.58 MB
Figure.4.7 (c)	64	100	0.906 sec	7.77 MB

Table 4.4: Summary of simulation results of algebraic analysis of both versions of Grain

Version	Total internal state bits	Deg of g	Deg of h	No. of bits guessed	No. of bits recovered	Time to find solution	key-bits used
Grain-v1	160	6	3	83	77	0.204 sec	150
Grain-128	256	2	3	192	64	0.906	100

4.5 Summary of the Results and Time Complexity of overall Attack

Our results of algebraic analysis on both versions of Grain are summarized in Table 4.4. Here, the best results are included for both cases.

It can be observed here, that non-linear update function of Grain-v1 is of the higher degree than Grain-128. Also, the overall degrees of system of algebraic equations of Grain-1 are much higher than those of Grain-128, still more state bits can be recovered in case of Grain-v1. This happens because output bit, in case of Grain-128, is masked with a large number of variables from NLFSR. Consequently, all linear equations include a large number of unknown bits. To find all unknowns, a large number of equations needs to be generated. However, as described in the above section,

it is not possible within the available resources. Hence, not more than 64 unknown bits can be obtained. In case of Grain-v1, although there are some equations with the very high degrees but when half of the internal state bits are guessed, enough linear equations in fewer variables can be obtained. And as a result, comparatively more variables can be recovered. Time complexity of overall attack is given in Table 4.5

Table 4.5: Time Compelxities of Solving Equations of Grain-v1 and Grain-128

Cipher	Time (in seconds)
Grain-v1	$2^{80.70}$
Grain-128	$2^{191.86}$

4.6 Conclusion

In this chapter algebraic analysis of Grain, that is a cipher with NLFSR is performed. We aimed to find the maximum possible internal state bits from the algebraic equations of the ciphers, while some bits are guessed. Algebraic equations of both versions of Grain are extensively analyzed by guessing bits at different positions to obtain best results.

Chapter 5

Algebraic Analysis of Trivium and Trivium/128

5.1 Introduction

Trivium Cannière and Preneel (2005a); Cannière (2006); Cannière and Preneel (2005b), is one of the ECRYPT stream cipher project eSTREAM (2005) candidates. It is hardware oriented, synchronous cipher which supports a key size of 80 bits and an IV size of 80 bits. It has remained unchanged since it was submitted. A few papers on its cryptanalysis can be found on the eSTREAM website Maximov and Biryukov (2007); Khazaei and Hassanzadeh (n.d.); Turan and Kara (2007); Raddum (n.d.); Babbage (n.d.). Results given in Khazaei and Hassanzadeh (n.d.) and Turan and Kara (2007) proved Trivium to be in general strong against linear sequential approximation attacks. Another approach which is algebraic in nature presented in Raddum (n.d.) establishes that Trivium can withstand the attack; however, a two-round variant of Trivium is shown to be compromised. Some other suggestions and ideas on attacking Trivium are given by Babbage in Babbage (n.d.). Maximov and Biryukov Maximov and Biryukov (2007) gave theoretical complexity for recovering internal state bits of Trivium with time complexity around $c2^{83.5}$. They have also proposed a tweaked version of Trivium which can resist their proposed attacks.

The objective of our work is to find the time complexity of actually recovering the internal state bits of the cipher and to compare it with the tweaked structure proposed in Maximov and Biryukov (2007). Trivium updates its internal states non-linearly and that is why, the degrees of algebraic equations, which relate internal states with output bits, increase with clocking. Thus, solving

equations of Trivium to obtain all the 288 unknown bits, is not possible within the practically feasible resources. However, suitable guessing of selective bits may lead to a system of equations with remaining variables, which can be solved in practical time. Resources used for simulation of equation generation and solving are same as mentioned in chapter 1.

In Maximov and Biryukov (2007) authors also discussed the issue of increasing the secret key from 80 bits to 128 bits. Instead of giving any theoretical bounds, we give experimental results of our simulations. Our experiments reveal that within the available resources, 168 bits can be recovered with 120 guessed bits of the original Trivium, whereas for tweaked version, half bits can be recovered with half bits guessed. With somewhat better resources these results can be further improved. Thus, it can be concluded that the tweaked structure is also not suitable for 128 bit key.

Bivium Raddum (n.d.) is a reduced and simplified version of Trivium which has been extensively used for analysis in literature. Since, its design philosophy is same as that of Trivium, therefore, it is easy to experiment different analysis techniques with Bivium. We have applied our guessing technique on Bivium also and results are compared with the existing results on it.

Rest of the paper is organized as follows: next section presents the brief description of the structure of Trivium and its modified structure proposed in Maximov and Biryukov (2007). In Section 5.3 and Section 5.4, we describe the analysis of algebraic equations of cipher and also results of our experiments, followed by some comments on our approach in Section 5.6. We also compare our results with other existing algebraic attacks which have been applied with different approaches. This chapter will be concluded in Section 5.7.

5.2 Brief Description of Trivium

Trivium is a simple hardware oriented synchronous stream cipher. The proposed design uses 80-bit secret key and 80-bit IV. It consists of an iterative process, which extracts the values of 15 specific state bits and uses them both to update 3 bits of the state and to compute 1 bit of the key stream. The state bits are then rotated and the process is repeated. The cipher is shown to be suitable to generate up to 2^{64} bits of key stream from a pair of key and IV.

Let the 288-bit internal state of the cipher be represented as $(s_1, s_2, ...s_{288})$, then the complete description of the cipher is given by the following simple pseudo-code:

procedure $Trivium - original(s_1, s_2, ...s_{288})$

1: **for** $t = 1$ to n **do**

2: $t_1 := s_{66} + s_{93}$

3: $t_2 := s_{162} + s_{177}$

4: $t_3 := s_{243} + s_{288}$

5: $z_t := t_1 + t_2 + t_3$

6: $t_1 := t_1 + s_{91}.s_{92} + s_{171}$

7: $t_2 := t_2 + s_{175}.s_{176} + s_{264}$

8: $t_3 := t_3 + s_{286}.s_{287} + s_{69}$

9: $(s_1, s_2...s_{93}) := (t_3, s_1, ...s_{92})$

10: $(s_{94}, s_{95}...s_{177}) := (t_1, s_{94}, ...s_{176})$

11: $(s_{178}, s_{179}...s_{288}) := (t_2, s_{178}, ...s_{287})$

12: **end for**

For key initialization, the 80 bit key and IV is directly assigned to the internal state of the cipher

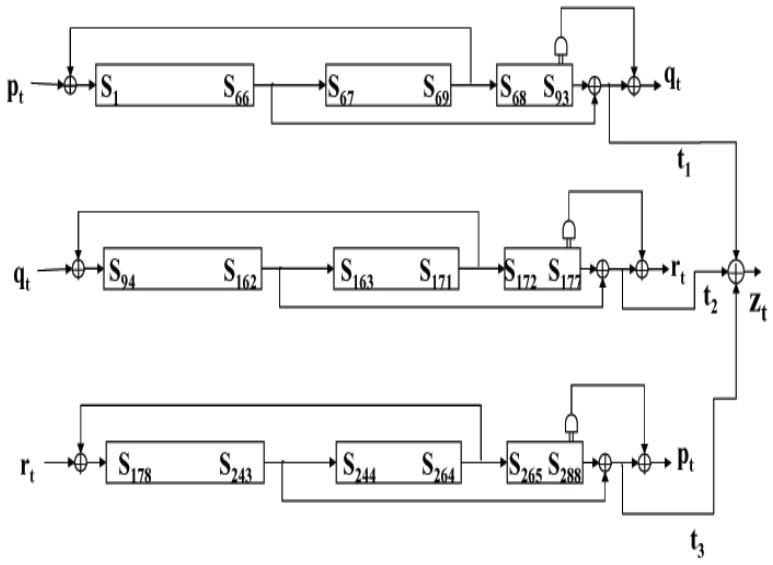

Figure 5.1: Design of the Trivium Cipher

and the remaining bits (except the last three) are set to zero. Then, the cipher is clocked 4 full cycles without producing any output. Figure 5.1 shows the original structure of Trivium. Modified version of Trivium proposed in Maximov and Biryukov (2007), has additional three AND gates, which are connected backward. With these changes, the tweaked structure can be described with the following psuedo-code:

procedure $Trivium - 128(s_1, s_2, ...s_{288})$

1: **for** $t = 1$ to n **do**

2: $t_1 := s_{65} + s_{93}$

3: $t_2 := s_{161} + s_{177}$

4: $t_3 := s_{242} + s_{288}$

5: $z_t := t_1 + t_2 + t_3$

6: $t_1 := t_1 + s_{91}.s_{92} + s_{162}.s_{164} + s_{171}$

7: $t_2 := t_2 + s_{175}.s_{176} + s_{243}.s_{245} + s_{264}$

8: $t_3 := t_3 + s_{286}.s_{287} + s_{66}.s_{68} + s_{69}$

9: $(s_1, s_2...s_{93}) := (t_3, s_1, ...s_{92})$

10: $(s_{94}, s_{95}...s_{177}) := (t_1, s_{94}, ...s_{176})$

11: $(s_{178}, s_{179}...s_{288}) := (t_2, s_{178}, ...s_{287})$

12: **end for**

Bivium, a smaller version of Trivium is composed of two registers instead of three of Trivium. The design philosophy of Bivium, however remains similar to that of Trivium. Thus with 177 internal state bits, its structure can be seen by the following psuedo-code:

procedure $Bivium(s_1, s_2, ...s_{177})$

1: **for** $t = 1$ to n **do**

2: $t_1 := s_{66} + s_{93}$

3: $t_2 := s_{162} + s_{177}$

4: $z_t := t_1 + t_2$

5: $t_1 := t_1 + s_{91}.s_{92} + s_{171}$

6: $t_2 := t_2 + s_{175}.s_{176} + s_{69}$

7: $(s_1, s_2...s_{93}) := (t_2, s_1, ...s_{92})$

8: $(s_{94}, s_{95}...s_{177}) := (t_1, s_{94}, ...s_{176})$

9: **end for**

5.3 Algebraic Analysis of Trivium

Trivium, due to non-linear update of internal states, has equations which vary with clocking. The increase in the degree of algebraic equations of Trivium is in steps. The increase in the degree of algebraic equations of Trivium, as they are generated is illustrated in Figure 5.2. Despite the fact that there are quite a reasonable number of linear and quadratic equations, still these cannot be

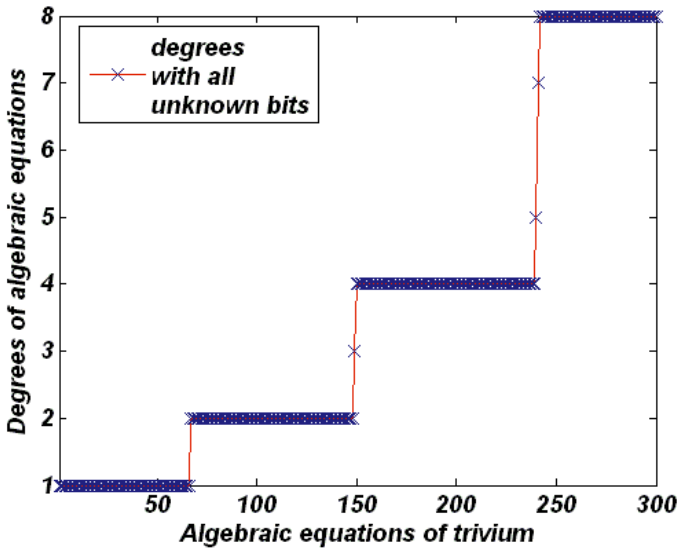

Figure 5.2: Variation in the degrees of algebraic equations of Trivium with all 288 state bits unknown

solved for all variables within the above mentioned resources. The reason being, each expression involves a large number of variables. As already discussed, we guess some of the state bits to obtain the remaining bits.

5.3.1 Results of Experiments on Original Trivium

In order to find out the bits that must be guessed to simplify our problem of solving equations, we first analyze the algebraic equations of the cipher. If the initial state bits are labeled as: $y_1, y_2, ...y_{288}$, the polynomial expressions of the generated output bits in terms of the initial state bits, during first five clocks can be seen as:

$y_{66} + y_{93} + y_{162} + y_{177} + y_{243} + y_{288}$,

$y_{65} + y_{92} + y_{161} + y_{176} + y_{242} + y_{287}$,

$y_{64} + y_{91} + y_{160} + y_{175} + y_{241} + y_{286}$,

$y_{63} + y_{90} + y_{159} + y_{174} + y_{240} + y_{285}$,

$y_{62} + y_{89} + y_{158} + y_{173} + y_{239} + y_{284}$

After 66 clocks, we obtain 2nd degree expressions, consider now a few of them:

$y_{243} + y_{288} + y_{286}.y_{287} + y_{69} + y_{27} + y_{96} + y_{111} + y_{162} + y_{177} + y_{175}.y_{176} + y_{264} + y_{222},$

$y_{242} + y_{287} + y_{285}y_{286} + y_{68} + y_{26} + y_{95} + y_{110} + y_{161} + y_{176} + y_{174}.y_{175} + y_{263} + y_{221},$

$y_{241} + y_{286} + y_{284}.y_{285} + y_{67} + y_{25} + y_{94} + y_{109} + y_{160} + y_{175} + y_{173}.y_{174} + y_{262} + y_{220},$

$y_{240} + y_{285} + y_{283}.y_{284} + y_{24} + y_{93} + y_{91}.y_{92} + y_{171} + y_{108} + y_{159} + y_{174} + y_{172}.y_{173} + y_{261} + y_{219},$

$y_{239} + y_{284} + y_{282}.y_{283} + y_{23} + y_{92} + y_{90}.y_{91} + y_{170} + y_{107} + y_{158} + y_{173} + y_{171}.y_{172} + y_{260} + y_{218}.$

The occurrence of alternate variables in above expressions reveals the importance of guessing alternate bits. If half of the state bits occurring at alternate positions are guessed, a large number of linear equations are obtained. As we have seen that 288 bits internal state of Trivium can be viewed as three registers each of length nearly 96 bits. Designers of Trivium have claimed that a guess and determine approach can be successful to determine immediately the remaining bits if overall 195 bits from three registers are guessed Cannière and Preneel (2005b). However, 144 that is half of the internal state bits can be easily recovered. Overall degrees of equations with half bits guessed is shown in Figure 5.3. Our result for solving equations with alternate guessed bits is shown in Table 5.1.

We can further improve our results if system of equations can be solved with less than half variables guessed. If first consecutive 32 bits are left as unknown bits along with the following alternate guessed bits, 160 bits can be recovered with 128 guessed bits. Figure 5.4 when first 32 consecutive bits along with next alternate bits are unknown.

Guessed bits can be further reduced to 120, if first 10 to 12 state bits from each of the three divisions of state bits $(s_1..s_{93}, s_{94}..s_{177}, s_{178}..s_{288})$ are guessed and for rest of the state bits, alternate

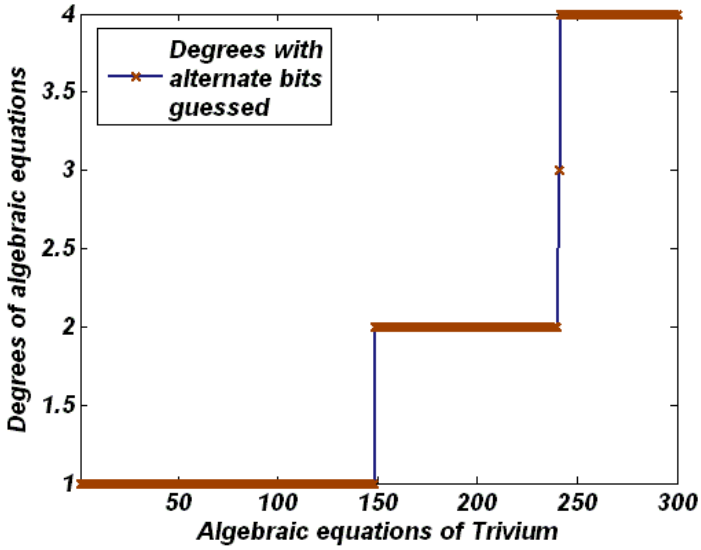

Figure 5.3: Variation in the degrees of algebraic equations of Trivium, with alternate state bits guessed.

positions are guessed. However, if number of unknown variables is further increased, the system of equations formed cannot be solved because memory requirements exceed the available. (All of our experiments are performed within the resources mentioned earlier, so with better memory and computing resources these results can be further improved). Our results of algebraic analysis of Trivium are summarized in Table 5.1. Here, the best results in terms of maximum number of state bits recovered and minimum time required are shown.

Table 5.1: Experimental results of solving equations of Trivium.

Total internal state bits	No. of bits guessed	No. of bits recovered	Time to find solution	Output-bits used
288	144(alternate)	144	0.141 sec	144
288	128	160	13 sec	250
288	120	168	1.5 min	250

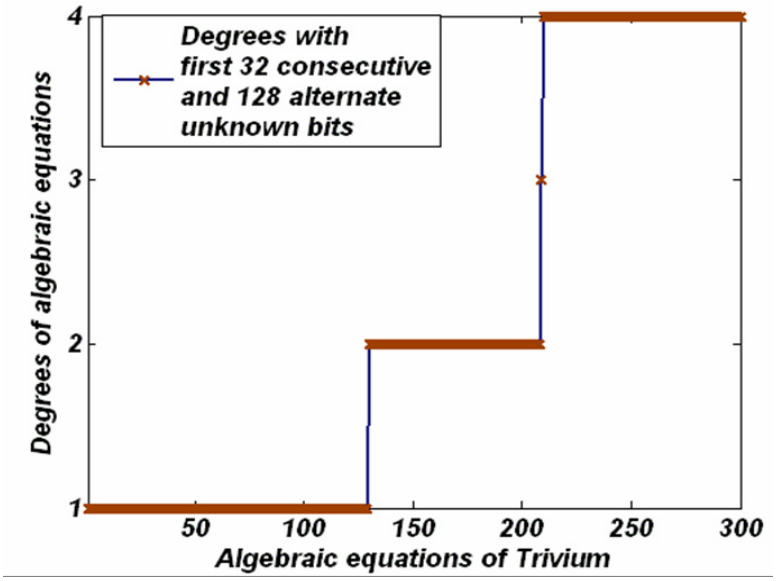

Figure 5.4: Variation in the degrees of algebraic equations of Trivium, with first 32 along with the following alternate unknown state bits respectively.

5.4 Experimental Results on Algebraic Analysis of Tweaked Trivium

Overall degrees of equations for the tweaked version are higher than the original version due to three additional AND gates. A comparison of the degrees of both versions of Trivium is presented in Figure. 5.5. Similar to previous section, if the initial state bits are assumed as: $y_1, y_2, ...y_{288}$, the polynomial expressions of the generated output bits in terms of the initial state bits, during first five clocks in the case of tweaked version can be seen as:

$y_{65} + y_{93} + y_{161} + y_{177} + y_{242} + y_{288}$,

$y_{64} + y_{92} + y_{160} + y_{176} + y_{241} + y_{287}$,

$y_{63} + y_{91} + y_{159} + y_{175} + y_{240} + y_{286}$,

$y_{62} + y_{90} + y_{158} + y_{174} + y_{239} + y_{285}$,

$y_{61} + y_{89} + y_{157} + y_{173} + y_{238} + y_{284}$

In this case also we obtain 65 linear polynomials followed by quardratic and then higher degree

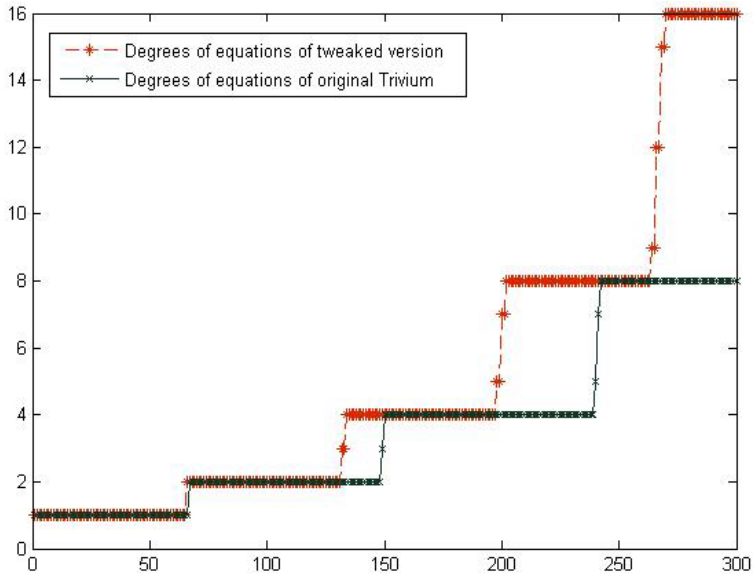

Figure 5.5: A comparison of the degrees of equations of Trivium and its tweaked version.

expressions. First five quardratic expressions are given below to examine their structure:

$y_{242}+y_{288}+y_{286}.y_{287}+y_{69}+y_{66}.y_{68}+y_{28}+y_{96}+y_{112}+y_{161}+y_{177}+y_{175}.y_{176}+y_{264}+y_{243}.y_{245}+y_{223}$,

$y_{241}+y_{287}+y_{285}.y_{286}+y_{68}+y_{65}.y_{67}+y_{27}+y_{95}+y_{111}+y_{160}+y_{176}+y_{174}.y_{175}+y_{263}+y_{242}.y_{244}+y_{222}$,

$y_{240}+y_{286}+y_{284}y_{285}+y_{67}+y_{64}y_{66}+y_{26}+y_{94}+y_{110}+y_{159}+y_{175}+y_{173}y_{174}+y_{262}+y_{241}y_{243}+y_{221}$,

$y_{239}+y_{285}+y_{283}y_{284}+y_{66}+y_{63}y_{65}+y_{25}+y_{65}+y_{93}+y_{91}.y_{92}+y_{171}+y_{162}.y_{164}+y_{109}+y_{158}+$
$y_{174}+y_{172}y_{173}+y_{261}+y_{240}y_{242}+y_{220}$,

$y_{238}+y_{284}+y_{282}.y_{283}+y_{65}+y_{62}y_{64}+y_{24}+y_{64}+y_{92}+y_{90}.y_{91}+y_{170}+y_{161}.y_{163}+y_{108}+y_{157}+$
$y_{173}+y_{171}y_{172}+y_{260}+y_{239}.y_{241}+y_{219}$

From above expressions, it is evident that guessing alternate bits can help in simplifying the equations but all second degree equations cannot be reduced to linear due to the presence of products like: $y_{66}.y_{68}, y_{243}.y_{245}, y_{162}.y_{164}$.... In this situation, system of equations cannot be solved even for 144 variables as easily as in the case of original Trivium. However, some appropriate guessing may lead to simpler equations. It can be observed from the psuedo-code of the cipher that all 288 bits can be divided into three segments namely $s_1..s_{93}, s_{94}..s_{177}, s_{178}..s_{288}$. Each of

these segments can be further divided into three sub-segments like:

$s_1..s_{65}, s_{66}..s_{69}, s_{70}..s_{93}$

$s_{94}..s_{161}, s_{162}..s_{171}, s_{172}..s_{177}$

$s_{178}..s_{242}, s_{243}..s_{264}, s_{265}..s_{288}$

The product terms of those variables which are not at alternate positions lie in the middle segment of each of the three divisions. If we guess two bits after one bit each, within these segments and for the remaining segments alternate bits are guessed, we will obtain some more linear equations. The number of linear equations is not decreased if first 5 to 6 states of each of the three main divisions is kept as unknown variables. In this way, system of equations obtained by guessing half of the bits is solvable within given resources. Figure 5.6 shows a comparison of the degrees of equations with all unknown state bits and with half state bits guessed but selected as mentioned above. Our experimental results on tweaked version of Trivium are summarized in Table 5.2

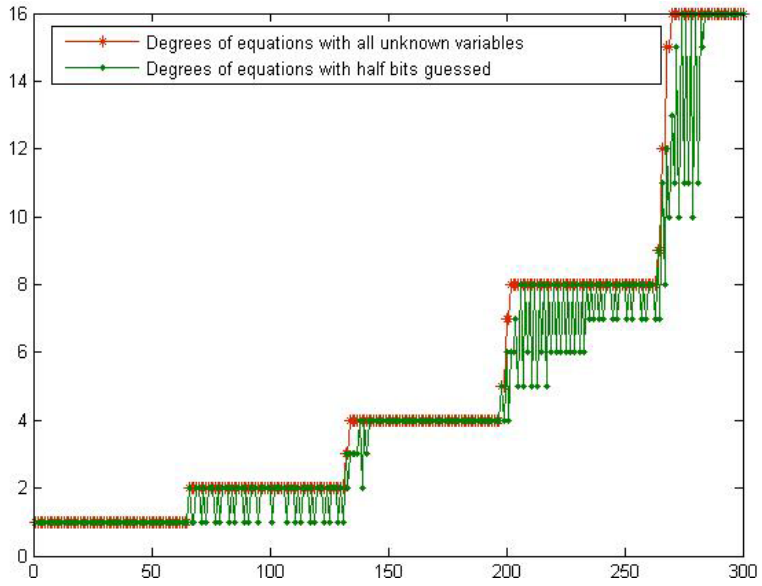

Figure 5.6: Variation in the degrees of algebraic equations of Trivium, with alternate state bits guessed.

Table 5.2: Experimental results of solving equations of tweaked Trivium

Total internal state bits	No. of bits guessed	No. of bits recovered	Time to find solution	Output-bits used
288	144	144	4.3 sec	200

Our approach of finding the maximum number of bits that can be recovered from the algebraic equations of Trivium and its tweaked version, aims at investigating the resistance that the cipher with non-linear update offers against algebraic cryptanalysis. Since, Trivium is a live candidate of eSTREAM project, our experimental work is significant. These results show that we can recover 168 internal state bits of Trivium, with 120 guessed bits and for its tweaked version, half of the bits can be obtained while remaining half are guessed in a few seconds. It should also be noted that to mount an actual attack with this approach, it is required that for each guess a system of equations be solved. However, it is important that for a wrong guess, equations become inconsistent and Magma decides inconsistency in a fraction of second by giving Groebner basis of the system as 1. Our experiments are performed in limited resources, and it can be concluded that with better resources these results can be further improved.

Trivium actually offers a security level of 80 bits, but as already mentioned in Maximov and Biryukov (2007), on account of initialization, the complexity of exhaustive search of 80 bit key will be more than 2^{80}. Thus, our approach can recover internal state bits of Trivium with time complexity comparable to exhaustive search, if not better. Although, tweaked version offers somewhat complex equations and also with our resources, we do not succeed to recover more than half state bits, still the structure is not suitable to offer 128 bit security.

5.5 Results on Bivium

Using the same guessing technique, we have obtained results on Bivium too. We have found that within the available resources, we can recover 133 bits out of 177 while 44 secret internal state bits are guessed. In this case also, like Trivium, we have guessed alternate bits from 46 to 93 and then from position 139 to 177. While bits from 1 to 45 and 94 to 138 are all unknown. Experimental results on Bivium are summarized in Table 5.3

Table 5.3: Experimental results of solving equations of Bivium

Total internal state bits	No. of bits guessed	No. of bits recovered	Time to find solution	Output-bits used
177	49	128	1.59 sec	200
177	44	133	236.5 sec	200

5.6 Overall Time Complexity of Attack on Bivium and Trivium and Comparison with Existing Attacks

We can make a comparison of time complexity of solving equations of both Trivium and Bivium with some previous results. Best result on Trivium with our guessing approach is to guess 120 bits to obtain remaining bits in 1.5 min that is 90 seconds ($\approx 2^{6.5}$). Since, 120 bits have to be guessed, this means that we will have to repeat the experiment for 2^{120} times as a worst case scenario before we get the right solution. As already mentioned for a wrong guess we will obtain inconsistent system of equations and Groebner basis for such a system would be one. Most of the time, Magma gives solution for inconsistent system of equations in less time than the consistent system. However, to avoid any confusion we take here the time taken for solving equations with right guessed bits. Results of our experiments are compared with another solving technique using

SAT solvers given in McDonald et al. (2009) and are shown in Table 5.4

Table 5.4: A Comparison of Time Compelxities of Solving Equations of Trivium and Bivium.

Cipher	Time (in seconds)			
	MiniSat	RSat	PicoSati	Magma(F_4) with our approach of guessing bits
Trivium	$2^{159.9}$	$2^{161.9}$	$2^{161.6}$	$2^{126.5}$
Bivium	$2^{42.7}$	$2^{46.6}$	$2^{45.9}$	$2^{49.66}$

Although time complexity of the proposed attack is far from the practical threat still it can be seen that our approach gives better results on Trvium than the existing practical attacks McDonald et al. (2009).

5.7 Conclusion

Trivium is a simple hardware oriented candidate cipher of eSTREAM project. We have analyzed and compared the original design of Trivium with a proposed tweaked version. Our approach is experimental, and we aimed at finding the maximum possible internal state bits, while others are guessed. The results show that tweaked version is also not suitable to offer 128 bit security.

Chapter 6

Resitance of Ciphers like Mickey for Algebraic Attack

6.1 Introduction

We have seen that structures like A5/1 with mutually clocked registers can resist algebraic cryptanalysis in Chapter 3. In this chapter, we will discuss the structure of Mickey Babbage and Dodd (n.d.b,n) a candidate cipher of eSTREAM project which has been included in its final portfolio. Internal registers of Mickey are updated non-linearly as in the case of Grain and Trivium. Our approach of solving equations with some guessed bits shows Mickey to be most resistant against this approach.

Rest of the chapter is organized as follows: Section 6.2 gives a brief description of the structure of Mickey and Section 6.3 presents analysis of Mickey. In Section 6.4 some comments and discussion are given and Section 6.5 will conclude the chapter.

6.2 Structure of Mickey

Two versions of Mickey have been proposed Babbage and Dodd (n.d.b,n). Mickey offers a security level of 80 bits with internal states of 160 bits, whereas, Mickey-128 has an internal state of 256 bits and security level of 128 bits. The design strategy of both versions is same and follows the irregular clocking of registers C.J.A.Jansen (2004). The key generation mechanism of the cipher has two registers both of length 80. Bits $r_i(0 \leq i \leq r_{79})$ of LFSR R are clocked depending upon a control bit. Another register S also has an internal state of 80 bits labelled as

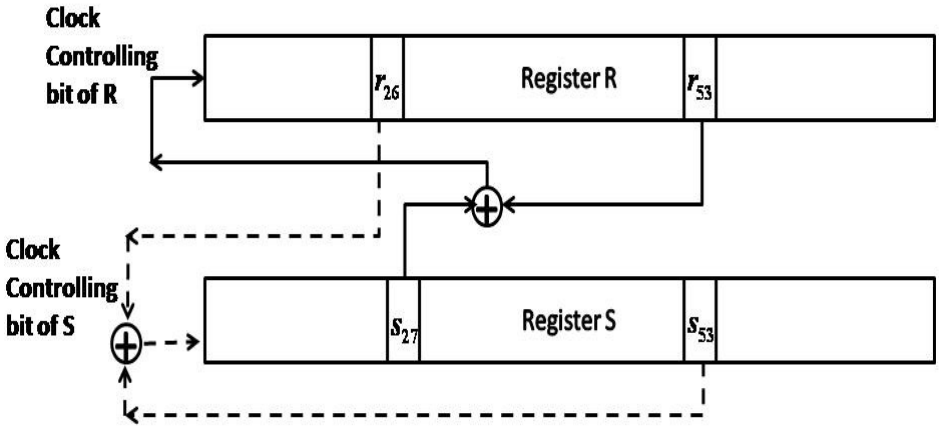

Figure 6.1: The Mutual Clocking of Mickey.

$s_i(0 \leq i \leq s_{79})$. This register is non-linear and its clocking is also made in two ways based on another control bit. We do not give here the detailed structure of Mickey, that can be found from their design specifications Babbage and Dodd (n.d.b,n). We, however, use the same terminology and names etc. of the registers as are used originally. The main design strategy of the cipher is of irregular clocking which is achieved by mutual interdependence of registers. That is clock control bits are derived from both registers. Interdependent clocking of Mickey can be seen from Figure 6.1.

6.3 Analysis of Mickey

Generation of algebraic equations, which can relate internal state bits of the Mickey with the output bits, is not straightforward as in Grain and Trivium.

To generate algebraic equations of Mickey cipher which can relate internal state bits with the output bits, we have to turn the parts of the algorithm, which involve conditional logic, to the expression without it. For instance, the register "R" is clocked according as:

If $CONTROL - BIT - R = 1$

$For 0 \leq i \leq 79$

$\acute{r_i} = \acute{r_i} \oplus r_i$

For generation of equations above logic must be modified as follows:

For $0 \leq i \leq 79$

$\acute{r_i} = \acute{r_i} \oplus r_i * CONTROL - BIT - R$

And similarly the conditional part of the algorithm for clocking register "S" can be seen as follows: Register "S" has the following code for clocking:

If $CONTROL - BIT - S = 0$

$For 0 \leq i \leq 79$

$\acute{s_i} = \hat{s_i} \oplus (FB0_i.FEEDBACK - BIT)$

If instead $CONTROL - BIT - S = 1$

For $0 \leq i \leq 79$

$\acute{s_i} = \hat{s_i} \oplus (FB1_i.FEEDBACK - BIT)$

This can be viewed as below:

For $0 \leq i \leq 79$

$\acute{s_i} = (1 \oplus CONTROL - BIT - S).(\hat{s_i} \oplus (FB0_i.FEEDBACK - BIT) \oplus (CONTROL - BIT - S).(\hat{s_i} \oplus (FB1_i.FEEDBACK - BIT)$

Internal states are updated non-linearly in this case also. However, due to irregular clocking the rise in the degree of equations is much higher than Grain and Trivium. Our simulations for Mickey resulted in the generation of equations with very high degree. For example, rise in the degrees of first five equations is given as: 1, 2, 4, 7, 11.... So that even generation of more equations becomes

quite difficult. Attempts to obtain some low order equations by guessing even a large number bits also do not succeed. It is notable here that a number of experiments are performed in which state bits are guessed at different locations based on the structure of equations. However, due to interdependence of registers, these attempts do not succeed. It is thus concluded that design of Mickey has very tight security margin against any allowance for state bits to be recovered using an algebraic approach.

6.4 Some Comments

As evident from the previous chapters, structures of Grain-1 and Trivium are easier for internal state recovery algebraically as compared to Grain-128 and Mickey. Non-linear update of registers is a way to defend against algebraic cryptanalysis but structure must also be strong enough so that guessing some of the bits may not result in a large number of linear and low degree equations, as happens with Trivium.

Mickey uses the concept of irregularly clocking of registers. Although algebraic attack against clock controlled stream ciphers has been studied with focus on ciphers, in which one generator controls the clocking of other generators Al-Hinai et al. (2006). Whereas, ciphers that involve the clocking in which none of the registers are regularly clocked, are quite resistant as is the case in A5/1. Mickey also combines irregular clocking and interdependence of registers for clocking and as a result, its algebraic equations gain very fast increase in degree. Also the form of equations does not give any help in selectively guessing some of the bits to find others. Both of its registers are mutually dependent and guessing one or the other or in any other form does not at all help in obtaining linear or low order equations to make the solution possible

6.5 Conclusion

Design of a stream cipher must ensure that it can resist all known attacks. In this context, identifying design of the ciphers which can resist any particular strategy for their analysis also has importance. This chapter aimed at doing algebraic analysis of the Mickey that has been selected for the final portfolio of eSTREAM project. It has been found, that due to the clocking with interdependent registers, our approach of finding secret state bits by solving algebraic equations with guess and determine, does not work. Mickey, is thus, resistant to internal state recovery attack.

Chapter 7

Improving the Resistance of Grain-V1 against Algebraic Attack

7.1 Introduction

In chapter 4, we have seen algebraic analysis of Grain-v1. It has been experimentally verified that half bits of Grain-v1 can be recovered if remaining half are guessed by solving algebraic equations in a fraction of a second within limited resources. Although, this attack is not better than exhaustive search, still it gives a method to attack the cipher other than brute force attack. It can also be argued that solving equations to obtain half of the internal state bits is better than brute force attack because for that we will have to include the complexity of initialization process. Moreover, results can be further improved with better resources. This attack is possible because degrees of the equations formed due to the internal mechanics of the cipher is greatly reduced when some selective bits are guessed.

Grain-v1 has already been revised and Grain-128 Hell et al. (n.d.) has been proposed that supports a key size of 128 bits. In this chapter, we focus on the improvement of Grain-v1 without increasing its internal state or key size. This work aims at giving an improved version by analyzing the algebraic equations of Grain-v1, so that it may give better resistance against above mentioned algebraic attack. We propose a small modification in the structure of Grain-v1 that can give a defense against algebraic analysis without increasing the number of its internal state bits. Our proposed modification improves upon the algebraic structure of the cipher and it intend to avoid recovery of internal state bits due to solving algebraic equations with guess and determine

approach. Moreover, this modification does not affect the overall complexity of enciphering or deciphering process because we do not add any additional components.

Rest of the chapter has following organization: Next Section gives the proposed modified version of Grain and also gives an analysis and comparison of the proposed and original version in the light of algebraic attack using guess and determine approach. The chapter is concluded in Section 7.3.

7.2 Proposed Modifications In Grain-v1

Based on the analysis given in previous chapters, it can be seen that if input of h includes more than one bits from NLFSR, we will not obtain linear equation, even in the initial output bits, in terms of variables from NLFSR on guessing even all bits of LFSR. Our experiments show that adding one more variable is enough in this case. With this small modification, output function can be viewed as:

$keybit_k := x_1 + h(x_{64}, x_{10}, y_{47}, y_{26}, y_4)$

Here, y_{64} is replaced by x_{10} as second input variable of Boolean function h. Thus, the influence of NLFSR in output bit is increased. The second variable from the NLFSR is taken from the early states, so that non-linear update of NLFSR can play its effective role in the initial equations. The equations obtained with this modified output function will have much higher degrees even with all bits of LFSR guessed. It has been experimentally verified that system of equations, thus obtained cannot be solved within the same resources as has been used in solving the equations of Grain-v1.

Above mentioned modification does not affect the overall degrees of the equations of Grain-v1 when all variables are unknown. Figure 7.1 illustrates this fact by giving a comparison of the degrees of algebraic equations of the original and proposed version.

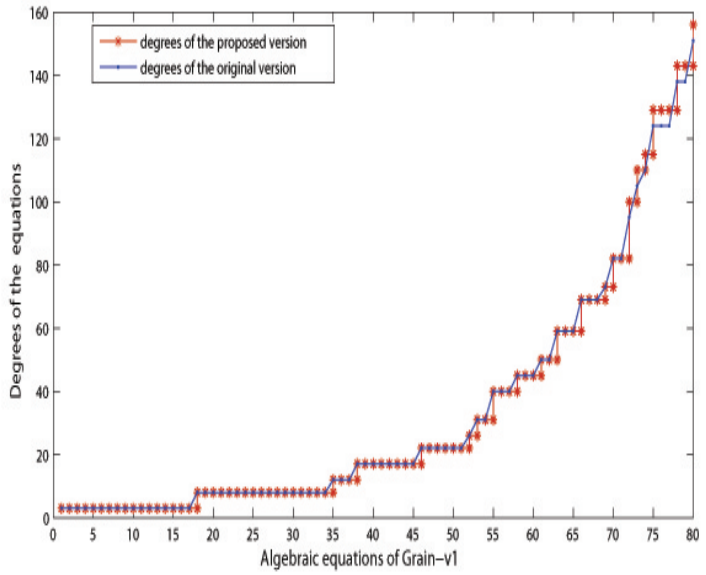

Figure 7.1: A Comparison of the degrees of Grain-v1 for proposed and original version with all unknown variables.

However, this modification will have a remarkable effect when bits from LFSR are being guessed to obtain bits from NLFSR. Justification for this increase in the degree is due to the function h. Since ,

$$h(x_{64}, x_{10}, y_{47}, y_{26}, y_4) = y_{26} + x_{64} + y_4.x_{10} + y_{47}.x_{10} + x_{10}.x_{64} + y_4.y_{26}.y_{47} + y_4.y_{47}.x_{10} + y_4.y_{47}.x_{64} + y_{26}.y_{47}.x_{64} + y_{47}.x_{10}.x_{64}$$

Above mentioned form of the function results in the initial algebraic equations to be of second degree due to the presence of the term $x_{10}.x_{64}$. Since in the original function h the term x_{64} is multiplied with a variable from LFSR and when bits from LFSR are guessed, this product will lead to a linear term. However, in the modified h the term $y_{47}.x_{10}.x_{64}$ also exists, therefore, whenever y_{47} is 1, equation becomes linear. However, since for different initial states, the term y_{47} can be 1 or 0 with equal probability, therefore half of the equations will be quadratic in comparison to all linear equations in the case of original version. After a few clocks, degrees of the equations rise in both cases due to the nonlinear update states of NLFSR.

It should also be noted that replacing any of the four input variables of the NLFSR in original h with one bit from LFSR will create the same result. To demonstrate the concept described above, a comparison of the degrees of the equations for the original and proposed version is presented in Figure 7.2. Here, the guessed bits of NLFSR are as follows:

$[0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0,$

$0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0,$

$0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0,$

$0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1]$

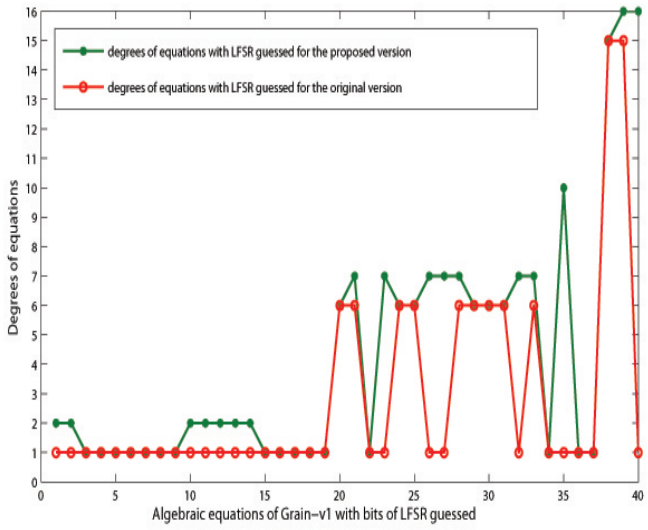

Figure 7.2: A comparison of the degrees of original and proposed version when bits of LFSR are guessed.

It is also important to see the impact of this modification on the algebraic equations when bits of LFSR are guessed instead of NLFSR. Figure 7.3 presents the degrees of the algebraic equations of Grain-v1 with guessed bits of LFSR for both: the original and proposed versions of Grain-v1. It can be seen clearly that degrees of the equations for both versions is same. It is known from chapter 4 that with LFSR bits guessed, equations thus formed cannot be solved. Thus, with this

modified structure also, system of equations cannot be solved within the available resources.

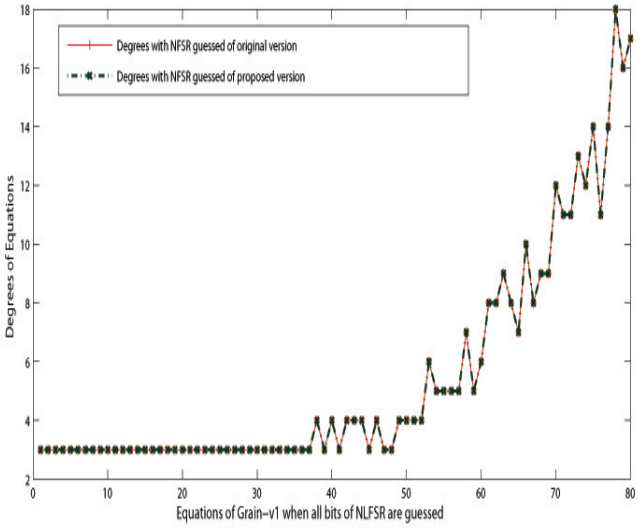

Figure 7.3: A comparison of the degrees of original and proposed version when bits of NLFSR are guessed.

The proposed version replaces one input bit from LFSR of function h with a bit from NLFSR to obtain higher degree equations for the situation when all bits from LFSR are guessed to obtain all unknown bits of NLFSR. This substitution is made to provide a defense against an algebraic attack which is successful in retrieving all bits of NLFSR when all bits of LFSR are guessed.

Now we will consider the case that if we replace two input bits of h, so that three input bits of h are from NLFSR. The equations so obtained are of even higher degree and none of the equations constructed after guessing LFSR will be linear. But in that case, bits from LFSR can be retrieved by guessing bits from NLFSR. This can be verified in Figure 7.4, where a comparison between the degrees of equations of original version of Grain-v1 is given with bits from NLFSR guessed with modified version when two input bits of the function h are substituted. Because of many linear equations, it has been experimentally verified that this system of equations can be easily solved.

Therefore, it can be concluded that substitution of two input bits of function h will increase the degrees of equations with one guessing situation but will decrease for the other guessing strategy. Thus, substitution of two bits is not suitable. However, substitution of one bit will increase the degrees of equations in the case where LFSR is guessed in comparison with the equations formed for the original version of Grain-v1. And this substitution also does not decrease the degree of equations and thus complexity of solving equations for the situation where NLFSR is guessed to obtain bits from LFSR.

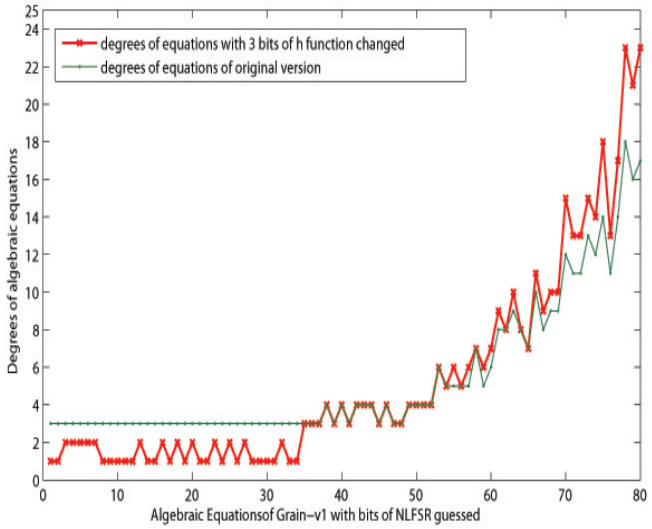

Figure 7.4: A comparison of the degrees of original and a modified version where two input bits of function h are substituted and bits of NLFSR are guessed.

A single bit alteration in the input variables of the output function can successfully help in improving the security level of Grain-v1 against recovery of internal state bits by solving algebraic equations. It is, however, important to see the effect of this modification on other attacks on the cipher also. In general, a small change in the cipher may increase its resistance against one while decrease against some other cryptanalytic technique. In case of Grain-v1, a well known attack is correlation attack Berbain et al. (2006). However, it has also been recommended in the

same paper that by increasing the influence of NLFSR in output relation correlation attack can be avoided. Therefore, it is expected that the proposed modification will show better results against correlation attack also.

7.3 Conclusion

A small modification by changing the input variable of the output function in the design of Grain-v1 is proposed to improve the resistance of Grain-v1 against algebraic attack with guess and determine approach. It has been verified that the proposed design shows better resistance against the guessing technique which otherwise works for the original version. The proposed version has also been analyzed and compared with the original design of Grain-v1.

Chapter 8

Modifications in the Design of Trivium to Increase its Security Level

8.1 Introduction

Trivium has an internal state of 288 bits but provides a security level of only 80 bits, an obvious desire is to raise its security level to at least 128 bits without increasing its internal state bits. The study and analysis of Trivium given in chapter 5 has proved that mainly the algebraic structure of the cipher has the propensity that a guess and determine type of attack, to recover the internal state bits of the cipher by solving algebraic equations, can be mounted. In this chapter, we propose some modifications in the structure of Trivium so that its elegant structure can be used to give a larger security margin. The modified versions show better resistance to the internal state recovery attack. Trivium/128 Maximov and Biryukov (2007) uses one additional AND gate in each register to improve its security margin. We, however, present two simple modifications without any additional AND gates and show that our proposed modifications can provide better security level than actual Trivium. Other two modifications with additional AND gates, are also proposed but these can provide remarkably better resistance than Trivium/128, and thus can give a larger security margin with the same number of internal state bits. We compare our proposed structures with the original Trivium and Trivium/128.

The analysis presented in chapter 5, shows that the possibility of recovering state bits can be reduced if we can have lesser number of linear equations in the start and also the variables in the product terms are distributed in a way that degrees of equations do not decline on guessing

some of the selective bits. The proposed modifications have this inspiration. We, therefore, give a comparison of the degrees of equations, which are formed from our proposed modifications, with degrees of original Trivium and Trivium/128. In general, the degrees of equations cannot be taken as the only criteria for deciding about a system difficult enough or not to be solved. In this case, as the original structure let too many bits to be recovered owing to the presence of many lower-degree equations, therefore, keeping the main design strategy intact, we compare our modified designs with the original design based on these criteria.

Rest of the chapter is organized as follows: Next two sections present the proposed modifications of Trivium with and without additional AND gates respectively. A detailed comparative analysis of the proposed modifications is given in succeeding section. The article is concluded in the last Section.

8.2 Modifications of Trivium without Additional Product Terms

In this section, we propose two modified designs of Trivium. Our focus is to increase the degrees of algebraic equations to be higher than the original version without adding any product terms so that the efficiency of the original structure is not declined.

8.2.1 First Modification of Trivium Trivium-A:

Our first proposal for modification is simple, and the idea is to reduce the number of linear equations that emerges during the earlier clocks of the cipher. The system of equations thus formed has a higher degree than the original Trivium. We have seen in the preceding sections that the structure of algebraic equations of the original cipher has the allowance that if some bits are guessed at some selective positions, many linear and other lower degree equations can be obtained. And as a result remaining secret bits can be recovered in a few seconds within the limited resources.

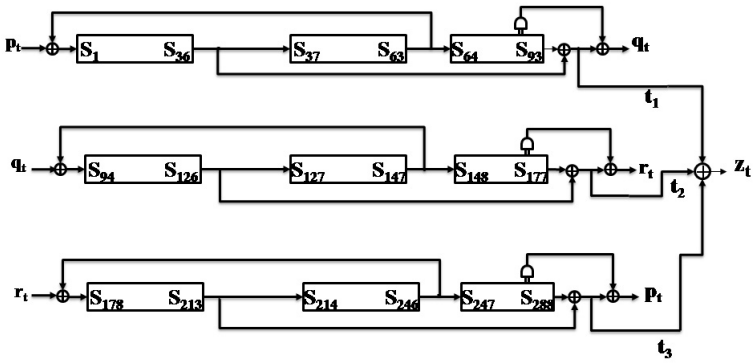

Figure 8.1: First Modified Version Trivium-A.

The proposed modification tries to avoid this easy recovery of bits even when half of the bits are guessed. This may be achieved if feedback of the product term appears in the output bits earlier. Consider the possible modification as given in Figure 8.1, keeping the above-mentioned criterion in mind. The following pseudo-code represents the above mentioned modified version:

procedure $Trivium - A(s_1, s_2, ...s_{288})$

1: **for** $t = 1$ to n **do**

2: $t_1 := s_{36} + s_{93}$

3: $t_2 := s_{126} + s_{177}$

4: $t_3 := s_{213} + s_{288}$

5: $z_t := t_1 + t_2 + t_3$

6: $t_1 := t_1 + s_{91}.s_{92} + s_{147}$

7: $t_2 := t_2 + s_{175}.s_{176} + s_{246}$

8: $t_3 := t_3 + s_{286}.s_{287} + s_{63}$

9: $(s_1, s_2...s_{93}) := (t_3, s_1, ...s_{92})$

10: $(s_{94}, s_{95}...s_{177}) := (t_1, s_{94}, ...s_{176})$

11: $(s_{178}, s_{179}...s_{288}) := (t_2, s_{178}, ...s_{287})$

12: **end for**

In this case, we obtain 33 linear equations compared with 66 of the original structure and 65 of Trivium/128 Maximov and Biryukov (2007). As already pointed out that in case of Trivium-A, the feedback of product terms appear in output earlier than the original version of Trivium. In both cases s_1, s_{94} and s_{178} are replaced by t_3, t_1, and t_2 respectively. In case of Trivium and Trivium/128, the distance between the taps and the outcome pins of the gates is minimum 66 and 65 respectively and that is why first second degree term appears after 65 clocks with update of s_1, s_{94} or s_{178}. Whereas, Trivium-A is designed with the view that difference between the taps and outcome pins is maintained to be not less than 32.

This modification does not involve any additional AND gate; just a change in the placement of the feedback function can help in reducing the number of linear equations. A comparison between the degrees of algebraic equations of Trivium-A with those of the original Trivium and Trivium/128 is presented in Figure 8.2. It can be seen that degrees of Trivium-A are higher than both those of the original Trivium and Trivium/128. A thorough analysis of the proposed modification Trivium-A and its comparison with Trivium and Trivium/128 is presented in Section 8.4.

8.2.2 Second Modification of Trivium-Trivium-B

In this section, we introduce another possible modification in the structure of Trivium. Here also no additional AND gate is used; rather the product of variables is placed due to the mutual dependence introduced within the internal state bits. Due to this, second-degree feedback is generated, thus the product of variables in steps 6, 7, and 8 of the procedure of the original Trivium

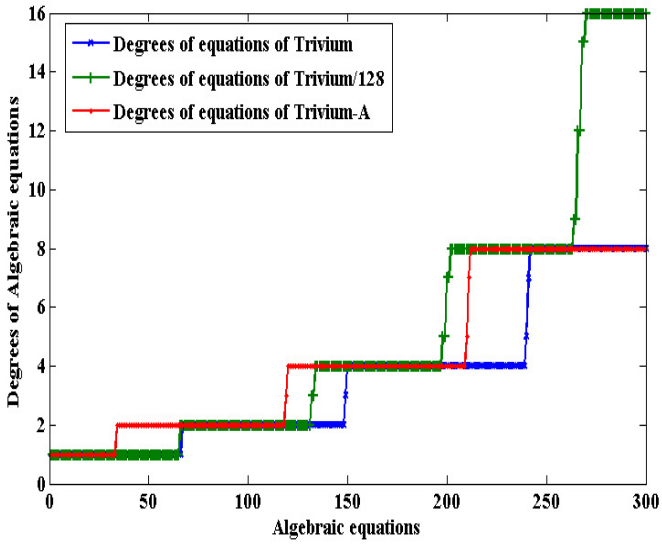

Figure 8.2: A Comparison of the degrees of Algebraic equations of Trivium-A with the original version of Trivium and Trivium/128.

is replaced.

In stream ciphers, mutual dependence of registers in clocking can resist algebraic cryptanalysis due to the quick rise in the degree of equations Afzal and Masood (2007, 2008d); Al-Hiana et al. (2007). The structure of Trivium, as shown in Figure 5.1, also has three registers. We propose that if updating of the feedback variables of the three registers becomes mutually dependent, the degrees of equations rise more rapidly. However, it will naturally result in another product of variables or an additional AND gate in implementation of the cipher. Therefore, to maintain the number of AND gates as in the original structure, we eliminate the product terms at subsequent steps. Given below is the procedure for updating the variables t_1, t_2 and t_3 based on mutual dependence of the three portions of the state bits:

1: **if** $s_{162} = 1$ **then**

2: $\quad t_1 := t_1 + s_{66} + s_{147}$

3: **else**

4: $t_1 := t_1 + s_{147}$

5: **end if**

6: **if** $s_{243} = 1$ **then**

7: $t_2 := t_2 + s_{162} + s_{246}$

8: **else**

9: $t_2 := t_2 + s_{246}$

10: **end if**

11: **if** $s_{66} = 1$ **then**

12: $t_3 := t_3 + s_{243} + s_{63}$

13: **else**

14: $t_3 := t_3 + s_{63}$

15: **end if**

This change in the updated variables results in the following complete procedure of the proposed modified version Trivium-B:

procedure $Trivium - B(s_1, s_2, ...s_{288})$

1: **for** $t = 1$ to n **do**

2: $t_1 := s_{36} + s_{93}$

3: $t_2 := s_{126} + s_{177}$

4: $t_3 := s_{213} + s_{288}$

5: $z_t := t_1 + t_2 + t_3$

6: $t_1 := t_1 + s_{66}.s_{162} + s_{147}$

7: $t_2 := t_2 + s_{162}.s_{243} + s_{246}$

8: $t_3 := t_3 + s_{243}.s_{66} + s_{63}$

9: $(s_1, s_2 ... s_{93}) := (t_3, s_1, ... s_{92})$

10: $(s_{94}, s_{95} ... s_{177}) := (t_1, s_{94}, ... s_{176})$

11: $(s_{178}, s_{179} ... s_{288}) := (t_2, s_{178}, ... s_{287})$

12: **end for**

With the modification Trivium-B we obtain first 33 linear equations similar to Trivium-A, as compared to the 66 linear equations from the original Trivium. Afterwards, degrees are increased step wise. A comparison between the degrees of algebraic equations of the original Trivium with the Trivium/128 and Trivium-B is presented in Figure 8.3. Compared with the original

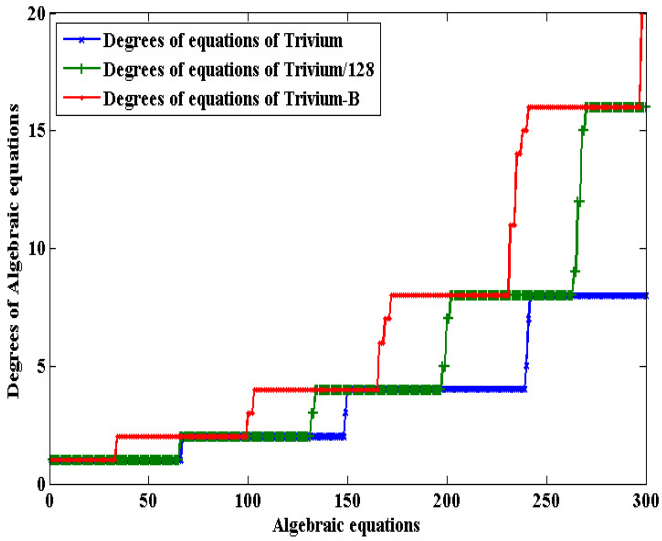

Figure 8.3: A Comparison of the degrees of Algebraic equations of original version of Trivium with the proposed modified version Trivium-B.

version, Trivium-B achieves considerably high degrees. Moreover, in case of Trivium-B, alternate guessing of bits is not helpful because instead of product terms of alternate bits here, bits from different registers are combined with AND gate. It is clear from Figure 8.3, that the proposed version has degrees higher than even Trivium/128, whereas Trivium/128 also has an additional AND gate to improve the security level to 128 bits. A detailed analysis is given in Section 8.4.

8.3 Proposed Modifications with Additional Product Terms

In this section, we propose some other modified designs of the Trivium again without altering its main design philosophy. Here, our approach is to increase the degrees of algebraic equations much higher without disturbing the original design of the cipher. We aim to achieve a form of equations such that with guessing some of the bits, one cannot turn the degrees of equations lower enough to make the system solvable. These modifications with additional AND terms aim to provide security level of 128 bits and we have shown experimentally that our modifications provide better resistance to the algebraic attack with guess and determine approach as compared to Trivium/128.

8.3.1 Third Modification of Trivium- Trivium-C

This proposed design uses the same idea as in Trivium-B. Here, we introduce the same product terms resulting from the interdependent updating of three portions of the internal state of the cipher. Unlike Trivium-B, here we do not remove the product of alternate bits in the original design of Trivium. Thus, the AND gates introduced in Trivium-B are additionally included within the design. Moreover, to eliminate the linear equations, the non-linear terms are included in the output from start. We compare this design for degrees of equations with not only the original Trivium, from which it must offer higher-degree equations but also with Trivium/128. The comparison with Trivium/128 shows that our proposed version gives much higher degrees of the system of equations, although the proposed version also uses three additional AND gates as in Trivium/128. Proposed design of Trivium-C is illustrated with the following pseudo-code:

procedure $Trivium - C(s_1, s_2, ...s_{288})$

1: **for** $t = 1$ to n **do**

2: $t_1 := s_{66}.s_{162} + s_{36} + s_{93}$

3: $t_2 := s_{162}.s_{243} + s_{126} + s_{177}$

4: $t_3 := s_{243}.s_{66} + s_{213} + s_{288}$

5: $z_t := t_1 + t_2 + t_3$

6: $t_1 := t_1 + s_{91}.s_{92} + s_{147}$

7: $t_2 := t_2 + s_{175}.s_{176} + s_{246}$

8: $t_3 := t_3 + s_{286}.s_{287} + s_{63}$

9: $(s_1, s_2...s_{93}) := (t_3, s_1, ...s_{92})$

10: $(s_{94}, s_{95}...s_{177}) := (t_1, s_{94}, ...s_{176})$

11: $(s_{178}, s_{179}...s_{288}) := (t_2, s_{178}, ...s_{287})$

12: **end for**

A comparison of the degrees of algebraic equations of Trivium-C with those of the original Trivium and Trivium/128 is presented in Figure 8.4 Similar to Trivium-B, alternate guessing of bits

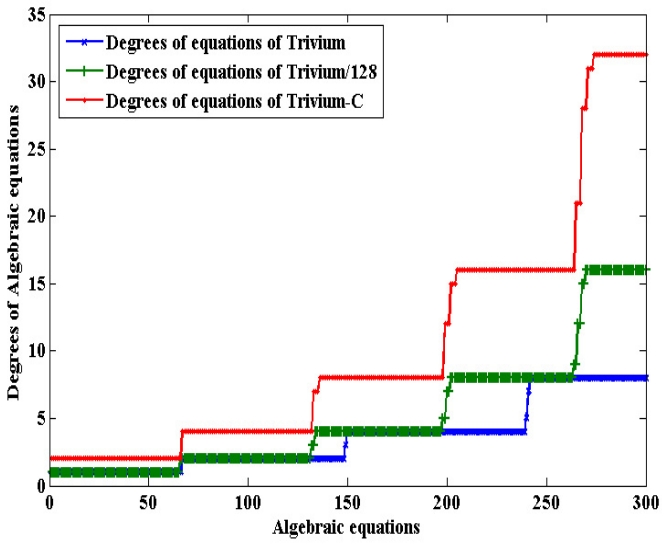

Figure 8.4: A Comparison of the degrees of Algebraic equations of original version of Trivium with the proposed modified version Trivium-B.

will not be helpful for Trivium-C. However, in this case, due to the additional product terms, even the selective guessing will not work. A detailed analysis is given in Section 8.4.

8.3.2 Fourth Modification of Trivium- Trivium-D

Another modification in the design of Trivium is proposed here, to increase the security level of cipher with the same number of internal state bits. As we have seen in the original design of Trivium, states are updated nonlinearly with the second-degree expressions. However, the output is simple XOR of three state bits. That is why linear equations are obtained in the first 66 clocking, and afterwards degrees are increased in steps. If the output function is also made nonlinear, then there will be no linear equation even at the start of the clocking. Here, the nonlinear update and the remaining design approach are not altered at all.

For illustration, we have selected simple majority function: $f(x_1, x_2, x_3) = x_1x_2 \oplus x_1x_3 \oplus x_2x_3$. In this case, we have added three product terms or in other words three AND gates but as a result we obtain much higher degree equations as compared to Trivium/128. Pseudo-code of the Trivium-D, with modification described above is given below:

procedure $Trivium - D(s_1, s_2, ...s_{288})$

1: **for** $t = 1$ to n **do**

2: $t_1 := s_{66} + s_{93}$

3: $t_2 := s_{162} + s_{177}$

4: $t_3 := s_{243} + s_{288}$

5: $z_t := t_1t_2 + t_1t_3 + t_2t_3$

6: $t_1 := t_1 + s_{91}.s_{92} + s_{171}$

7: $t_2 := t_2 + s_{175}.s_{176} + s_{264}$

8: $t_3 := t_3 + s_{286}.s_{287} + s_{69}$

9: $(s_1, s_2 ... s_{93}) := (t_3, s_1, ... s_{92})$

10: $(s_{94}, s_{95} ... s_{177}) := (t_1, s_{94}, ... s_{176})$

11: $(s_{178}, s_{179} ... s_{288}) := (t_2, s_{178}, ... s_{287})$

12: **end for**

Rise in the degrees of equations of Trivium-D with the original Trivium and Trivium/128 is given in Figure 8.5 The modified version Trivium-D also has second-degree terms of alternate state bits,

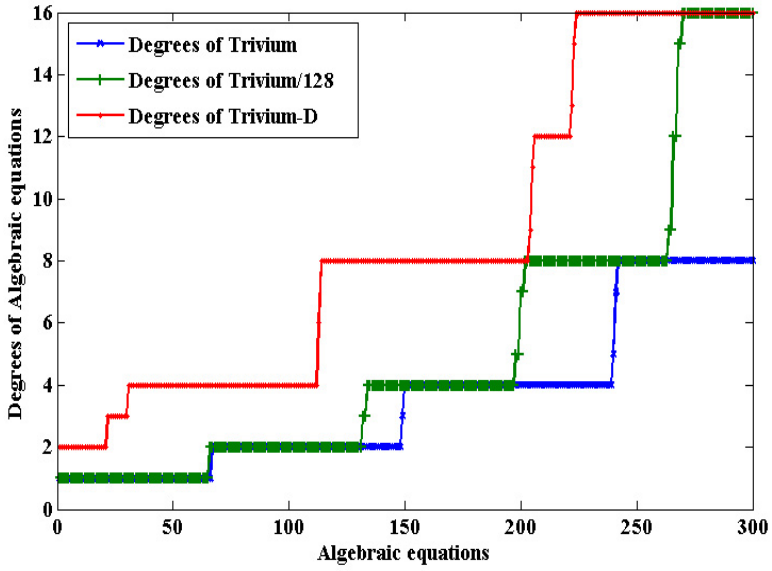

Figure 8.5: A Comparison of the degrees of Algebraic equations of original version of Trivium with the proposed modified version Trivium-D

but here alternate guessing of bits does not reduce the overall degrees of output equations. The reason naturally lies in the fact that there are many second-degree terms in the output polynomial attributed to the combining second-degree Boolean functions. Thus, even if a large number of state bits is guessed, linear equations or lower-degree equations are not obtained, and the system is not solvable.

The idea of Trivium-D is of placing a more complex Boolean function at the output instead of

using simple XOR of bits from three registers. Although, structures having LFSR with combining Boolean functions are quite an easy prey to algebraic attack, but in the case of Trivium all the three registers are non-linearly updated. We have selected a simple Boolean function of degree two, non-linearity two and correlation immunity zero to combine three bits; because along with improving the security level of the cipher, we want to maintain its good performance in hardware and software as near as possible to the original Trivium. In further researches, the structure of Trivium combined with some Boolean functions of different characteristics (non-linearity, correlation immunity, algebraic immunity etc.) can be tried with different trade-offs of security, speed and area.

8.4 Comparative Analysis of Proposed Modifications

This section is devoted to the analysis of proposed modified versions of Trivium so that changes in the design may be justified. We will also discuss the performance and the resistance of these designs against other cryptanalytic attacks.

8.4.1 Trivium-A

In Trivium-A, linear taps are shifted backwards. However, change of taps has to be done with some important considerations. First concern in this context is related to the period of the Trivium. Since the internal state of Trivium evolves in a nonlinear way therefore, it is hard to determine its period Cannière and Preneel (2005b). But still, as a design principle of Trivium all of its taps are divisible by three and the linear taps of the Trivium are selected in a way that if the AND gates are omitted (resulting in a completely linear scheme), then any key/IV pair would generate a stream with a period of at least 2_{96-3} Cannière and Preneel (2005b); Y.Tian et al. (2009). While modifying the structure to Trivium-A, the same design philosophy is maintained. The linear taps

of Trivium-A are also selected with the property that if the AND gates are omitted, then any initial state would generate a stream with a period of at least 2_{96-3}, because its characteristic polynomial is primitive of order 93.

Another important feature of Trivium is that its design provides a way to parallelize its operations. This is done by ensuring that any state bit is not used for at least 64 iterations after it has been modified. This way, up to 64 iterations can be computed at once, provided that the three AND gates and eleven XOR gates in the original scheme are duplicated, a corresponding number of times. Trivium-A, however, provides only 32-bit parallelism. A comparison of the performance of Trivium-A and other modifications with original Trivium is given in next section.

The overall rise in the degrees of equations is already given. Since, in this case the form of equations is not changed, therefore, alternate guessing of bits will simplify the equations. However, it has been experimentally verified that due to the presence of higher-degree equations, the system of equations will not be solved with half bits guessed. Since, the proposed modifications are aimed to show better resistance against algebraic recovery of internal state bits, therefore, for simulations we have used similar resources as have been used to mount a practical algebraic attack on Trivium 5.3.1.

8.4.2 Trivium-B

Linear taps of Trivium-B are modified similar to Trivium-A. Therefore, it has all the properties of Trivium-A as discussed in previous section. But the difference between these two modifications is in product terms. Therefore, the alternate guessed bits do not help finding lower degree equations in case of Trivium-B. Due to the structure of Trivium-B, some selective guessing of bits can be useful. As described earlier, the internal states of Trivium can be divided into three segments.

If alternate bits of two of the segments and some consecutive bits from the third segment are guessed, we may obtain some linear equations. Figure 8.6 shows a comparison of the degrees of original Trivium when state bits at alternate places are guessed with proposed Trivium-B when state bits at alternate and selective positions are known. It can be seen from the Figure 8.6 that

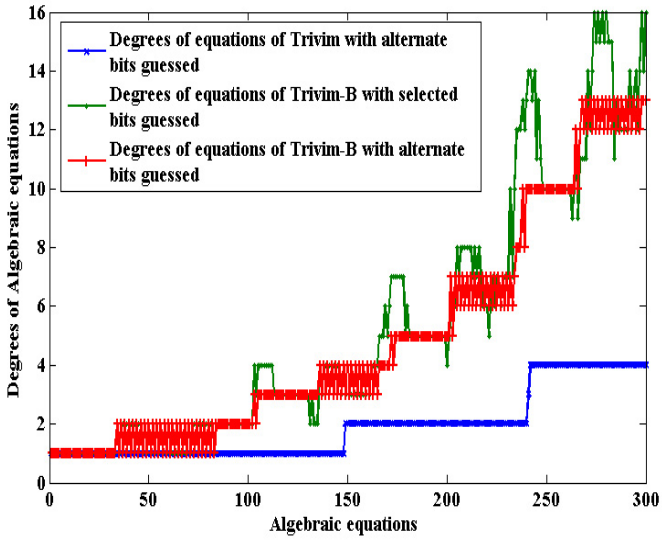

Figure 8.6: A Comparison of the degrees of Algebraic equations of original version of Trivium with the proposed modified version Trivium-B with alternate and selective bits guessed

as compared to the large number of linear equations obtained as a result of guessing state bits at alternate places of original Trivium, the degrees of equations of Trivium-B are much higher. We do not consider here the case of guessing more than half the state bits because in the case of the original Trivium, we can do with fairly less than half to recover the rest of the internal state bits. Thus, we conclude after trying a number of ways for guessing (with half of the bits guessed), Trivium-B does not give enough lower-degree equations that can be solved. It can thus be concluded that the Trivium-A and the Trivium-B provide resistance to the algebraic attack with guess and determine approach. Thus, the recovery of internal secret state bits of Trivium is more complex than before, so that it will have a better security margin.

8.4.3 Trivium-C and Trivium-D

Both of these versions have been proposed with additional gates to provide a security level of 128 bits. It is concluded after experiments with many options based on the structure of the equations that even after guessing a large number of state bits (a lot more than half), we do not obtain a simplified enough system that could be solved within resources similar to those that were used for within the similar resources as were used for the original Trivium and Trivium/128 5.3.1. Due to an additional product term in which alternate taps are not used alternate guessing of bits is not as useful in Trivium/128, Trivium-C and Trivium-D as in Trivium. But due to inclusion of non-linear terms in the earlier clocks, in case of Trivium-C and Trivium-D no linear terms appear in their case unlike Trivium/128. A comparison of degrees of Trivium/128, Trivium-C and Trivium-D with half bits guessed in all the three cases is shown in Figure 8.7.

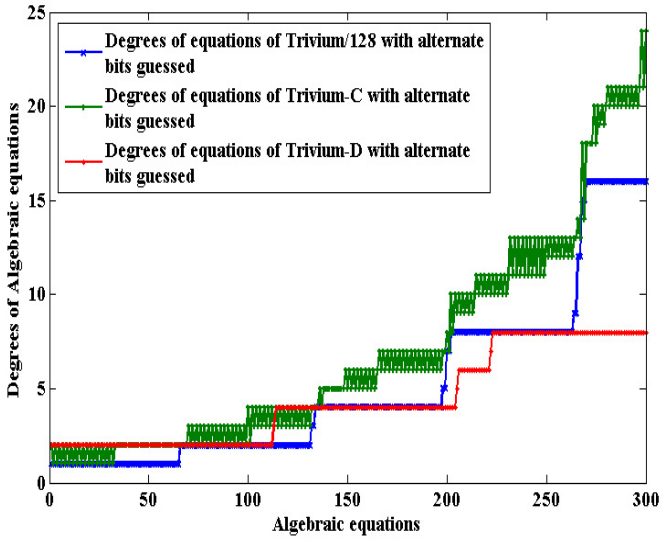

Figure 8.7: A Comparison of the degrees of Algebraic equations of Trivium/128 with the proposed modified version Trivium-C and Trivium-D with alternate bits guessed

In all the four modifications discussed above, we did not attempt to guess more than half the state

bits and, therefore, have not found the limit to the minimum number of state bits that need to be guessed to make the system of equations feasible to be solved. All the situations discussed here show better results in generating complex equations than original Trivium. Moreover, the proposed versions with additional AND gates are more difficult to solve than Trivium/128.

A comparison of the number of obtained linear equations for all the proposed modifications and Trivium and Trivium/128 is presented in Table 8.1. Although for Trivium/128, more linear equations can be obtained when guessed bits are not alternate but selected in a way based on its structure 5.4. But here we compare the case with alternate bits guessed. Only the number of linear equations is compared because the large number of linear equations obtained actually helps in solving the algebraic system obtained in remaining bits.

Table 8.1: A Comparison of the number of linear equations with all unknown and half guessed bits for proposed modifications with Trivium and Trivium/128

Version	Number of Linear equations	Number of Linear equations with alternate bits guessed
Trivium	66	148
Trivium/128	65	65
Trivium-A	33	118
Trivium-B	33	58
Trivium-C	0	16
Trivium-D	0	0

8.4.4 Resistance against Other Attacks

The proposed modifications of Trivium in this article aim mainly to increase the complexity of Algebraic attack against Trivium. Due to the structure of Trivium, using guess and determine approach a large number of its state bits can be recovered by solving algebraic equations. However, it can be prospected that while trying to increase the complexity of one attack, we may make our cipher vulnerable to any other attack. Attack scenarios in Maximov and Biryukov (2007) are built on the fact that non-linear equations are not obtained immediately as the clocking is started rather non-linearity comes into play after some delay, thus guessing the AND terms some linear equations are obtained to give some bits of the internal state. Since, all of our modifications attempt to decrease the linear equations, therefore, it is not difficult to see that proposed modifications will make the attack scenario of Maximov and Biryukov (2007) also more difficult.

Correlation attack against stream ciphers with linear feedback shift registers is very important, however, Trivium's state evolves in a nonlinear way. Correlations in Trivium have been considered in a few researches. Results given in Khazaei and Hassanzadeh (n.d.) and Turan and Kara (2007) discuss linear approximation attacks. Trivium is shown to be strong against linear sequential circuit approximation attack, in spite of the linearity of its output function and the fact that, except three, all of the next state functions are linear Khazaei and Hassanzadeh (n.d.). Two of our modifications have similar number of linear and non-linear relations that can result in similar difficulty in deriving linear relations, whereas two other proposed modifications have increased number of nonlinear relation that will increase the difficulty in deriving linear approximations with good correlation coefficient. Yet another approach of finding correlation may find any good correlation in proposed modification also.

We have analyzed our proposed modified designs theoretically against all existing attacks on Triv-

ium and found that complexity of most of the attacks remain unchanged, if not increased. Its main reason is that we have not changed the basic design philosophy of the Trivium cipher. It should also be noted that while proposing a design, no one can claim that no attack can ever be designed against it. Generally, claims are made on the existing attacks and their results. Therefore, the proposed designs are as vulnerable to some new attack or some existing attack designed in a different way as is the original Trivium or any other new cipher. To the best of our knowledge, there is no other existing successful attack on the cipher and therefore, this article mainly discusses the effect of proposed modifications on the recovery of internal state bits by exploiting its algebraic structure of the cipher.

8.4.5 Performance Comparison

In the original Trivium, no taps appear on the first 64 bits of each shift register, so each novel state bit is not used until at least 64 rounds after it is generated. This is the key to Trivium's software performance and flexibility in hardware. A straightforward hardware implementation of Trivium would use 3488 logic gates and produce one bit per clock cycle. However, because each state bit is not used for at least 64 rounds, 64 state bits can be generated in parallel at a slightly greater hardware cost of 5504 gates Cannière and Preneel (2005b).

Changing the taps of register may disturb its property of parallelism. As mentioned in previous sections, Trivium-A, Trivium-B and Trivium-C will support 32 bit parallelism, whereas Trivium-4 is suitable for 64-bit parallelism. A comparison of estimation of the gate count for different degrees of parallelization for Trivium, Trivium/128 and all the proposed modifications are given in Table 8.2. Estimation is done based on Cannière and Preneel (2005b) i.e., 12 NAND gates per Flip-op, 2.5 gates per XOR, and 1.5 gates per AND:

Table 8.2: Estimated gate counts of 1-bit to 64-bit hardware implementations.

	Components	1-bit	8-bit	16-bit	32-bit	64-bit
Trivium	Flip-flops	288	288	288	288	288
	AND gates	3	24	48	96	192
	XOR gates	11	88	176	352	704
	NAND gate count	3488	3712	3968	4480	5504
Trivium/128	Flip-flops	288	288	288	288	288
	AND gates	6	48	96	192	384
	XOR gates	15	120	240	480	960
	NAND gate count	3502	3828	4200	4944	6432
Trivium-A	Flip-flops	288	288	288	288	N/A
and Trivium-B	AND gates	3	24	48	96	
	XOR gates	11	88	176	352	
	NAND gate count	3488	3712	3968	4480	
Trivium-C	Flip-flops	288	288	288	288	N/A
	AND gates	6	48	96	192	
	XOR gates	15	120	240	480	
	NAND gate count	3502	3828	4200	4944	
Trivium-D	Flip-flops	288	288	288	288	288
	AND gates	6	24	48	96	192
	XOR gates	11	88	176	352	704
	NAND gate count	3492	3748	4040	4624	5792

8.5 Concluson

Trivium, a selected candidate for the final portfolio of the eSTREAM project, has a simple and elegant structure. Since its internal state is quite larger than its key-stream length, a natural question arises whether its bit security level can be increased with the same number of internal state bits. Based on the experiments, it was found that this can be achieved with small design modifications. We have proposed here some modified versions of Trivium to increase its bit security level. Compared with an earlier proposed tweak, Trivium/128, our modifications show better resistance against the algebraic cryptanalysis of Trivium with a larger bit security level. Two of our proposed modifications do not use any additional AND gates but still achieve better security margin than the original Trivium. Whereas, the other two proposed versions with additional AND gates show a much better security level than Trivium/128.

Chapter 9

Summary and Conclusions

9.1 Introduction

This chapter is meant to provide a summary of previous chapters so, as to conclude this work. It gives an overview of our proposed algebraic attack on different stream ciphers with non-linear update and also highlights the main contributions of this work.

The chapter is organized as follows. Next section summarizes the algebraic cryptanalysis of the selected stream ciphers presented in previous chapter. Following Section is about the proposed modifications in Grain-v1 and Trivium. Chapter will be concluded in last section.

9.2 Algebraic Cryptanalysis of Stream Ciphers A5/1, A5/2, Grain, Trivium and Mickey

Due to their extensive application in secure communication, many new stream ciphers have been proposed recently. Stream cipher project eSTREAM eSTREAM (2005) has played a very important role in identifying some new and promising stream ciphers. Designers of most of the newly proposed ciphers, try to establish that their proposed designs are strong against the existing cryptanalytic attacks. Also the design strategy, which has been victimized greatly, is generally avoided. Due to the success of algebraic attacks against stream ciphers, the design strategy of LFSR with high order Boolean functions, which is the ideal one for algebraic cryptanalysis, cannot be found in recently appreciated ciphers. In eSTREAM project, the ciphers like Sfinks and WG Braeken et al. (2005); Nawaz and Gong (2005) with this design philosophy were proposed

but Sfinks was not included in the 2nd phase, whereas WG was no more in the 3rd phase of the project. Successful analysis against Sfinks can be seen in Courtois (2005b).

It is thus important to identify also the design philosophy in stream ciphers that can show resistance to recovery of internal state bits using algebraic approach. If internal states of the registers involved in a cipher are updated non-linearly then the degrees of underlying equations of the cipher will increase with clocking of the registers. This may defeat the algebraic attack. However, different structures with non-linear update may offer different levels of security against algebraic attack.

This thesis was aimed at identifying the design strategy of stream ciphers, which can resist internal state recovery with algebraic approach. For this purpose, we consider the ciphers with non-linear update in their structure and analyze their resistance or otherwise vulnerability to algebraic cryptanalysis. We have analyzed and compared the design of the key generating mechanism, therefore, we have not considered the cipher as such, along with its initialization mechanism etc. The key generating parts of different ciphers are analyzed and are compared on the basis that how many of their internal state bits can be recovered by solving non-linear algebraic equations. Due to rise in the degree of equations with each clock, some of the internal state bits are guessed to recover the remaining bits. Our analysis reveals that due to some tactful guessing, one cipher may allow to recover a lot more bits than the other.

Grain, Trivium and Mickey are three candidate ciphers of eSTREAM project which have been included in its final portfolio. These ciphers have come under much scrutiny and concerning the existing cryptanalytic attacks, they satisfy the security margin that their designs offer. Our approach of solving equations with some guessed bits also confirms their security level. However, it is found that for Grain-v1 and Trivium, a large number of bits can be recovered on a PC in a

few seconds, thus showing a thin security margin with respect to the algebraic cryptanalysis. This reveals that it is also prospective that with some better resources algebraic cryptanalysis of these ciphers may show better results than brute force attack. For Grain-128 quite a few state bits can be recovered, and Mickey seems to be most resistant against this approach. Significance of this work also lies in making a comparison of these worthy members of the hardware portfolio of eSTREAM project based on their allowance to give away the number of internal state bits by solving algebraic equations. This analysis can also help in revising any of these designs to show better security margin so, it may be used with increased bits of secret key with the same number of internal state bits.

We also have included GSM encryption algorithms A5/1 and A5/2 in our analysis. Both of these ciphers have clock controlled structures with the difference that A5/1 uses clocking with mutually controlled registers whereas in A5/2 an independently and regularly clocked register controls the clocking of other three registers. Due to this fact A5/1, has been proved to be resistent to algebraic attack, whereas A5/2 is an easy prey for this attack. Structure of A5/2 has been extensively analyzed algebraically in literature, we, however, have applied an attack on A5/2 using Groebner basis. Due to this, data requirement in the form of frame numbers is reduced.

Grain-v1 although have a higher degree of Boolean function in comparison to the Grain-128, but overall almost 50% of the state bits can be recovered in case of Grain-v1, whereas Grain-128 only permits to recover 25%. This occurs because output bit, in case of Grain-128, is masked with a large number of variables from NLFSR. Consequently, all linear equations include several indeterminate values. Furthermore, the system cannot be solved for more than 64 unknowns. For Grain-v1, although there are some equations with very high degrees, still guessing half of the internal states gives enough linear equations in fewer variables. As a result, relatively more

variables can be recovered. Hence, Grain-1 hardly satisfies the security of margin of 80 bits whereas, Grain-128 has rather safe margin.

Trivium on the other hand, with all its elegance and ease of structure gives a large number of linear equations even without guessing a single bit. Moreover, the presence of alternate state bits in the algebraic equations suggests some ways of guessing the bits, which made the analysis easier. Thus, Trivium also gives a thin security margin and its design actually allows finding quite a number of state bits. This also suggests difficulty in increasing the security level of the cipher with same design and number of state bits. We have also shown that with our guessing approach, time complexity of retrieving bits from Trivium using F_4 algorithm implemented in Magma is less than the other solving methods. This research, however, can be further extended by solving the equations generated with our guessing approach with some other equations solving techniques like Raddum (n.d.).

Mickey uses the idea of irregularly clocking of registers. Although algebraic attack against the clock controlled stream ciphers has been studied with focus on ciphers in which one generator controls the clocking of other generators Al-Hinai et al. (2006). Whereas, ciphers that involve the clocking in which none of the registers are regularly clocked are quite resistant as is the case in A5/1. Mickey also combines irregular clocking and interdependence of registers for clocking and as a result, its algebraic equations gain very fast increase in the degree. Furthermore, the form of equations does not give any help in selectively guessing some of the bits to find others. Both of its registers are mutually dependent and guessing one or the other or in any other form does not at all help in gaining linear or low order equations to make the solution possible.

Algebraic cryptanalysis is although difficult to apply on stream ciphers in which the internal secret state bits are updated non-linearly but not impossible altogether. With some careful guessing,

some structures can reveal a large number of their secret state bits. However, if the registers are mutually interdependent, algebraic attack can be defeated even with guess and determine approach.

9.3 Some Suggestions for the Design of Grain-v1 and Trivium based on the Analysis

We have seen in the previous chapter that 80 bits of Grain-v1 can be recovered very easily thus giving a marginal security of Grain-v1. Therefore, we have proposed some design modifications in Grain-v1 so that its defense can be enhanced against algebraic attack without any increase in the internal state length or making any other big changes in its structure.

Generally, for a stream cipher, the number of internal state bits is important to decide its security level. Thus, for the cipher with internal state bits which are larger in number than the secret key bits, a natural question arises, whether the security level of the cipher can also be increased. Our analyses of these promising and interesting designs of ciphers also try to find the answer to these questions. For example, if we want to increase the security level of Trivium from 80 bits, should its internal states be increased also? Whereas, it already has a large number of internal state bits: 288 bits. But in the case of Mickey, no reasonable number of bits can be recovered algebraically at least within the on hand resources, thus showing its strength against recovery of internal state bits. In this case, with the same design and number of internal state bits, one can think of increasing the security level of the cipher. Obviously, here it is assumed that no other attack on the cipher exists, or steps to counter that attack are already taken. Our proposed modifications for Trivium aim to achieve an increase in the key bit length of Trivium. We have also proposed some changes in the design to improve its resistance to internal state recovery with the same key bit length. Our main effort is not to disturb the actual design philosophy of Trivium design.

9.4 Conclusion

Studying the vulnerability or otherwise, resistance of different cipher structures to different types of cryptanalytic attacks is an important task for design and cryptanalysis of both stream and block cipher systems. The thesis is about the algebraic cryptanalysis of those stream ciphers whose internal state is updated non-linearly. It has been concluded that while some structures are vulnerable to such attacks due to their structures while some other structures can defeat this attack due to their interdependence of state registers. In this chapter, we have summarized the analysis presented in previous chapters.

Bibliography

A. Bogodanov, T. E. and Rupp, A. (2003), A Hardware-Assisted Realtime Attack on A5/2 Without Precomputations, *in CHES'07*, Vol. 4727 of *Lecture Notes in Computer Science*, Springer-Verlag, pp.394–412.

Afzal, M. and Masood, A. (2007), Algebraic Attack on A5-type Irregularly Clocked Key Stream Generator, *in IMECS*, pp.670–674.

Afzal, M. and Masood, A. (2008a), Algebraic analysis of Trivium and Trivium/128, *Int. Journal. Electronic Security and Digital Forensics* **1**(4), 344–352.

Afzal, M. and Masood, A. (2008b), Algebraic Cryptanalysis of A NLFSR Based Stream Cipher, *in International Conference on Information & Communication Technologies: from Theory to Applications, ICTTA'08*, IEEE, pp.1–6.

Afzal, M. and Masood, A. (2008c), Experimental Results on Algebraic Analysis of Trivium and Tweaked Trivium, *in International Conference on Golbal e-Security*, Vol. 12 of *Communications in Computer and Information Science*, Springer Berlin Heidelberg, pp.93–101.

Afzal, M. and Masood, A. (2008d), On generating algebraic equations for A5-type key stream generator, *in Trends in Intelligent Systems and Computer Engineering*, Vol. 6 of *LNEE*, Springer-Verlag, pp.443–452.

Afzal, M. and Masood, A. (2008e), Resistance of Stream Ciphers to Algebraic Recovery of Internal Secret States, *in ICCIT '08: Proceedings of the 2008 Third International Conference on Convergence and Hybrid Information Technology*, IEEE Computer Society, Washington, DC, USA, pp.625–630.

Afzal, M. and Masood, A. (2009), Improving the Resistance of Grain-V1 against Algebraic Attack, *in Fourth International Conference on Computer Sciences and Convergence Information Technology, ICCIT'09*, IEEE Computer Society, Washington, DC, USA, pp.1086–1090.

Afzal, M. and Masood, A. (2010), Modifications in the Design of Trivium to Increase its Security Level, *Proceedings Pakistan Academy of Sciences* **In Press**.

Afzal, M., Kausar, F. and Masood, A. (2006), Comparative analysis of the structures of eSTREAM submitted stream ciphers, *in International Conference on Emerging Technologies, ICET '06*, IEEE, pp.245–250.

Afzal, M., Masood, A. and Shehzad, N. (2008), Improved results on algebraic cryptanalysis of A5/2, *in Global E-Security*, Vol. 12 of *Communications in Computer and Information Science, CCIS*, Springer Berlin Heidelberg, pp.182–189.

Al-Hiana, S., Baten, L. M. and Colbert, B. D. (2007), Mutually clock-controlled feedback shift registers provide resistance to algebraic attacks, 8th International Conference on Finite Fields and Applications (FQ8).

Al-Hinai, S., Baten, L. M., Colbert, B. D. and Wong, K. (2006), Algebraic Attacks on Clock-Controlled Stream Ciphers, *in ACISP'06*, *LNCS* **4058**, Springer-Verlag, pp.1–16.

Armknecht, F. (2004a), Improving Fast Algebraic attacks, *in Fast Software Encryption-2004*, Vol. 3017 of *Lecture Notes in Computer Science*, Springer-Verlag, pp.65–82.

Armknecht, F. (2004b), On the Existence of Low Degree Equations for Algebraic Attacks.

Armknecht, F. and Karuse, M. (2003), Algebraic Attacks on Combiners with memory, *in Advances in Cryptology-Crypto 2003*, Vol. 2729 of *Lecture Notes in Computer Science*, Springer-Verlag, pp.162–175.

Babbage, S. (n.d.), Some Thoughts on Trivium, http://www.ecrypt.eu.org/stream/.

Babbage, S. and Dodd, M. (n.d.a), The stream cipher MICKEY-128 (version 1),Algorithm specification issue 1.0, available at http://www.ecrypt.eu.org/stream/.

Babbage, S. and Dodd, M. (n.d.b), The stream cipher MICKEY (version 1),Algorithm specification issue 1.0, available at http://www.ecrypt.eu.org/stream/.

Barkan, E., Biham, E. and Keller, N. (2003), Instant Ciphertext-Only Cryptanalysis of GSM Encrypted Communication, *in Advances in Cryptology - Crypto'03*, Vol. 2729 of *Lecture Notes in Computer Science*, Springer-Verlag, pp.600–616.

Bellare, M. (ed.) (2000), *Fast Correlation through reconstruction of linear polynomials*, Vol. 1880 of *Lecture Notes in Computer Science*, Springer-Verlag.

Berbain, C., Gilbert, H. and Maximov, A. (2006), Cryptanalysis of Grain, *in Fast Software Encryption-FSE'06*, Vol. 4047 of *Lecture Notes in Computer Science*, Springer-Verlag, pp.15–29.

Biham, E. and Dunkelman, O. (2000), Cryptanalysis of the A5/1 GSM Stream Cipher, *in Progress in Cryptology-IndoCrypt'00*, Vol. 1977 of *Lecture Notes in Computer Science*, Springer-Verlag, pp.43–51.

Braeken, A., Lano, J., Mentens, N., Preneel, B. and Verbauwhede, I. (2005), SFINKS: A Synchronous stream cipher for restricted hardware environments.

Briceno, M., Goldberg, I. and Wagner, D. (1999), A Pedagogical Implementation of the GSM A5/1 and A5/2 Voice Privacy Encryption Algorithms, www.scard.org.

Buchberger, B. (1985), Groebner Bases: An Algorithmic Method in Polynomial Ideal Theory, *Multidimensional System Theory, Dordrecht* pp.184–232.

Cannière, C. D. (2006), Trivium: A Stream Cipher Construction Inspired by Block Cipher Design Principles, *in ISC*, pp.171–186.

Cannière, C. D. and Preneel, B. (2005a), Trivium: A Stream Cipher Construction Inspired by Block Cipher Design Principles, http://www.ecrypt.eu.org/stream/papersdir/2006/021.pdf.

Cannière, C. D. and Preneel, B. (2005b), Trivium Specifications, *in ECRYPT Stream Cipher Project Report 2005/030*.

C.J.A.Jansen (2004), Streamcipher Design: Make your LFSRs jump!, *in ECRYPT SASC, State of the Art in Stream Ciphers, workshop*, In the workshop record at http://www.isg.rhul.ac.uk/research/projects/ecrypt/stvl/sasc-record.zip.

Courtois, N. (2001), The Security of Hidden Field Equations (HFE), *in CT-RSA'01*, Vol. 2020 of *Lecture Notes in Computer Science*, Springer Berlin/Heidlberg, pp.266–281.

Courtois, N. (2002), Higher order Correlation attacks, XL algorithm and Cryptanalysis of Toyocrypt, *in International Conference on Information Security and Cryptology-ICISC 2002*, Vol. 2587 of *Lecture Notes in Computer Science*, Springer-Verlag, pp.182–199.

Courtois, N. (2003), Fast Algebraic Attacks on Stream Ciphers with Linear Feedback, *in Advances in Cryptology-Crypto03*, *LNCS* **2729**, pp.176–194.

Courtois, N. (2005a), Algebraic Attacks on Combiners with Memory and Several Outputs, *in Information Security and Cryptology-ICISC 2004*, Vol. 3506 of *Lecture Notes in Computer Science*, Springer Berlin/Heidlberg, pp.3–20.

Courtois, N. (2005b), Cryptanalysis of Sfinks, *in ICISC*, pp.261–269.

Courtois, N. and Meier, W. (2003), Algebraic attacks on Stream Ciphers with Linear Feedback, *in Advances in Cryptology - Eurocrypt 2003*, Vol. 2656 of *Lecture Notes in Computer Science*, Springer-Verlag, pp.345–359.

Courtois, N. and Pieprzyk, J. (2002), Cryptanalysis of Block Ciphers with Overdefined Systems of Equations, *in Advances in Cryptology-Asiacrypt02*, Vol. 2501 of *Lecture Notes in Computer Science*, Springer Berlin/Heidlberg, pp.267–287.

Davies, D. W. (ed.) (2000), *On a fast Correlation attack certain stream ciphers*, Vol. 1880 of *Lecture Notes in Computer Science*, Springer-Verlag.

E. Barkan, E. B. and Keller, N. (2003), Instant Ciphertext-Only Cryptanalysis of GSM Encrypted Communication, *in Advances in Cryptology-Crypto'03*, Vol. 2729 of *Lecture Notes in Computer Science*, Springer-Verlag, pp.66–616.

Ekdahl, P. and Johansson, T. (2003), Another Attack on A5/1, *IEEE Transactions on Information Theory* **49**(1), 284–288.

eSTREAM (2005), ECRYPT Stream Cipher Project.

Faugère, J. C. (1999), A new efficient algorithm for computing Gröbner basis (F 4), *Journal of Pure and Applied Algebra* **139**(1-3), 61–88.

Faugère, J. C. (2002), A New Efficient Algorithm for Computing Groebner Bases without Reduction to Zero (F5), *in International Symposium on Symbolic and Algebraic Computation-ISSAC'02*, ACM Press, pp.75–83.

Faugere, J. C. and Joux, A. (2003), Algebraic Cryptanalysis of Hidden Field Equation (HFE) Cryptosystem Using Groebner Bases, *in Advances in Cryptology-Crypto'03*, Vol. 2729 of *LNCS*, Springer-Verlag, pp.44–60.

Feistel, H. (1970), Block Cipher Cryptographic System.

Feistel, H. (1973), Cryptography and Computer Privacy, *Scientific American* **228**, 15–23.

Froeberg, R. (1999), *An Introduction to Groebner Bases*, Vol. 3 of *Graduate Studies in Mathematics*, John Wiley & Sons.

Golic, J. (1997), Cryptanalysis of Alleged A5 Stream Cipher, *in Advances in Cryptology-Eurocrypt'97*, Vol. 1233 of *Lecture Notes in Computer Science*, Springer-Verlag, pp.239–255.

Gunter, C. (ed.) (1988), *Fast Correlation attacks on stream ciphers*, Vol. 330 of *Lecture Notes in Computer Science*, Springer-Verlag.

Gwenole Ars, Jean-Charles Faugere, H. I. M. K. and Sugita, M. (2004), Comparison Between XL and Groebner Basis Algorithm, *in Advances in Cryptology - Asia 2004*, Vol. 3329 of *Lecture Notes in Computer Science*, Springer Berlin/Heidelberg, pp.338–353.

Hell, M., Johansson, T., Johansson, T. and Meier, W. (n.d.), A Stream Cipher Proposal: Grain-128, *in Information Theory, 2006 IEEE International Symposium on*.

I. Goldberg, D. W. and Green, L. (1999), The Real-Time Cryptanalysis of A5/2, the Rump Session of Crypto99.

J.D.Golic and Conner, L. O. (1994), Embedding and Probabilistic Correlation Attacks on Clock-Controlled Shift Registers, *in Advances in Cryptology-Eurocrypt'94*, Vol. 950 of *Lecture Notes in Computer Science*, Springer-Verlag, pp.90–100.

Johansson, T. and Jonsson, F. (1999), Improved fast correlation attacks on stream ciphers via convolutional codes, *in* J. Stern (ed.), *Advances in Cryptology - Eurocrypt'99*, Vol. 1592 of *Lecture Notes in Computer Science*, Springer-Verlag, pp.347–362.

Johansson, T. and Jonsson, F. (2000), Fast Correlation Attacks Through Reconstruction of Linear Polynomials, *in Advances in Cryptology - Crypto 2000*, Vol. 1880 of *Lecture Notes in Computer Science*, Springer-Verlag, pp.300–315.

Khazaei, S. and Hassanzadeh, M. (n.d.), Linear Sequential Circuit Approximation of the Trivium Stream Cipher, available at http://www.ecrypt.eu.org/stream/.

Kipnis, A. and Shamir, A. (1999), Cryptanalysis of the HFE Public Key Cryptosystem by Relinearization, *in Advances in Cryptology, Crypto99*, Vol. 1666 of *Lecture Notes in Computer Science*, Springer Berlin/Heidlberg, pp.19–30.

Krause, M. (2002), BDD-Based Cryptanalysis of Keystream Generators, *in Advances in Cryptology-Eurocrypt'02*, Vol. 2332 of *Lecture Notes in Computer Science*, Springer-Verlag, pp.222–237.

M. Blaze, R. J. A. and Roe, M. (1999), Notes on A5, http://jya.com/crack-a5.htm.

M. Hell, T. J. and Meier, W. (2005a), Grain - A Stream Cipher for Constrained Environments, ECRYPT Stream Cipher Project Report 2005/010.

M. Hell, T. Johansson, A. M. and Meier, W. (2005b), A Stream Cipher Proposal, ECRYPT Stream Cipher Project Report, 2005.

Makoto Sugita, M. K. and Imai, H. (n.d.), Relation Between XL and Groebner Basis Algorithm, *in IACR e-print Server*, http://eprint.iacr.org//2004/112/.

Maple (n.d.), Mathematics and Engineering software, Maplesoft.

Maximov, A. (June, 2006), Some Words on Cryptanalysis of Stream Ciphers, PhD thesis, Department of Information Technology, Lund University, Sweden.

Maximov, A. and Biryukov, A. (2007), Two Trivial Attacks on Trivium, *in Selected Areas in Cryptography*, pp.36–55.

McDonald, C., Charnes, C. and Pieprzyk, J. (2009), An Algebraic Analysis of Trivium Ciphers based on the Boolean Satisfiability Problem.

Meier, W. and Staffelbach, O. (1989), Fast correlation attacks on stream ciphers, *Journal of Cryptology* **1(3)**, 159–176.

Meier, W., Pasalic, E. and Carlet, C. (2004), Algebraic attacks and decomposition of Boolean functions, *in Advances in Cryptology - EUROCRYPT 2004*, Vol. 3027 of *Lecture Notes in Computer Science*, Springer-Verlag, pp.474–491.

Nawaz, Y. and Gong, G. (2005), The WG Stream Cipher.

NESSIE (1999), New European Schemes for Signatures, Integrity, and Encryption.

Petrovic, S. and Fuster-Sabater, A. (2000), Cryptanalysis of the A5/2 Algorithm, IACR ePrint Report 200/52.

Pornin, T. and Stern, J. (2000), Software-Hardware Trade-Offs: Application to A5/1 Cryptanalysis, *in CHES'2000*, Vol. 1965 of *Lecture Notes in Computer Science*, Springer-Verlag, pp.318–327.

Preneel, B. (ed.) (2000), *Improved fast correlation attacks using parity-check equations of weight 4 and 5*, Vol. 1807 of *Lecture Notes in Computer Science*, Springer-Verlag.

Raddum, H. (n.d.), Cryptanalytic results on Trivium, http://www.ecrypt.eu.org/stream/.

Rueppel, R. A. (1986), *Analysis and design of stream ciphers*, Springer-Verlag New York, Inc., New York, NY, USA.

S. Khazaei, M. H. and Kiaei, M. (n.d.), Distinguishing attack on Grain, ECRYPT Stream Cipher Project Report, 2005.

Shannon, C. E. (1949), Communication Theory of Secrecy Systems, *Bell System Technical Journal* **28**, 656–715.

Siegenthaler, T. (1984), Correlation Immunity of non-linear combining functions for Cryptographic applications, *IEEE Transactions on Information Theory* **30**, 776–780.

Siegenthaler, T. (1985), Decrypting a class of stream ciphers using ciphertext only, *IEEE Transactions on Computers* **34**(1), 81–85.

Steve Babbage, Christophe De Canni‘ere, A. C. C. C. H. G. T. J. M. P. B. P. V. R. and Robshaw, M. (2008), The eSTREAM Portfolio.

System, M. C. A. (n.d.), Magma, http://magma.maths.usyd.edu.au/.

T A. Biryukov, A. S. and Wagner, D. (2001), Real Time Cryptanalysis of A5/1 on a PC, *in*

Fast Software Encryption-FSE'00, Vol. 1978 of *Lecture Notes in Computer Science*, Springer-Verlag, pp.1–18.

Turan, M. S. and Kara, O. (2007), Linear approximation for 2-round trivium, *in First International Conference on Security of Information and Networks, SIN*, Trafford Publishing, pp.96–105.

Wiener, M. J. (ed.) (1999), *Fast Correlation attacks based on Turbo code techniques*, Vol. 1666 of *Lecture Notes in Computer Science*, Springer-Verlag.

Y.Tian, Chen, G. and Li, J. (2009), On the design of Trivium, http://eprint.iacr.org/2009/431.